TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Mona Aghababaeetafreshi

Software Defined Radio Solutions for Wireless Communications Systems

Julkaisu 1595 • Publication 1595

Tampere 2018

Tampereen teknillinen yliopisto. Julkaisu 1595
Tampere University of Technology. Publication 1595

Mona Aghababaeetafreshi

Software Defined Radio Solutions for Wireless Communications Systems

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Sähkötalo Building, Auditorium SA203, at Tampere University of Technology, on the 23rd of November 2018, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2018

Doctoral candidate: Mona Aghababaeetafreshi
Laboratory of Electronics and Communications Engineering
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Tampere, Finland

Supervisor: Mikko Valkama, Professor
Laboratory of Electronics and Communications Engineering
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Tampere, Finland

Instructor: Jarmo Takala, Professor
Laboratory of Pervasive Computing
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Tampere, Finland

Pre-examiner: Fernando H. Gregorio, Professor
Electric and Computer Engineering
The National University of South
Bahía Blanca, Argentina

Pre-examiner and opponent: Luigi Carro, Professor
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil

Opponent: Janne Janhunen, D.Sc.
Solmu Technologies
Oulu, Finland

ISBN 978-952-15-4254-1 (printed)
ISBN 978-952-15-4260-2 (PDF)
ISSN 1459-2045

ABSTRACT

Wireless technologies have been advancing rapidly, especially in the recent years. Design, implementation, and manufacturing of devices supporting the continuously evolving technologies require great efforts. Thus, building platforms compatible with different generations of standards and technologies has gained a lot of interest. As a result, software defined radios (SDRs) are investigated to offer more flexibility and scalability, and reduce the design efforts, compared to the conventional fixed-function hardware-based solutions.

This thesis mainly addresses the challenges related to SDR-based implementation of today's wireless devices. One of the main targets of most of the wireless standards has been to improve the achievable data rates, which imposes strict requirements on the processing platforms. Realizing real-time processing of high throughput signal processing algorithms using SDR-based platforms while maintaining energy consumption close to conventional approaches is a challenging topic that is addressed in this thesis.

Firstly, this thesis concentrates on the challenges of a real-time software-based implementation for the very high throughput (VHT) Institute of Electrical and Electronics Engineers (IEEE) 802.11ac amendment from the wireless local area networks (WLAN) family, where an SDR-based solution is introduced for the frequency-domain baseband processing of a multiple-input multiple-output (MIMO) transmitter and receiver. The feasibility of the implementation is evaluated with respect to the number of clock cycles and the consumed power. Furthermore, a digital front-end (DFE) concept is developed for the IEEE 802.11ac receiver, where the 80 MHz waveform is divided to two 40 MHz signals. This is carried out through time-domain digital filtering and decimation, which is challenging due to the latency and cyclic prefix (CP) budget of the receiver. Different multi-rate channelization architectures are developed, and the software implementation is presented and evaluated in terms of execution time, number of clock cycles, power, and energy consumption on different multi-core platforms.

Secondly, this thesis addresses selected advanced techniques developed to realize inband full-duplex (IBFD) systems, which aim at improving spectral efficiency in today's congested radio spectrum. IBFD refers to concurrent transmission and reception on the same frequency band, where the main challenge to combat is the strong self-interference (SI). In this thesis, an SDR-based solution is introduced, which is capable of real-time mitigation of the SI signal. The implementation results show possibility of achieving real-time sufficient SI suppression under time-varying environments using low-power, mobile-scale multi-core processing platforms.

To investigate the challenges associated with SDR implementations for mobile-scale devices with limited processing and power resources, processing platforms suitable for hand-held devices are

selected in this thesis work. On the baseband processing side, a very long instruction word (VLIW) processor, optimized for wireless communication applications, is utilized. Furthermore, in the solutions presented for the DFE processing and the digital SI canceller, commercial off-the-shelf (COTS) multi-core central processing units (CPUs) and graphics processing units (GPUs) are used with the aim of investigating the performance enhancement achieved by utilizing parallel processing.

Overall, this thesis provides solutions to the challenges of low-power, and real-time software-based implementation of computationally intensive signal processing algorithms for the current and future communications systems.

PREFACE

This thesis is based on the research work carried out during the years 2014–2017 in the Laboratory of Electronics and Communications Engineering, Tampere University of Technology, Tampere, Finland. I would like to gratefully acknowledge the financial support I received from the Tampere University of Technology Graduate School (during the years 2014–2017), Nokia Foundation, and Tuula and Yrjö Neuvo Research Fund. The research work carried out for this thesis was also partially supported by the Finnish Funding Agency for Technology and Innovation (TEKES) under the Parallel Acceleration (ParallaX) project.

First and foremost, I would like to sincerely thank my supervisors Prof. Mikko Valkama and Prof. Jarmo Takala for their invaluable help, guidance, and support during these years. It has been a privilege to learn from their extensive knowledge and experience. I am also very grateful to Prof. Luigi Carro and Prof. Fernando Gregorio for acting as the pre-examiners of this thesis, and providing their valuable comments and insights. Furthermore, I wish to thank Prof. Luigi Carro and D.Sc. Janne Janhunen for agreeing to act as the opponents in the public examination of this thesis.

I am also very grateful to D.Sc. Toni Levanen, D.Sc. Pekka Jääskeläinen, and D.Sc. Dani Korpi for sharing their deep knowledge in this field with me along the way. In addition, I wish to thank my co-authors Lasse Lehtonen, Matias Koskela, D.Sc. Juha Yli-Kaakinen, and Maliheh Soleimani for our fruitful collaborations.

I would also like to thank all my friends, especially Parinaz, Kamiar, Nader, Afsaneh, Saeed, and Sajjad who have always lifted my spirits and made life much more fun from the very first days of my studies in Tampere.

Finally, my deepest and most sincere thanks go to my parents, who have never stopped supporting and encouraging me in life. None of this would have been possible without their endless love and support. And last but not least, I would like to express my warmest thanks to Orod. There are no words to describe how grateful I am to have him both in my personal and professional life.

Espoo, October 2018

Mona Aghababaeetafreshi

Contents

Acronyms

5G	fifth generation
6G	sixth generation
ALU	arithmetic logic unit
ANPI	average noise power indicator
AP	access point
ASIC	application specific integrated circuit
ASIP	application specific instruction-set processor
BCC	binary convolutional codes
BW	bandwidth
COTS	commercial off-the-shelf
CP	cyclic prefix
CPU	central processing unit
CS	carrier sensing
CSD	cyclic shift diversity
DFE	digital front-end
DRAM	dynamic random-access memory
DSP	digital signal processing
FDD	frequency-division duplexing
FEC	forward error correction
FFT	fast Fourier transform
FIR	finite impulse response

FPGA field-programmable gate array

GI guard interval

GPP general-purpose processor

GPU graphics processing unit

GSM global system for mobile communications

HARQ hybrid automatic repeat request

HSPA high speed packet access

HT high throughput

I/Q in-phase/quadrature

IBFD inband full-duplex

IEEE Institute of Electrical and Electronics Engineers

IFFT inverse fast Fourier transform

ISM industrial, scientific, and medical

L-LTF non-HT long training field

L-SIG non-HT SIGNAL field

L-STF non-HT short training field

LAN local area networking

LDPC low-density parity check

LMMSE linear minimum mean square error

LMS least mean squares

LNA low-noise amplifier

LS least square

LTE long-term evolution

LUT lookup table

MAC medium access control

MIMO multiple-input multiple-output

MU multi-user

NDP null data packet

NR new radio

OFDM orthogonal frequency-division multiplexing

OpenCL open computing language

PA power amplifier

PE processing element

PHY physical

QAM quadrature amplitude modulation

RCPI received channel power indicator

RF radio frequency

RSNI received signal to noise indicator

SDR software defined radio

SI self-interference

SIMD single instruction multiple data

SoC system on chip

SPMD single program, multiple data

STBC space-time block codes

SVD singular value decomposition

TDD time-division duplexing

UE user equipment

UMTS universal mobile telecommunications system

VHDL VHSIC hardware description language

VHT very high throughput

VHT-LTF VHT long training field

VHT-SIG-A VHT signal A field

VHT-SIG-B VHT signal B field

VHT-STF VHT short training field

VLIW very long instruction word

WCDMA wideband code division multiple access

WLAN wireless local area networks

LIST OF PUBLICATIONS

[P1] M. Aghababaeetafreshi, L. Lehtonen, M. Soleimani, M. Valkama and J. Takala, "IEEE 802.11AC MIMO transmitter baseband processing on customized VLIW processor," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, May 4-9, 2014, pp. 7500-7504, doi: 10.1109/ICASSP.2014.6855058.

[P2] M. Aghababaee Tafreshi, L. Lehtonen, T. Levanen, M. Valkama and J. Takala, "IEEE 802.11ac MIMO receiver baseband processing on customized VLIW processor," in *IEEE Workshop on Signal Processing Systems,* Belfast, UK, Oct. 22-24, 2014, pp. 1-6, doi: 10.1109/SiPS.2014.6986092.

[P3] M. Aghababaeetafreshi, L. Lehtonen, T. Levanen, M. Valkama and J. Takala, "IEEE 802.11ac MIMO transceiver baseband processing on a VLIW Processor", *Journal of Signal Processing Systems*, Oct 2016, 85(1), pp. 167–182, doi: 10.1007/s11265-015-1032-2.

[P4] M. Aghababaeetafreshi, J. Yli-Kaakinen, T. Levanen, V. Korhonen, P. Jääskeläinen, M. Renfors, M. Valkama and J. Takala, "Parallel processing intensive digital front-end for IEEE 802.11ac receiver," in *49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, 8-11 Nov, 2015, pp. 1619-1626, doi: 10.1109/ACSSC.2015.7421422.

[P5] M. AghababaeeTafreshi, M. Koskela, D. Korpi, P. Jääskeläinen, M. Valkama and J. Takala, "Software defined radio implementation of adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio," in *IEEE Global Conference on Signal and Information Processing*, Washington, DC, USA, 7-9 Dec, 2016, pp. 733-737, doi: 10.1109/GlobalSIP.2016.7905939.

[P6] M. Aghababaeetafreshi, D. Korpi, M. Koskela, P. Jääskeläinen, M. Valkama and J. Takala, "Software defined radio implementation of a digital self-interference cancellation method for inband full-duplex radio using mobile processors," *Journal of Signal Processing Systems*, Oct 2018, 90(10), pp. 1297–1309, doi: 10.1007/s11265-017-1312-0.

CHAPTER 1

INTRODUCTION

With the rapid evolution of wireless network standards, emerges the need for more flexible radios, which can easily adapt to new technologies. For this reason, the concept of software defined radios (SDRs) was introduced in the early 1990s [1], and is still being developed to this day. The aim of SDRs is to liberate the radio implementation from the restrictions of a hard-wired system, which, as a result, introduces a high degree of flexibility and programmability that cannot be achieved with the traditional solutions using fixed-function hardware-based approaches.

In addition to offering higher flexibility, SDRs reduce the design efforts considerably, and thus time-to-market cycles are shortened compared to application-specific solutions. Furthermore, designing platforms that could support different existing, and even upcoming standards increases significantly the costs and complexity of the implementation process. Consequently, SDR-based approaches can, in general, minimize the efforts and costs of design, fabrication, testing, and maintenance [2, 3].

With SDRs, the characteristics of the radio system, such as bandwidth (BW), air interface protocol, and functionality would no longer be static. However, the behavior of the radio can be dynamically modified through software. In other words, a new software upgrade can update the operation of the radio, rather than having to replace the whole hardware. This facilitates the development of a multi-standard, multi-band, and multi-functional systems, which, for example, can operate with different carrier frequencies, BWs, modulation schemes, and coding rates [4]. As a result, a future can be realized, in which the radio system can be re-configured in the field to operate in a different frequency band or transmission scenario. Furthermore, SDRs promise to facilitate deployment of new features and functionalities with the rapid development of technologies.

Some of the characteristics of an SDR architecture include re-programmability, scalability, and flexibility. Additionally, SDRs are usually associated with commercially available and affordable platforms, which support low cost and fast market delivery goals. SDR platforms range from general-purpose processors (GPPs) and graphics processing units (GPUs) to digital signal processing (DSP) cores, each offering different degrees of flexibility and development cycles. Fig. 1.1 provides a perspective into the trade-off between reconfigurability and development time for some known platforms [4, 5]. One important advantage of GPPs is the very low programming complexity, which allows for much faster prototyping compared to the other platforms. DSPs, on the other hand, offer great processing performance, however, they are considered less general

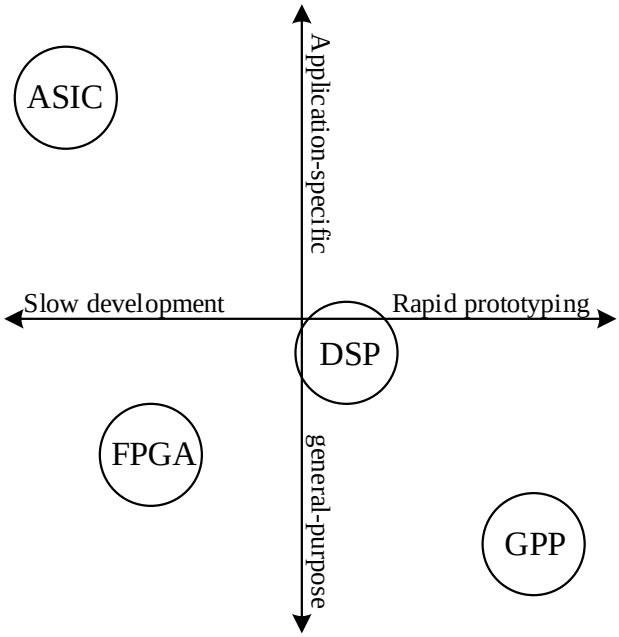


Figure 1.1: Trade-off between reconfigurability and development cycles in different platforms

purpose than field-programmable gate arrays (FPGAs) and GPPs, as they are optimized for digital signal processing algorithms.

In today's radio systems, programmable dedicated hardware is typically used for the main processing tasks, particularly at physical layer, while control tasks are carried out using GPPs. However, ideally, in an entirely software-defined system, all the radio functionality could be implemented on general-purpose processing platforms, and only antennas, basic amplification and coarse filtering stages, and digital-to-analogue and analogue-to-digital converters would be added. Realizing such systems comes with many challenges which need to be studied and overcome.

Application specific hardware platforms are believed to provide better performance compared to flexible software-based solutions as they are tailored to a dedicated functionality. Furthermore, being customized for specific operations means less area on the silicon, and consequently lower power consumption. Thus, the feasibility of the software-based solutions as a more flexible, yet efficient, alternative to the application specific integrated circuits (ASICs) is an interesting research area.

With the ever-growing amount of wireless data and its applications, wireless networks are constantly evolving to meet the demands for higher capacity and more efficiency. One of the most widely used wireless connectivity standards is the Institute of Electrical and Electronics Engineers (IEEE) 802.11 wireless local area networks (WLAN) family, which is continuously developing. Thus, utilization of software-based solutions in this area can considerably accelerate the progress, improve service life-cycle, and portability to other platforms.

The state of the art technology in the WLAN family is the IEEE 802.11ac. The 802.11ac amendment is designed to significantly improve the throughput to above gigabit ranges. This is achieved by using wider bandwidth, up to 160 MHz, in the 5 GHz industrial, scientific, and medical (ISM) band. Furthermore, higher order modulation, up to 256-quadrature amplitude modulation (QAM), and multi-user (MU) multiple-input multiple-output (MIMO) with up to eight

spatial streams are defined in the standard specification [6, 7]. A software-based implementation of the IEEE 802.11ac transceiver can offer high degree of flexibility in terms of adjusting the bandwidth, modulation order, coding rate, and MIMO configuration.

As available spectral resources for wireless communications are scarce, pursuing high spectral efficiency is a common target for most of the wireless standards. As a result, different technologies are being studied and developed which help to improve the efficiency of spectrum use, e.g., inband full-duplex (IBFD) communications. Full-duplex communication systems can utilize the spectral resources more efficiently by transmitting and receiving simultaneously on the same frequency within the same device, while in more ordinary duplexing methods, transmission and reception are based on sharing either the time-domain (time-division duplexing (TDD)) or frequency-domain (frequency-division duplexing (FDD)) resources. Thus, IBFD systems can potentially double spectral efficiency compared to traditional duplexing systems. However, deployment of these systems comes with challenge, particularly the inherent self-interference (SI) [8–11].

The cancellation of the SI signal is, in theory, rather simple as the transmitted signal is known by the transceiver. However, in practice, it is far more challenging since the overall effective coupling channel is not accurately known [12]. Furthermore, the system should dynamically adapt to the constantly changing environment, especially around a mobile device, which could benefit from the flexibilities offered by SDRs.

1.1 Objectives and Scope of the Work

This thesis addresses current and future wireless technologies and the challenges related to their implementation. The demanding requirements of these technologies, such as very high throughput, has led to very strict timing constraints for the implementation platforms. These constraints are even more challenging in case of mobile terminals with less processing and power resources. In the work carried out in this thesis, we target processing platforms which are suitable for hand-held devices.

This research work evaluates the implementation of different computationally intensive digital signal processing algorithms from physical (PHY) layer baseband and digital front-end (DFE) processing of the IEEE 802.11ac standard to full-duplex communications systems on different mobile-scale platforms. The solutions are examined in terms of performance, power, and energy consumption to investigate, the extent to which these solutions can be utilized in today's communications systems.

In this thesis work, a very long instruction word (VLIW) processor, specifically designed for wireless communication applications, is selected for the IEEE 802.11ac baseband processing. For the DFE channelization concept and the SI canceller implementation, multi-core general-purpose central processing units (CPUs) and GPUs are used to investigate the performance enhancement achieved by utilizing parallel processing. This topic has gained a lot of interest since clock rate scaling and aggressive uniprocessor performance scaling has reached its limits [13].

The main target of the research carried out during the work of this thesis is to develop and analyze software-based solutions for the implementation of the aforementioned computationally intensive algorithms on selected processors. Mainly, commercial off-the-shelf (COTS) platforms are adopted, which highlight the benefits of SDR based implementations. Then, the feasibility of the proposed solutions for achieving real-time operation is investigated. Furthermore, power and energy consumption are measured to evaluate the viability of the implementations.

The feasibility of the proposed solutions is studied by measuring the performance of the implementations. The objective is to deliver reprogrammable solutions that can provide real-time processing,

while consuming relatively low power/energy suitable for mobile devices. The SDR-based implementation is declared feasible when execution times and consumed power/energy are less or comparable with the traditional fixed-function implementations. In case of mobile-scale devices, where processing and energy resources are more limited, the requirements are even more strict. This thesis proposes software-based solutions that can meet the tight timing and power/energy consumption requirements of today's wireless standards while offering high flexibility.

1.2 Main Results and Outline

The main contributions of the Thesis can be highlighted as follows:

- A software-based solution for the MIMO transmitter and receiver baseband processing conforming to the IEEE 802.11ac standard is proposed, and the feasibility of achieving a real-time operation using a customized VLIW processor is shown by manually optimizing the implementation and exploiting the processor intrinsic instructions [P1][P2][P3].
- A DFE concept to divide the 80 MHz bandwidth of the IEEE 802.11ac is introduced, which uses circular and linear filtering based multi-rate channelization architectures. Additionally, an SDR implementation on mobile-scale COTS CPUs and GPUs is presented and analyzed, which is optimized by exploiting the parallel resources of the processors [P4].
- A self-adaptive nonlinear digital self-interference cancellation method for full-duplex transceivers is presented. Furthermore, an SDR based implementation is proposed for the SI canceller, which optimally parallelizes the computing resources of multi-core CPUs and GPUs for better performance [P5][P6].
- In all of the above cases, implementations are evaluated and the performance results are reported.

This thesis is divided into five chapters. Chapter 2 provides an introduction to the wireless technologies investigated in the scope of this thesis and presents the implemented algorithms. Chapter 3 describes the proposed implementations for IEEE 802.11ac baseband and DFE processing, where the main existing challenges are explained and solutions are provided. Then the performance of the implementations is evaluated and the results are reported in terms of execution time, and power/energy consumption. Chapter 4 gives a detailed description of the implementation methods for an adaptive digital SI canceller method used in IBFD systems. The corresponding challenges are pointed out and addressed, and finally the implementation results are discussed and analyzed. Chapter 5 presents the conclusions and discusses open issues and future directions.

1.3 Author's Contributions

The Author of the Thesis has been the main author in all the publications [P1]-[P6]. In [P1]-[P3], the Author of this thesis has done mathematical algorithm modifications to reduce the computational complexity of the algorithms for implementation purposes. The author has carried out the software implementation for the algorithms and further manual optimizations for better performance on the selected platforms using the processor's intrinsic instructions. Additionally, all the performance measurements have been done by the Author.

In [P4], the Author has designed and optimized the open computing language (OpenCL) kernels implementing the channelization filters. The Author has also carried out the measurements related to the implementation performance.

In [P4]-[P6], the Author has done the software implementations and manual optimizations for parallel processing on the selected multi-core platforms. The author has modified parts of the original methods with the aim of reducing the computational complexity of the algorithms to fit the available processing resources of the selected COTS platforms. Furthermore, the performance evaluations for the implementations were carried out by the Author.

CHAPTER 2

WIRELESS TECHNOLOGIES

Wireless communication systems continue to evolve to provide faster, more reliable and more energy efficient connectivity to Internet and other wireless applications. Based on the current predictions, the overall amount of wireless data will increase exponentially in the coming years. This, along with the numerous new wireless applications emerging everyday have resulted in the need for great advances and improvements in the future generations of wireless standards. In this chapter, first the IEEE 802.11ac, a leading WLAN technology is introduced. Then, addressing one of the most important issues in wireless networks, i.e. limited spectral resources, IBFD communication systems, and the corresponding challenges and solutions are described.

2.1 WiFi and IEEE 802.11ac

WiFi is a local area networking (LAN) technology for wireless connectivity, which provides indoor broadband coverage for fixed, portable, and mobile stations. WiFi is developed by IEEE standards association and promoted by the WiFi Alliance®. First IEEE 802.11™ standard was published in 1999, and since then it has been growing through different amendments to meet the high demands for more traffic, increasing number of devices, and new applications [14].

2.1.1 History of WiFi

Introduced in 1999, the IEEE 802.11b with maximum raw data rate of 11 Mbps using the 2.4 GHz band, established a basis for the WiFi industry. However, with the growing popularity, and thus the big market for WiFi, grew the expectations for higher data rates, better quality, and more security [15]. Thus, new amendments had to be developed to add support for higher data density and new applications.

The next amendment, the IEEE 802.11a, operated in the 5 GHz band and increased the throughput to 54 Mbps. However, due to lack of backward compatibility with the 2.4 GHz band used in 802.11b devices, it required two radios, and thus failed to gain a big market.

The amendment which followed 802.11a was the IEEE 802.11g. Similar to the 802.11b, this amendment used the 2.4 GHz band, and was able to achieve data rates up to 54 Mbps using

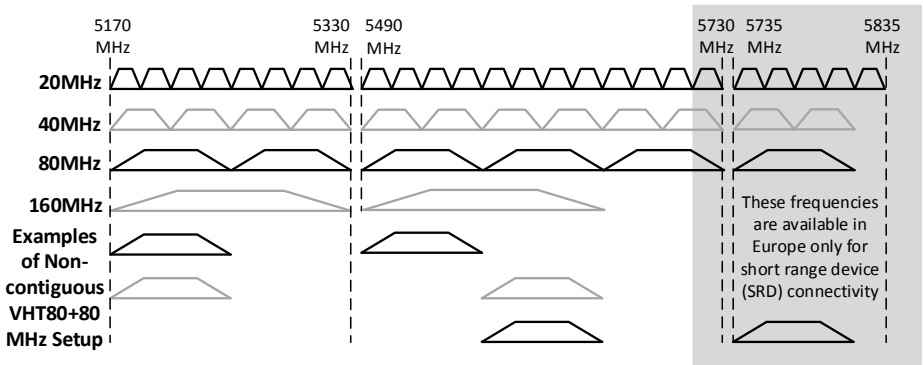


Figure 2.1: The different channelization configurations for the IEEE 802.11ac at the 5 GHz band

orthogonal frequency-division multiplexing (OFDM). Unlike the 802.11a, the 802.11g was backward compatible with 802.11b, and became a big success.

As WiFi Continued to develop, the IEEE 802.11n amendment was introduced with higher data rates, bigger range, improved security, and more reliability. The 802.11n standardized the use of MIMO and was able to yield a throughput of 150 Mbps. Both 2.4 and 5 GHz bands were supported by this amendment and it took advantage of a 40 MHz bandwidth. The 802.11n was also known as the high throughput (HT) amendment.

Next section describes the IEEE 802.11ac, also referred to as the very high throughput (VHT) amendment, and its main improvements compared to the 802.11n. The higher performance delivered by the 802.11ac opened the door to various new application areas. As an example, high definition video streaming is made possible, which was a challenge for the existing 802.11n devices.

2.1.2 IEEE 802.11ac Features

The IEEE 802.11ac amendment aims to provide extremely higher throughput and better user experience quality compared to its predecessor. For this reason, the 802.11ac has adopted several new techniques to improve its performance, some of which are briefly introduced in the following.

More Channel Bonding The IEEE 802.11ac supports channel bonding up to 160 MHz. Thus, two non-overlapping adjacent 40 MHz channels can be used to form an 80 MHz channel. Furthermore, two 80 MHz channels can be used to form either a contiguous or non-contiguous 160 MHz channel. The use of the 160 MHz channel is not mandatory in this amendment. The possibility of a non-contiguous channel setup provides more flexibility for channel assignment in 802.11ac [16]. Examples of different channelization configurations for the 802.11ac at 5 GHz band can be seen in Fig. 2.1.

Mandatory 5 GHz Operation The previous amendments mostly operated in the 2.4 GHz band, and the 802.11n supported the optional use of 5 GHz band. However, as a result of the legacy WiFi devices crowding the 2.4 GHz band, this band is susceptible to higher interference. Furthermore, more non-overlapping channels are available in the 5 GHz band, which provides more flexibility for channel assignment. Thus, the IEEE 802.11ac mandates operation in the 5 GHz band [17].

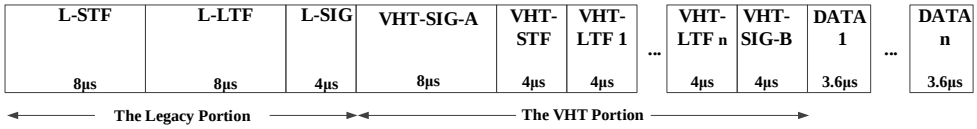


Figure 2.2: IEEE 802.11ac PHY layer packet structure assuming short GI for the data symbols.

Higher Order Modulation The IEEE 802.11ac allows use of denser modulation schemes compared to its predecessors. Increasing from the 64-QAM used in the 802.11n, this amendment supports constellation configurations up to 256-QAM, yielding up to 33% increase in data rates [7].

Higher Order MIMO While the HT amendment, the first to introduce MIMO in WiFi specifications, allowed four spatial streams, the VHT adds support for up to eight spatial streams. This improvement can double the total network throughput compared to the 802.11n [17].

Multi-User MIMO The IEEE 802.11ac is the first amendment to introduce MU-MIMO. This feature allows multiple users to be scheduled in the same time-slot, which means that the access point transmitter can simultaneously transmit multiple packets to multiple users by diving the available streams among the stations.

2.1.3 IEEE 802.11ac PHY Packet Structure

The PHY layer packet defined in the IEEE 802.11ac consists of a header and a data part. The header itself comprises of non-HT (legacy) and VHT fields. The legacy field includes non-HT short training field (L-STF), non-HT long training field (L-LTF), and non-HT SIGNAL field (L-SIG). VHT signal A field (VHT-SIG-A), VHT short training field (VHT-STF), VHT long training field (VHT-LTF), and VHT signal B field (VHT-SIG-B) are the VHT specific portions. The PHY layer packet structure defined in this specification can be seen in Fig. 2.2, where the duration of each field, assuming a short guard interval (GI) for the data symbols, is shown.

2.1.4 IEEE 802.11ac Baseband Processing

This section describes the transceiver functional blocks of the IEEE 802.11ac baseband processing which were included in the works carried out for this thesis.

2.1.4.1 Transmitter Processing

In this section, only the transmitter processing related to DATA symbols is covered, as the processing of preamble symbols is rather straight-forward, and thus less computationally intensive. Fig. 2.3 illustrates the building blocks related to the processing of a DATA symbol, where the grey colored blocks are not included in the work of this thesis.

It is assumed that, first, the bits are scrambled, and then encoded in the forward error correction (FEC) unit, using either binary convolutional codes (BCC) or low-density parity check (LDPC) codes. The high complexity encoding is to be carried out on a separate hardware accelerator. Then, the rest of the processing is performed using our proposed software-based solution. At the final stage, inverse fast Fourier transform (IFFT) is executed on a dedicated hardware, after which the time-domain processing starts.

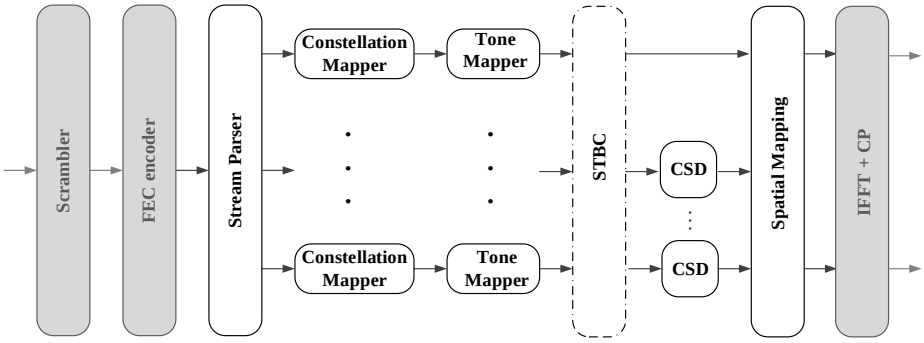


Figure 2.3: Principal block diagram of the IEEE 802.11ac transmitter baseband processing. The blocks marked with grey color are not included in the software-based implementation.

The functionality of the transmitter blocks, implemented in the work of this thesis, are briefly explained in the following.

Stream Parsing To create the required number of spatial streams, the incoming bits from the encoder are divided into the number of spatial streams (N_{ss}). Each stream receives a group of s bits in a round robin fashion, as defined in (2.1).

$$s = max\left\{1, \frac{N_{BPSCS}}{2}\right\} \tag{2.1}$$

Here, N_{BPSCS} is the number of coded bits per single subcarrier for each spatial stream and is equivalent to the modulation order.

According to (2.1), having 256-QAM modulation, s would be equal to four. Thus, in case of $N_{ss} = 2$, each of the two streams receives a block of four bits in each round, thus dividing the incoming bit stream $[y_0, y_1, y_2, \ldots, y_i]$ into streams of $[y_0, y_1, y_2, y_3, y_8, y_9, \ldots]$, and $[y_4, y_5, y_6, y_7, y_{12}, y_{13}, \ldots]$.

Modulation Mapping Each group of N_{BPSCS} bits received from the stream parser are mapped to constellation points. BPSK, 16-QAM, 64-QAM, or 256-QAM with Gray-coded mapping can be used as the modulation scheme. The resulting complex numbers are normalized by a factor of K_{mod}. Thus, output values are calculated as:

$$d_{mod} = (I + Qj) \times K_{mod}. \tag{2.2}$$

With 256-QAM, the modulation scheme employed in this work, we have: $K_{mod} = \frac{1}{\sqrt{170}}$ [7].

LDPC Tone Mapping Since LDPC coding is selected as the FEC method here, LDPC tone mapping should be carried out after the modulation. The purpose of tone mapping is to achieve full frequency diversity in 80 MHz and 160 MHz bands. The tone mapper places the received consecutive constellations at tones with distance D_{TM} from each other. The LDPC tone mapping distance parameter D_{TM} is constant for each bandwidth. Fig. 2.4 illustrates the tone mapping process assuming 80 MHz bandwidth ($D_{TM} = 9$).

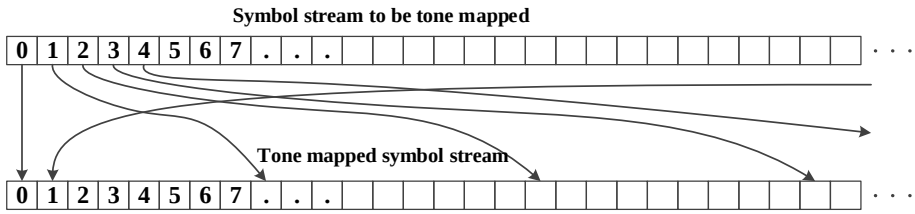


Figure 2.4: The tone mapping process for 80 MHz bandwidth

STBC Coding A generalized version of the well-known Alamouti codes [18], space-time block codes (STBC), is performed at this stage. Space-time coding exploits spatial and temporal diversity by transmitting multiple copies of a data stream over different antenna streams. This helps to compensate for multipath fading, and as a result, higher reliability and robustness of data transmission is achieved. STBC codes are orthogonal and can achieve full diversity.

Having a two-antenna configuration with one spatial stream, STBC coding is performed as follows. At time instance t_1, antenna 1 and antenna 2 transmit symbols x_1 and x_2, respectively. Then, at time instance $t_2 = t_1 + T$, symbols $-x_2^*$ and x_1^* are transmitted from antenna 1 and antenna 2, respectively. Here, T is the symbol duration, and x^* represents the complex conjugate of symbol x.

CSD Cyclic shifts, referred to as cyclic shift diversity (CSD), are performed on the signal in order to decorrelate space-time streams. As a result, there will be a large phase shift between the signals transmitted from different antennas, and unwanted beamforming is avoided. These phase shifts are translated to delays in time-domain. Different shift values are used for VHT and non-VHT fields.

Spatial Mapping This last step performs a mapping between the space-time streams and the antennas. Thus, the final signals to be transmitted are produced at this stage. In the scope of this work, space-time streams are mapped to the transmit antennas directly after getting scaled by a normalization factor. Scaling factor is defined as $\sqrt{N_{STS}}$, where N_{STS} is the number of space-time streams.

2.1.4.2 Receiver Processing

In the receiver side, it is assumed that, first, time-domain processing is carried out, and then fast Fourier transform (FFT) is performed in a dedicated hardware unit. The samples then go through the processing implemented in the work of this thesis, shown in Fig. 2.5 and Fig. 2.6. Finally, at the last stage, LDPC decoding and descrambling are carried out in a separate hardware.

As the received preamble symbols are used to measure received signal quality and estimate the channel, the processing carried out on these symbols in the receiver is much more complex than

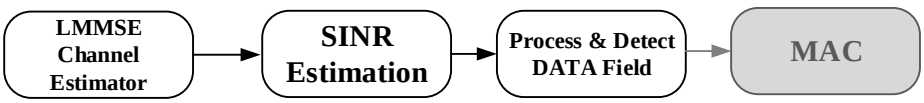


Figure 2.5: Overall logical block diagram of the receiver side processing

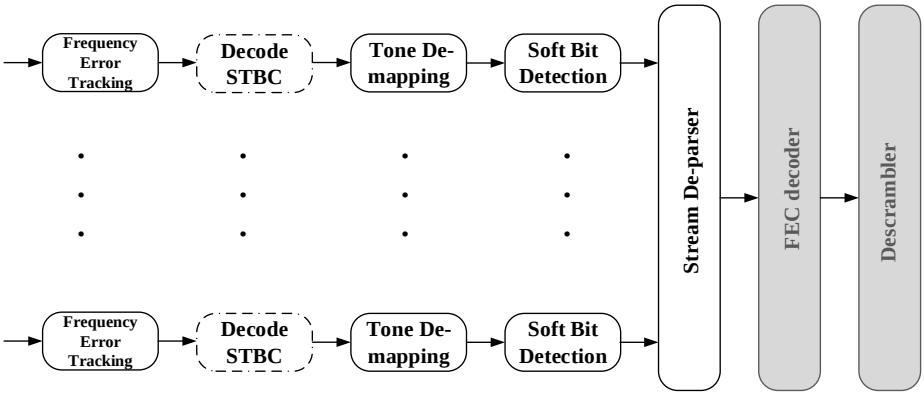


Figure 2.6: Principal block diagram of the IEEE 802.11ac receiver baseband processing. The blocks marked with grey color are not included in the software-based implementation.

the transmitter side. Thus, the receiver processing related to the preamble symbols is described in this section in addition to the DATA processing.

The stream de-parser and LDPC tone de-mapper blocks in the receiver simply reverse the functionality of their counter blocks in the transmitter. Thus, they are left out from the discussions of this section.

The functionality of the receiver blocks, implemented in the work of this thesis, are briefly explained in the following.

SINR Estimation SINR measurement is carried out in the receiver to evaluate the quality of the received signal. This information then can be sent to the transmitter to adjust the transmit power, or the modulation and coding scheme. The calculated indicators are received channel power indicator (RCPI), average noise power indicator (ANPI), and the received signal to noise indicator (RSNI).

RCPI RCPI measures the received radio frequency (RF) power in the channel, which includes the signal, noise, and interference. It is measured over the DATA portion of the received frame. However, if a null data packet (NDP) is received, VHT-SIG-B or VHT-LTF symbols can be used. The power is measured over all active non-pilot subcarriers and is then averaged over all antennas. Thus, RCPI can be calculated as:

$$RCPI = \frac{1}{N_{R_x} N_{sc} N_d} \sum_{R_x} \sum_{d} \sum_{i \in I} \left| y_{R_x,d,i} \right|^2 , \tag{2.3}$$

where $R_x = 1, 2, \ldots, N_{R_x}$ is the receiver antenna index, $d = 1, 2, \ldots, N_d$ is the DATA symbol index, and $N_{sc} = |I_{active,\ non-pilot\ subcarriers}|$, where $|I|$ is the cardinality of the set I.

In this implementation, calculation is carried out after reception of each DATA symbol, and the average is updated with every symbol until reception of the whole frame is completed.

ANPI ANPI is a medium access control (MAC) layer indicator, which calculates average noise plus interference power, and is used for symbol detection. ANPI can be measured when the channel is idle as defined by three simultaneous conditions: 1) the virtual carrier

sensing (CS) mechanism indicates idle channel, 2) the station (STA) is not transmitting a frame, and 3) the STA is not receiving a frame [14].

ANPI can be calculated over any received frame during any period. In this work, it is calculated over non-DC null subcarriers. L-STF and VHT-STF are selected for ANPI measurement, as these symbols include many zero-valued subcarriers, in addition to the non-active carriers. This means that any change in their values can be considered as noise. Thus, ANPI can be written as:

$$ANPI = \frac{1}{N_{R_x} N_z} \sum_{N_{R_x}} \sum_{N_{i \in I}} |y_{R_x,i}|^2 , \tag{2.4}$$

where $N_z = |I_{active,\ zero-valued\ pilot\ subcarriers}|$.

For the purpose of this measurement, we assume that the accuracy of time and frequency synchronization is enough, in a way that zero-valued subcarriers would be only carrying noise.

RNSI RSNI is the signal to noise plus interference ratio of a received frame as defined in [14]. Having calculated RCPI and ANPI, RNSI can be written as:

$$RSNI = 10 \log_{10} \frac{RCPI - ANPI}{ANPI}, \tag{2.5}$$

where ANPI and RCPI are in linear scale. Averaging RCPI, ANPI, and RSNI can help to improve stability. It should be noted that averaging must be done closely in time for high correlation.

Channel Estimation The preamble symbols used for channel estimation are L-LTF and VHT-LTF symbols. Two channel estimates are required in the receiver, one for the precoded and one for the non-precoded symbols. For this reason, VHT-LTF symbols are precoded as defined in [7], whereas L-LTF symbols are not.

Channel estimator for the legacy part Having transmitted the training symbols $x_{L-LTF,k}$, the received L-LTF symbols per symbol index t, $t = [1, 2]$, per subcarrier index k, $k \in I_{active,\ non-pilot,\ L-LTF\ subcarriers}$, can be written as:

$$\begin{aligned} \mathbf{y}_{k,t} &= \mathbf{H}_k \mathbf{x}_{L-LTF,k} + \mathbf{n}_{k,t} \\ &= x_{L-LTF,k} \begin{bmatrix} \frac{1}{N_{T_x}} \sum_{j=1}^{N_{T_x}} h_{1,j} \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{N_{T_x}} \sum_{j=1}^{N_{T_x}} h_{N_{R_x},j} \end{bmatrix} + \mathbf{n}_{k,t} \\ &= x_{L-LTF,k} \mathbf{h}_{eff,k} + \mathbf{n}_{k,t}, \end{aligned} \tag{2.6}$$

where $\mathbf{H}_k$ is a $(N_{R_x} \times N_{T_x})$ complex channel matrix, $\mathbf{x}_{L-LTF,k}$ is a $(N_{T_x} \times 1)$ real vector containing $x_{L-LTF,k}$ symbols, $\mathbf{n}_{k,t}$ is a $(N_{R_x} \times 1)$ complex Gaussian noise vector, and $\mathbf{h}_{eff,k}$, expanded in (2.6), is the legacy $(N_{R_x} \times 1)$ effective channel vector.

Having transmitted only one and minus one symbols in the two L-LTF symbols $t = [1, 2]$, the effective least square (LS) channel estimate per subcarrier k can be written as:

$$\hat{\mathbf{h}}_{LS,k} = \frac{x_{L-LTF,k}}{2} \sum_{t=1}^{2} \mathbf{y}_{k,t}. \tag{2.7}$$

The calculated LS channel estimate is used for linear minimum mean square error (LMMSE) channel estimation, FFT smoothing and wiener filtering.

Now, the LMMSE channel estimate per subcarrier k can be calculated using the LS channel estimate from (2.7) as:

$$\begin{aligned}\hat{\mathbf{h}}_{LMMSE,k} &= \hat{\mathbf{h}}_{LS,k}\hat{\mathbf{h}}_{LS,k}^{H} \\ &\times (\hat{\mathbf{h}}_{LS,k}\hat{\mathbf{h}}_{LS,k}^{H} + \frac{\sigma_n^2}{2} I_{N_{R_x}})^{-1}\hat{\mathbf{h}}_{LS,k}\end{aligned}. \tag{2.8}$$

Here $\mathbf{h}^H$ is the Hermitian transpose of vector $\mathbf{h}$, and σ^2 is the noise variance.

Channel estimator for the VHT part Using the above calculated legacy channel estimates, the L-SIG and VHT-SIG-A symbols can be detected. Thus, the number of transmitted VHT-LTF symbols ($N_{VHT-LTF}$) will be known for the VHT channel estimation.

As defined by [7], these symbols are precoded with precoding matrix $\mathbf{P}$. Furthermore, precoder matrix $\mathbf{Q}_j$ $j \in I_{active,\ VHT-LTF\ subcarriers}$ may also be applied to VHT-LTF symbols. Having transmitted the training symbols $x_{VHT-LTF,k}$, the received VHT-LTF symbols per symbol index t, $t = [1, \ldots, N_{VHT-LTF}]$, per subcarrier index k, $k \in I_{active,\ non-pilot,\ VHT-LTF\ subcarriers}$, can be written as:

$$\begin{aligned}\mathbf{y}_{k,t} &= \mathbf{H}_k\mathbf{Q}_k\mathbf{P}(:,t)x_{VHT-LTF,k} + \mathbf{n}_{k,t} \\ &= \mathbf{H}_{eff,k}\mathbf{P}(:,t)x_{VHT-LTF,k} + \mathbf{n}_{k,t}.\end{aligned} \tag{2.9}$$

The VHT-LTF symbols are precoded by matrix $\mathbf{P}$ and averaged over $N_{VHT-LTF}$ symbols to achieve effective channel estimates per space time stream. For clearer presentation, received samples on subcarrier k from all R_x antennas, and VHT-LTF symbols are put into a column vector. Thus, the received VHT-LTF symbols can be shown as:

$$\begin{aligned}\mathbf{y}_k &= \begin{bmatrix} \mathbf{y}_{k,1} \\ . \\ . \\ . \\ \mathbf{y}_{k,N_{VHt-LTF}} \end{bmatrix} \\ &= (\mathbf{P} \otimes \mathbf{I}_{N_{R_x}})^T \begin{bmatrix} \mathbf{h}_{eff,k,1} \\ . \\ . \\ . \\ \mathbf{h}_{eff,k,N_{VHT-LTF}} \end{bmatrix}. \\ &\times x_{VHT-LTF,k} + \mathbf{n}_k.\end{aligned} \tag{2.10}$$

Here $\mathbf{P} \otimes \mathbf{I}_{N_{R_x}}$ represents Kronecker tensor product of Matrices $\mathbf{P}$ and $\mathbf{I}_{N_{R_x}}$. Now the

received VHT-LTF training symbols after decoding diversity coding can be written as:

$$\tilde{\mathbf{y}}_k = \frac{1}{N_{VHT-LTF}}(\mathbf{P} \otimes \mathbf{I}_{N_{R_x}}) \begin{bmatrix} \mathbf{y}_{k,1} \\ . \\ . \\ . \\ \mathbf{y}_{k,N_{VHT-LTF}} \end{bmatrix} \tag{2.11}$$
$$= x_{VHT-LTF,k} \begin{bmatrix} \mathbf{h}_{eff,k,1} \\ . \\ . \\ . \\ \mathbf{h}_{eff,k,N_{VHT-LTF}} \end{bmatrix} + \mathbf{z}_k,$$

where $\mathbf{z}_k \in \mathcal{CN}(0, \frac{\sigma_n^2}{N_{VHT-LTF}})$. Thus, the effective LS channel estimate per subcarrier k can be given as:

$$\hat{\mathbf{H}}_{eff,LS,k} = x_{VHT-LTF,k} \hat{\tilde{\mathbf{Y}}}_k, \tag{2.12}$$

where $\hat{\tilde{\mathbf{Y}}}_k$ contains weighted received symbols $\tilde{\mathbf{y}}_k$ from all N_{R_x} antennas.

Using the effective LS channel estimate from (2.12) and (2.8), the LMMSE channel estimate for the VHT part can be expressed as:

$$\hat{\mathbf{H}}_{eff,LMMSE,k} = \hat{\mathbf{H}}_{eff,LS,k} \hat{\mathbf{H}}_{eff,LS,k}^H \tag{2.13}$$
$$\times (\hat{\mathbf{H}}_{eff,LS,k} \hat{\mathbf{H}}_{eff,LS,k}^H + \sigma_n^2 I_{N_{R_x}})^{-1} \hat{\mathbf{H}}_{eff,LS,k}.$$

Equation (2.13) should be simplified to avoid the complicated computations involved in the matrix inversion required for LMMSE channel estimation. We stack the columns of the effective $(N_{R_x} \times N_{T_x})$ LS channel estimate for $N_{R_x} = N_{T_x} = 2$ on top of each other in a way that $\hat{\mathbf{h}}_{LS} = [h_1, h_2, h_3, h_4]^T$. Thus, LMMSE channel estimate can be re-written as:

$$\hat{\tilde{\mathbf{h}}}_{LMMSE} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \begin{bmatrix} h_1^* h_2^* h_3^* h_4^* \end{bmatrix}$$
$$\times \left(\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \begin{bmatrix} h_1^* & h_2^* & h_3^* & h_4^* \end{bmatrix} + \mathbf{N} \right)^{-1} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}, \tag{2.14}$$

where h^* is the complex conjugate of h, and $\mathbf{N} = \sigma_n^2 \mathbf{I}_{N_{R_x}}$. Using the Shannon-Morrison law [19], (2.14) can be simplified to:

$$\hat{\tilde{\mathbf{h}}}_{LMMSE} = \frac{h_1 h_1^* + h_2 h_2^* + h_3 h_3^* + h_4 h_4^*}{\sigma_n^2 + h_1 h_1^* + h_2 h_2^* + h_3 h_3^* + h_4 h_4^*} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$
$$= \frac{\hat{\mathbf{h}}_{LS} \hat{\mathbf{h}}_{LS}^T \times conj(\hat{\mathbf{h}}_{LS})}{\sigma_n^2 + \hat{\mathbf{h}}_{LS}^T \times conj(\hat{\mathbf{h}}_{LS})}. \tag{2.15}$$

Same solution can be applied to reduce the computational complexity associated with matrix inversion of a $N_{R_x} = N_{T_x} = 4$ MIMO antenna configuration.

Frequency Domain Pilot Based Residual Frequency Error Tracking This procedure is carried out to estimate and correct the frequency error of the received symbols. The frequency error on the received pilot subcarriers is estimated by calculating the mean of phase angle differences between the current pilots and the ones from the preceding symbol.

As the received pilots can have low power due to the frequency selective channel fading, we define a weighting vector for the pilots to reduce the degradation of phase rotation estimates. Denoting the power of each received subcarrier as $\sigma^2_{I_P(N_p)}$, the weighting vector $\mathbf{w}$ can be written as:

$$\mathbf{w} = \frac{1}{\sum_{i=1}^{N_p} \sigma^2_{I_P(i)}} \left[\sigma^2_{I_P(1)} \sigma^2_{I_P(2)} \cdots \sigma^2_{I_P(N_p)}\right]^T, \tag{2.16}$$

where N_p is the number of pilot subcarriers in I_P set ($N_p = |I_P|$).

The phase rotation estimate between pilot subcarriers of two consecutive symbols at time instances t and $t-1$, produced as a result of frequency error $\mathbf{F}_{error}$, can be written as:

$$\hat{\Theta}_t = \mathbf{w}^T \left(arg\left(\mathbf{P}_{t-1}\right) - arg\left(\mathbf{P}_t\right)\right), \tag{2.17}$$

where $arg\,(x)$ returns the argument of the complex number x, and t, $t = [1, \ldots N_t]$ is the data symbol index. Here, data symbol zero ($t = 0$) is the VHT-SIG-B symbol.

Now frequency error at symbol index t can be written as:

$$\hat{\mathbf{F}}_{error,t} = \frac{1}{2\pi t\left(N_s + N_{GI}\right)} \sum_{i=1}^{t} \hat{\Theta}_i, \tag{2.18}$$

where N_s denotes the number of subcarriers, and N_{GI} is the number of GI samples. The phase rotation estimates are averaged over the symbols, and the accuracy of the estimate is improved by the end of the DATA field.

Thus, the frequency error of received symbols per subcarrier k can be corrected by:

$$\hat{\mathbf{y}}_k = exp\left(j \sum_{i=1}^{t} \hat{\Theta}_i\right) \mathbf{y}_k. \tag{2.19}$$

Symbol Detection Symbol detection is carried out using the LMMSE channel estimates obtained by (2.13). First the symbol detection matrix per subcarrier k is calculated as:

$$\mathbf{D}_{coeff,k} = (\hat{\mathbf{H}}_k^H \hat{\mathbf{H}}_k + \sigma_n^2 \mathbf{I}_{N_{STS}})^{-1} \hat{\mathbf{H}}_k^H, \tag{2.20}$$

where $\hat{\mathbf{H}}_k$ is the LMMSE channel estimate and N_{STS} is the number of space time streams.

Now the received symbols per subcarrier k can be detected as:

$$\hat{\mathbf{x}} = \mathbf{D}_{coeff,k} \mathbf{y}_k. \tag{2.21}$$

In case of STBC coding, channel estimates and received symbols should be defined accordingly. For instance, in case of $N_{T_x} = N_{R_x} = 2$, and STBC coding, $\hat{\mathbf{H}}_k$ is written as:

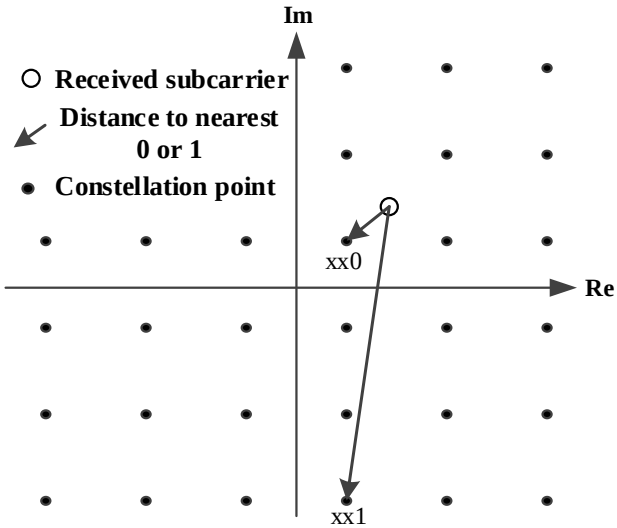


Figure 2.7: Soft bit detection

$$\hat{\mathbf{H}}_k = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix}, \tag{2.22}$$

where matrix elements h_{ij} denote the channel from i_{th} receiver antenna to j_{th} transmitter antenna.

Furthermore, the received symbols vector $\mathbf{y}_k$, having $N_{T_x} = N_{R_x} = 2$ antenna setting and STBC coding, is written as:

$$\mathbf{y}_k = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{21}^* \\ y_{22}^* \end{bmatrix}, \tag{2.23}$$

where y_{ij} is the received symbol at time slot i on receiver antenna j.

Channel estimate matrix and received symbols vector from (2.22) and (2.23) can be similarly extended for a $N_{T_x} = N_{R_x} = 4$ antenna configuration with STBC coding.

Similar to LMMSE channel estimation, calculation of $D_{coeff,k}$ involves a matrix inversion, which cannot be simplified in a similar manner. More details on reducing the complexity of symbol detection can be found in Chapter 3.

Soft Bit Detection The detected symbols go through tone de-mapping and arrive at soft bit detection block. In this step demodulation is carried out, and symbols are converted to bits. First, the corresponding constellation point is found for each symbol. Then, for each bit position, the difference between the distances to the nearest zero and one bit for the constellation point is calculated. This process is illustrated in Fig. 2.7. This method helps to reduce the complexity of soft bit detection by only calculating the distance to the nearest one and zero bits.

LDPC Decoding Similar to LDPC coding in the transmitter side, LDPC decoding is assumed to be carried out in a coarse-grain accelerator in the design. This accelerator can function in

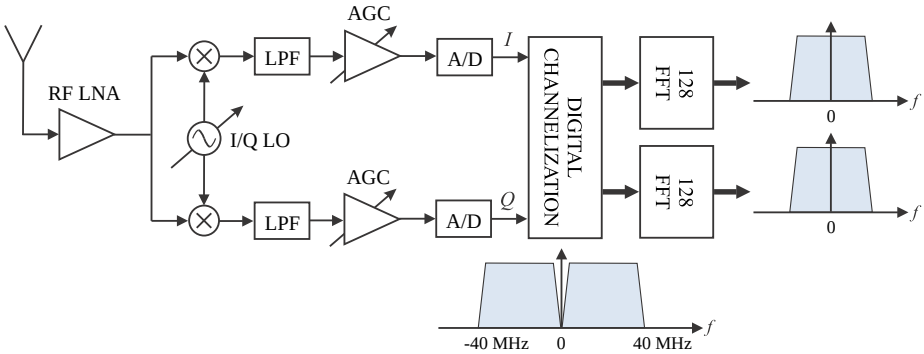


Figure 2.8: The overall receiver principle with digital channelization filtering yielding two 40 MHz sub-signals.

parallel with the rest of the frequency domain processing. The existing literature provides many implementations of LDPC architecture [20–22].

2.1.5 IEEE 802.11ac Digital Front-End Processing

This section describes a digital front-end concept developed for the IEEE 802.11ac receiver. The IEEE 802.11ac allows usage of both 80 MHz or 160 MHz bandwidths. The 80 MHz waveform is primarily comprised of two 40 MHz sub-signals, with three null subcarriers in between. Thus, with precise time-domain filtering, the 80 MHz band can be divided into two 40 MHz signals.

Then, having divided the waveform, the rest of the processing can be done with less complexity and in parallel for the two sub-bands. Furthermore, existing hardware for the IEEE 802.11n, which primarily used a 40 MHz band, can be employed. Including the chain from RF to baseband, the receiver principle incorporating this channelization concept is depicted in Fig 2.8. This concept can also be extended to the 160 MHz channel setup, introduced in the IEEE 802.11ac.

In this work, we focus on the 80 MHz bandwidth with 256 subcarriers, of which 234 are data and eight are pilot subcarriers. Thus, the positive and negative frequency components will each have 121 active subcarriers, resulting in $k = \pm[2, 3, \ldots, 122]$, and the three subcarriers around DC, $k = [-1, 0, 1]$ are zero.

The symbol duration in IEEE 802.11ac is defined as 4 μs, from which 800 ns is reserved for the GI. Having FFT size of 256, this translates to a cyclic prefix (CP) of 64 samples. Linear filtering is used to divide the signal into two 40 MHz waveforms, so that they can be further processed in parallel using two 128 point FFTs.

This channelization concept can be realized using a finite impulse response (FIR) filter. Furthermore, to reduce the computational complexity of the design, and minimize the required number of multiplications, halfband filters are selected.

2.1.5.1 Polyphase Halfband Filters

Halfband filters are commonly used in many digital communication systems, due to their efficiency in multi-rate applications. The transfer function for a halfband FIR filter is of the form [23]:

$$H(z) = \sum_{n=0}^{2M} h[n]z^{-n}, \qquad h[2M-n] = h[n]. \tag{2.24}$$

In these filters,

$$h[M] = 1/2, \qquad h[M+2r] = 0, for\ r = \pm 1, \pm 2, \ldots, \pm(M-1)/2, \tag{2.25}$$

where M is an odd integer.

A highpass/lowpass filter pair satisfying these conditions can be realized using a type II (M is odd) FIR transfer function $G(z^2)$ with a delay of M as [23]:

$$H(z) = G(z^2) \pm \frac{1}{2}z^{-M}, \tag{2.26}$$

where the order of the overall transfer function is $2M$ and the lowpass/highpass filter is realized by the plus/minus sign.

The magnitude response of the halfband and analytical filters are shown in Fig 2.9a and Fig 2.9b, respectively.

To decimate the signal by two, half of input samples go through $G(z)$ and half through $\frac{1}{2}z^{-M}$ filter. Thus, each filter branch works at the same rate as the output, which is half of the input sample rate. This structure is illustrated in Fig 2.10.

2.1.5.2 Cyclic Polyphase Halfband Filters

The linear halfband filter designed for channelization, described in the previous section, increases the time dispersion of the received signal, which invades the CP. To avoid the increase in length of the impulse response, cyclic convolution can be adopted instead of linear convolution. This is an effective solution, since cyclic convolution is carried out after CP removal, and thus CP would not be compromised. Cyclic filtering is performed block-wise in a way that the last $2M$ produced samples are added to the beginning of the block. An illustration of cyclic convolution using linear halfband filtering can be found in Fig. 2.11 [24].

2.2 Future Wireless Systems and Full-Duplex Communication

One common target being pursued in the new generations of wireless networks is providing very high capacity to meet the expected dramatic growth in mobile traffic. One example is the upcoming generation of the 3GPP cellular networks, i.e. fifth generation (5G), which is currently being standardized and developed, and aims for a 1000-fold increase in capacity compared to its preceding generations [25, 26].

Such big growth in capacity can be achieved by combining different mechanisms, namely increasing spectral efficiency, base-station densification, and using more spectrum (available in higher bands). In this section, IBFD communications is introduced, which enables more efficient use of the available spectrum. Full-duplex communications exploit the less efficient use of spectral and temporal resources in current available communication systems. For clarity, we state that 5G new radio (NR) still builds on FDD and TDD duplexing principles, while the IBFD technology may be adopted in future releases or in sixth generation (6G) era.

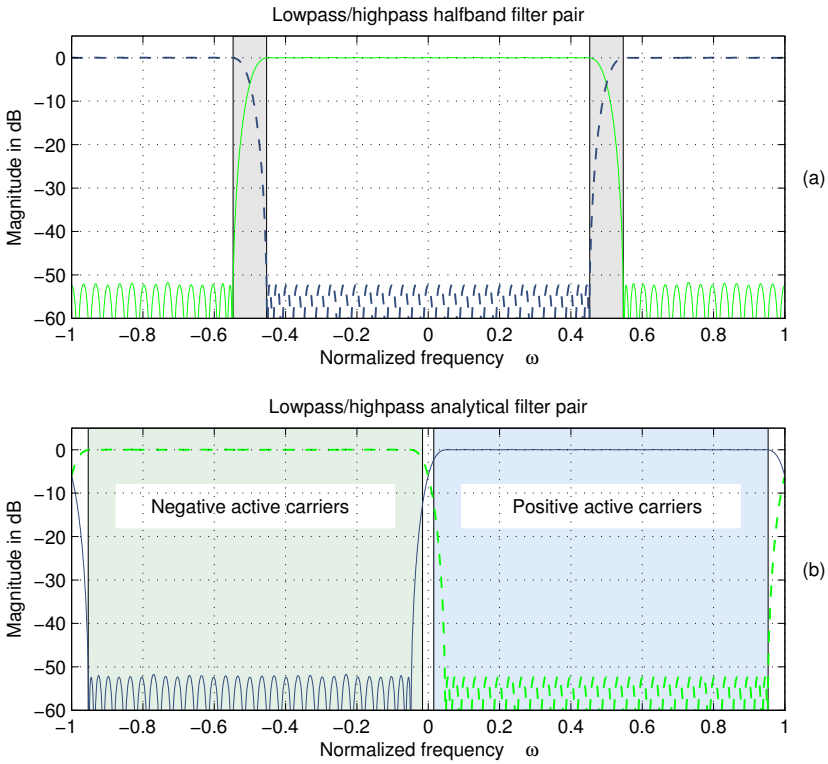


Figure 2.9: Magnitude response of (a) halfband and (b) analytical filters, grey areas in (a) indicate the transition bands, and in (b) the active subcarriers.

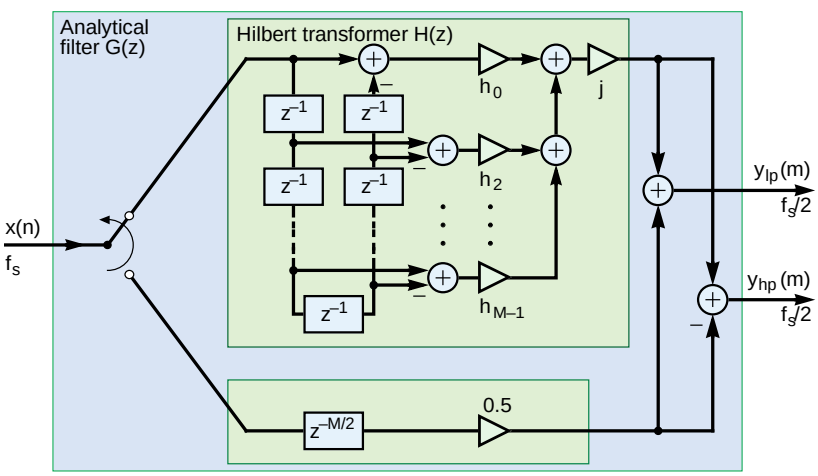


Figure 2.10: Structure of decimating analytical filter producing both the lowpass and highpass outputs $y_{lp}(m)$ and $y_{hp}(m)$, respectively

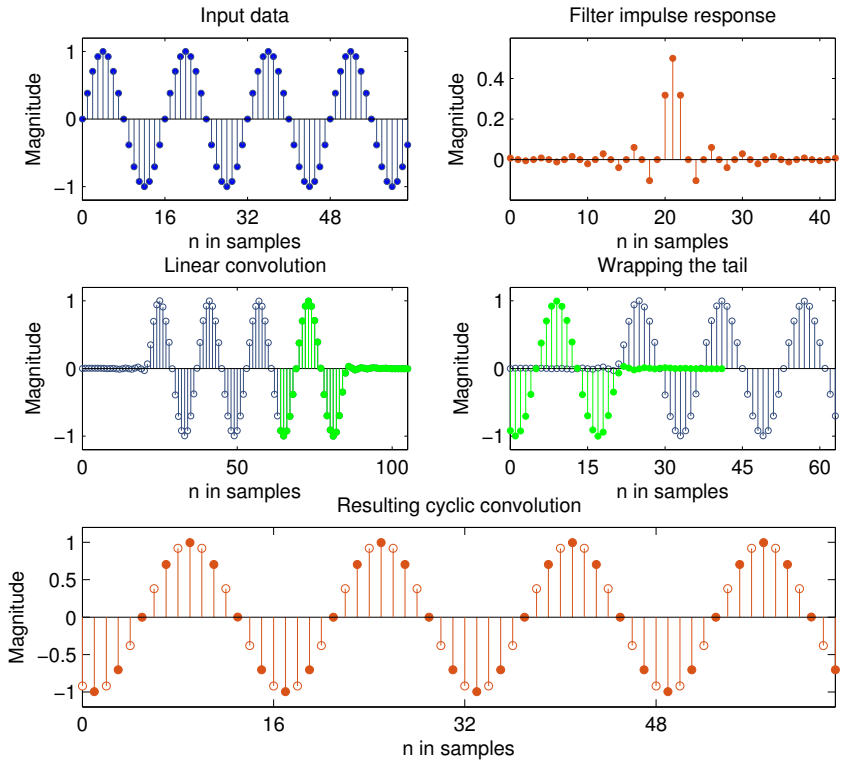


Figure 2.11: Cyclic convolution using linear halfband filtering

In the existing wireless networks, including 5G NR Release 15, the spectral and temporal resources are typically divided between downlink and uplink in two ways: FDD, and TDD. In FDD, which is used in most commercial cellular systems, transmission and reception are separated in frequency. TDD, on the other hand, divides downlink and uplink transmission in time [27].

In this section, we briefly introduce full-duplex radios and the challenges affiliated with realizing such systems. Then, a solution is described which can help to combat these challenges.

2.2.1 Full-Duplex Communication

All two-way wireless devices and systems existing today, such as global system for mobile communications (GSM), universal mobile telecommunications system (UMTS)/high speed packet access (HSPA), and WLAN/WiFi, are based on separating the transmission and reception either in time or frequency, i.e., systems operate in either TDD or FDD mode. This has the inevitable downside, however, that the spectral efficiency of using the radio frequencies is only half of its potential in theory. In contrast, full-duplex transmission is based on the challenging idea of simultaneously transmitting and receiving on a single frequency, and hence, in theory, doubling the efficiency of radio spectrum use compared to any existing system. However, as the strong transmit signal now directly couples to the sensitive receiver circuitry, substantial SI is generated, which needs to be tackled in the transceiver [28].

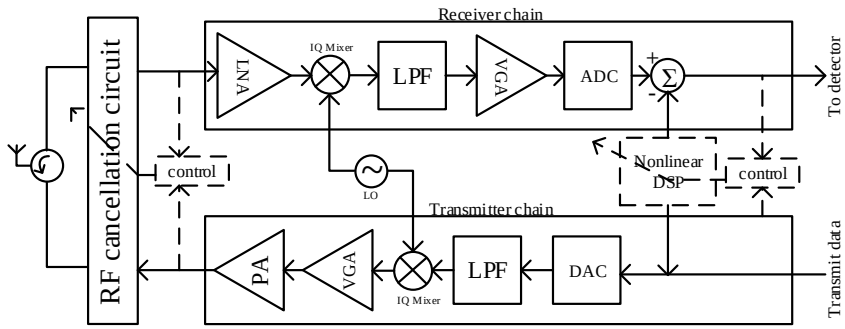


Figure 2.12: Principal illustration of full-duplex radio transceiver.

2.2.1.1 Digital Self-Interference Cancellation

Both RF and digital domain cancellations are required to reduce the SI signal to a level not interfering with the detection of the desired signal. SI cancellation in RF domain prevents the saturation of analogue-to-digital converter and receiver low-noise amplifier (LNA). However, the SI signal needs to be further suppressed in the digital domain to improve system performance.

The overall structure of a full-duplex transceiver, illustrating both the RF and digital cancellers is shown in Fig. 2.12. In this section, we discuss digital SI cancellation by first introducing a model for the SI signal.

2.2.1.2 Self-Interference Modelling

Numerous non-ideal components exist in the transmitter and receiver paths which cause both linear and non-linear distortion in the transmitted signal coupling at the receiver. These distortions originate from different sources such as, power amplifier non-linearities, local oscillator phase noise, transmitter and receiver in-phase/quadrature (I/Q) imbalance, and analogue-to-digital converter quantization noise.

In this work, the parallel Hammerstein model, commonly used for highly nonlinear power amplifiers (PAs), is adopted for modelling the signal. This is due to the fact that the PA is known as the most significant contributor to the non-linear distortion of the signal.

Denoting the PA input by $x_{PA,in}$, and using the aforementioned parallel Hammerstein model, the signal at the PA output can be written as [29]:

$$x_{PA,out} = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{k=0}^{K-1} h_p^{PA}(k) u_p(x_{PA,in}(n-k)), \tag{2.27}$$

where P is the highest non-linearity order of the PA model, K is the memory length of the PA, h_p^{PA} represents the pth-order model for the PA memory, and $u_p(x_{PA,in}(n))$ produces the pth-order basis function, and is computed using $|x_{PA,in}(n)|^{p-1} x_{PA,in}(n)$.

With the assumption of the PA as the most prominent source of non-linearity, the effective SI channel can be modelled using (2.27). Hence, denoting the original transmitted signal with $x(n)$, the received signal at the digital canceller input can be expressed as:

$$r_x(n) = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} h_p(l) u_p(x(n-l)) + z(n), \tag{2.28}$$

where L denotes length of the modelled SI channel memory, $h_p(l)$ contains the pth-order SI channel coefficients, and $z(n)$ represents noise and potential modelling mismatch. Having an estimation of the unknown SI channel coefficients, the signal after the digital canceller can be written as:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_p(l) u_p(x(n-l)), \tag{2.29}$$

where the estimated SI channel coefficients are denoted by $\hat{h}_p(l)$. According to (2.28) and (2.29), a precise estimation of the SI channel coefficients will result in only noise remaining after cancellation, which consequently means that $e(n) \approx z(n)$.

It is important to update the SI channel estimates frequently, as the environment surrounding a moving device varies in time. However, the estimation method should be also of low computational complexity to be compatible with the limited processing power available in mobile devices. Thus, a least mean squares (LMS) based solution, proposed in [29], is selected to meet the aforementioned requirements.

Furthermore, a novel basis function orthogonalization method, also proposed in [29], is adopted to help further improve the SI suppression. The following sections describe the two methods in more details.

2.2.1.3 Orthogonalization

Being produced from the same transmitted signal, the generated basis functions are expected to be correlated. As a result, the SI channel coefficient estimation process, using the LMS algorithm, would suffer from slow convergence and excess mean squared error. Thus, it is beneficial to orthogonalize the basis functions for more efficient LMS parameter learning. This is carried out using a method proposed in [29], which will be described briefly in the following.

This method uses a whitening transformation matrix for basis functions orthogonalization. The transformation matrix can be obtained from the eigen-decomposition of covariance matrix $\mathbf{\Sigma}$. Let us define the instantaneous basis function vector as:

$$\mathbf{u}(n) = \begin{bmatrix} u_1(x(n)) & u_3(x(n)) & \dots & u_p(x(n)) \end{bmatrix}^T, \tag{2.30}$$

where $u_p(x(n)) = |x(n)|^{p-1}x(n)$ is the pth-order non-linear basis function.

Now, the covariance matrix of the basis functions across different non-linearity orders can be defined as:

$$\mathbf{\Sigma} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}(n)^H]. \tag{2.31}$$

The eigen-decomposition of covariance matrix $\mathbf{\Sigma}$ can be written as:

$$\mathbf{\Sigma} = \mathbf{V}\mathbf{D}\mathbf{V}^H, \tag{2.32}$$

where eigenvalues of Σ comprise the diagonal matrix $\mathbf{D}$, and the corresponding eigenvectors build matrix $\mathbf{V}$. Having (2.32), the whitening transformation matrix $\mathbf{T}$ is defined as:

$$\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V}^H. \tag{2.33}$$

Here, $\mathbf{D}^{-\frac{1}{2}}$ denotes element-wise square root and inversion of the diagonal elements of matrix $\mathbf{D}$. Now, the basis functions can be orthogonalized using transformation matrix $\mathbf{T}$ by:

$$\tilde{\mathbf{u}}(n) = \mathbf{T}\mathbf{u}(n). \tag{2.34}$$

Having the orthogonalized basis functions from (2.34), (2.29) can be re-written as:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_{p,ort}(l)\tilde{u}_p(x(n-l)), \tag{2.35}$$

where the orthogonalized pth-order basis functions are represented by $\tilde{u}_p(x(n))$, and the SI channel estimates are denoted by $\hat{h}_{p,ort}(l)$. Now, (2.35) can be expressed using vector notations as:

$$e(n) = r_x(n) - \mathbf{h}^H \mathbf{u}_{ort}(n), \tag{2.36}$$

where

$$\begin{aligned} \mathbf{h} = \Big[& \hat{h}_{1,ort}(0), \hat{h}_{3,ort}(0), \ldots, \hat{h}_{P,ort}(0), \ldots \\ & \hat{h}_{1,ort}(L-1), \hat{h}_{3,ort}(L-1), \ldots, \hat{h}_{P,ort}(L-1) \Big]^T, \end{aligned} \tag{2.37}$$

and

$$\mathbf{u}_{ort}(n) = \Big[\tilde{\mathbf{u}}(n)^T, \tilde{\mathbf{u}}(n-1)^T, \ldots, \tilde{\mathbf{u}}(n-L+1)^T \Big]^T. \tag{2.38}$$

It should be noted that the covariance matrix $\mathbf{\Sigma}$ is only dependent on the statistical properties of the transmitted signal, and as a result, is time-invariant. Hence, we can assume that matrix $\mathbf{T}$ is pre-computed when used in the processing.

2.2.1.4 LMS Parameter Learning

This section describes the LMS-based method used to adaptively estimate the effective SI channel coefficients in a time-varying channel. The orthogonalized basis functions calculated in (2.34) are used to prevent high excess mean-squared error and slow convergence of the algorithm. Furthermore, different step sizes are used for different non-linear terms. The memory model of the channel includes both pre-cursor and post-cursor taps for more precision.

In this work, the proposed algorithm in [29] is adopted and modified to be more computationally friendly for our implementation purposes. Thus, as described in Algorithm 1, the SI channel estimates are not updated with every sample but only when a pre-defined number of samples are processed. The impact of this adjustment on the system performance is investigated and the results are presented in the following Chapters.

Algorithm 1 LMS-based adaptive nonlinear digital cancellation.

```
Initialize:
    h ← [0 ... 0]
    n ← L_post
while transmitting do
    u_ort(n) = [ũ(n + L_pre)^T ... ũ(n − L_post)^T]^T
    e(n) = r_x(n) − h(n)^H u_ort(n)
    if (n mod N == 0) then
        h(n + 1) ← h(n) + μ e*(n) u_ort(n)
    end if
    n ← n + 1
end while
```

Algorithm 1 presents the adopted LMS-based approach, where $\tilde{\mathbf{u}}$ is a vector of the orthogonalized basis functions, $\mathbf{h}$ contains the SI channel coefficient estimates, $r_x(n)$ denotes the received signal, $e(n)$ is the cancelled signal, and L_{pre} and L_{post} are the number of pre-cursor and post-cursor taps, respectively. Furthermore, $\boldsymbol{\mu}$ contains the step sizes for different non-linearity orders, and N defines how often the estimated $\mathbf{h}$ is updated.

CHAPTER 3

SDR SOLUTIONS FOR WIFI

In this chapter, the proposed SDR solutions for both the baseband and DFE processing of the IEEE 802.11ac are described. For each solution, first the employed processing platform is introduced. Then, the implementation is presented and evaluated in terms of execution time, number of clock cycles, power, and energy consumption. The achieved results are then analyzed to investigate the feasibility of a real-time software-based solution for the processing. The contents of this chapter are based on publications [P1]–[P4].

3.1 Related Work

Many works related to the implementation of the WLAN standard family have been reported in the literature. However, only a limited number of implementations with a software-based approach exists, particularly when it comes to the physical layer. As an example, [30] reports an ASIC implementation of an IEEE 802.11a transceiver PHY, with OFDM and up to 64-QAM. Similarly, in [31], an ASIC implementation of the HT IEEE 802.11n transceiver using a 40 MHz bandwidth, with two transmit and three receive antennas is described. An IEEE 802.11ac implementation can be found in [32], where the transceiver is tailored to operate with 80 MHz bandwidth and 4×4 MIMO.

The above-mentioned studies target to a fixed scenario, and thus lack the flexibility to operate in different modes. However, programmable approaches are gaining more interest due to the advances in processor technologies. The work carried out in [33] only focuses on the FFT/IFFT processing for the VHT amendment, where larger FFT sizes are required due to the wider bandwidths supported. The authors of [33] propose a software defined FFT/IFFT architecture that meets the point size, throughput and multiple data streams requirements of the IEEE 802.11ac. The implementation uses a customized soft stream processor on FPGA, and is then compared to a dedicated Xilinx FFT core. The comparison shows better resource efficiency using the flexible software defined architecture. This work, however, does not provide a solution for the rest of the transceiver processing.

Some research works found in the literature use application specific instruction-set processors (ASIPs) as a solution for a flexible, yet high performance transceiver design. With an ASIP core, the instruction-set can be optimized for specific tasks, which results in better performance for some

applications, but less flexibility in other areas. As an example, [34] presents an 802.11ac/ax design using an ASIP processor. The design includes channel singular value decomposition (SVD), channel compression/decompression, and beamforming weight computations to support the MU-MIMO features of the IEEE 802.11ac. The article reports the synthesis results, which show that while the ASIP core requires less lookup tables (LUTs) and DSP resources, it uses more registers and memories compared to the implementation with a dual ARM processor system and programmable logic. Furthermore, the measured latency was shown to fit the timing requirements of the standard. Another baseband ASIP design for SDR is presented in [35], where algorithms from 3G, 4G, and WiFi are analyzed and selected for implementation. The article reports execution costs for different algorithms and introduces heterogeneous ASIPs for different processing tasks.

Both [36] and [37] address SDR based baseband processing of the IEEE 802.11ac. The work presented in [36] covers most of the baseband functionality assuming 4×4 MIMO, 64-QAM, and 80MHz bandwidth. The radio processor used in this work, named RP-32 [38], has 256-bit data buses, 32-way single instruction multiple data (SIMD) operations, and 512-bit vector processing. The assumed clock frequency for this DSP core is 1 GHz. Similarly, [37] focuses on the inner part of the receiver (for synchronization and data detection) for IEEE 802.11ac with up to 80 MHz bandwidth, 4×4 MIMO, and 64-QAM, as well as long-term evolution (LTE) Cat-4/5/7 user equipment (UE). This work is based on an instance of the custom baseband processor template ADRES [39]. This instance, called BOARDES, has four vector processing units supporting 256-bit SIMD. The assumed clock frequency for achieving real-time processing is 800 MHz in this work.

On the DFE processing side, [40] reports a FPGA implementation of a DFE block for multi-carrier multi-antenna systems. The design includes decimation/interpolation filters implemented as polyphase filters, as well as a frame synchronization block and an automatic gain controller. The article reports the area of the implemented design, but lacks information about the timing and power consumption.

Another FPGA-based implementation for polyphase FIR filters can be found in [41], where emphasis has been put on an efficient pipelined implementation in VHSIC hardware description language (VHDL). The results in terms of area on FPGA are reported, and the design is shown suitable for synthesis on low-cost SDR hardware.

The work presented in [42] describes a similar concept to the one reported in this thesis, implementing polyphase filters using general-purpose GPUs. In this work, two GPU-based systems are employed, and CUDA is used as the software programming language [43]. The implemented polyphase filter bank channelizers on the two GPUs are compared to a CPU-based implementation, and results show that the parallelization on the GPUs can provide a speedup up to 9-16 times.

In the work presented in [44], a configurable architecture on transmitter side which is optimized for maximal hardware sharing between different modes is presented. This work, however, lacks actual hardware or software implementation. Another DFE design is discussed in [45], which uses Xilinx Virtex-5 XC5VSX50T FPGA for the implementation, and the maximum supported bandwidth is 2.9 MHz. While [45] is targeted solely for wideband code division multiple access (WCDMA) systems, [46] investigates the challenges of a multi-mode receiver DFE design suitable for cellular wireless standards from GSM to LTE, but does not cover an actual implementation.

Based on the presented state-of-the-art, a trend towards more flexible and software-based baseband and DFE implementations can be observed. However, the work of this thesis takes this trend further by covering more computationally aggressive physical layer processing scenarios, and including majority of the baseband processing, as well as an intensive DFE channelization concept. Furthermore, by adopting completely programmable processing units and COTS platforms, as

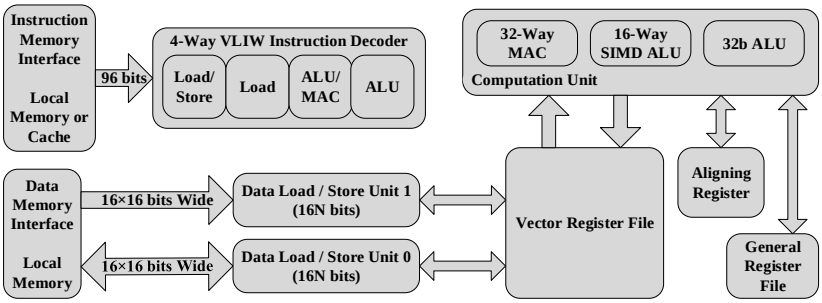


Figure 3.1: Connx BBE32 principal block diagram.

opposed to FPGAs, the feasibility of using SDRs for real time processing while maintaining the power consumption at reasonable levels is shown, which is currently rarely covered in the literature.

3.2 Baseband Processing

Targeting to very high throughputs, the IEEE 802.11ac imposes very challenging requirements on the processing platform. With the extensive amount of data to be processed, data level parallelism could be exploited using a SIMD processor architecture. Furthermore, employing a VLIW core helps to also take advantage of the instruction level parallelism.

For the above-mentioned reasons, we have chosen the Cadence Tensilica ConnX BBE32 [47] core for this work, which is a VLIW processor with vector processing capabilities. The BBE32 is a small, high performance, low power DSP core, which makes it specifically suitable for UE side processing [47].

The principal block diagram of the ConnX BBE32 can be found in Fig. 3.1. As shown in the figure, a 16-way SIMD arithmetic logic unit (ALU) and a four issue VLIW processing pipeline are included in the BBE32 architecture. These features make the BBE32 a proper fit for this application. Furthermore, there are 32 multiply-accumulate (MAC) units, and data can be accessed in blocks of 256 bits.

3.2.1 Transmission Scenarios

This work considers four transmission scenarios for the IEEE 802.11ac. For all cases, a common 80 MHz channel bandwidth is assumed, which means 256 OFDM subcarriers. The 256 subcarriers comprise of 234 data, 14 null and eight pilot carriers. All cases use 256-QAM constellation mapping. Table 3.1 presents the four transmission cases and the differences between them.

Table 3.1: The implemented transmission scenarios. For all cases 80 MHz BW, 256-QAM, LDPC coding, $3/4$ coding rate and short GI are assumed.

Cases	Number of antennas	Number of spatial streams	STBC coding
Case A	2	2	NO
Case B	4	4	NO
Case C	2	1	YES
Case D	4	2	YES

From the scenarios introduced above, cases with four antenna configurations require the heaviest computations, specifically for matrix inversion. To reduce the complexity, and thus number of clock cycles required for the processing of these cases, an accelerator for matrix inversion is developed. As it will be demonstrated in Results section, without the accelerator, the MIMO cases would require very high clock frequency from the processing platform. The accelerator is described in the following section.

3.2.2 Accelerator for Matrix Inversion

Matrix inversion is the most computationally intensive process in the transceiver chain. This issue has gained a lot of interest, especially with higher orders of MIMO emerging in both WLAN and cellular networks [48, 49].

We have designed an accelerator for the BBE32 core to lower the complexity of 4×4 matrix inversion. This solution is tailored for case B, the most complex scenario from Table 3.1. The results demonstrating the speedup achieved with the help of this accelerator can be found in section 3.2.3. The solution we have proposed for accelerating matrix inversion is described in the following.

Timing Requirements for Matrix Inversion As the matrix inversion accelerator is targeted for speeding up the calculation of detector coefficients in receiver using VHT-LTF symbols, the timing constraints stem from the VHT-LTF symbol duration. According to IEEE 802.11ac amendment, the duration of one VHT-LTF symbol is 4µs [7]. Thus, assuming a 500 MHz clock frequency for the platform, the VHT-LTF processing should be carried out in 2000 clock cycles.

Upon reception of a VHT-LTF symbol, LS and LMMSE channel estimations should be carried out, in addition to the calculation of detector coefficients. The accelerator is specifically designed for case B, which has a 4×4 antenna configuration without STBC coding. As the results presented in section 3.2.3, Table 3.4 show, the two channel estimations take 1200 clock cycles to complete. This leaves 600 cycles for detection of the coefficients.

For processing the 234 data, non-pilot subcarriers, 234 complex 4×4 matrix inversions are required. As shown in Fig. 3.1, BBE32 has two 16×16 bits wide interfaces to local memories. Complex numbers are seen as two 16-bit fixed point numbers. Thus, for each 4×4 complex matrix inversion, two clock cycles are consumed: one for reading, and one for writing to memory ($2 \times 16 \times 4 \times 4 = 2 \times 256$). This means that reading and writing data for all 234 subcarriers takes $2 \times 234 = 468$ clock cycles. Thus, only $600 - 468 = 132$ clock cycles are left for the computations, with one matrix read/write in every other cycle.

Modified Gram-Schmidt Algorithm The modified Gram-Schmidt algorithm is a more stable version of the classical Gram-Schmidt process for orthogonalization [50]. We perform QR decomposition using modified Gram-Schmidt to simplify the matrix inversion process. The accelerator implements the method proposed in [51], and uses $\log 2$ and x^2 domains for computations. Thus, the more time consuming arithmetic computations, such as multiplication and division turn into simple additions and subtractions, respectively.

The domain conversions are implemented as LUTs. However, realization of $\log 2$ conversions LUTs is rather inconvenient for complex numbers. For this reason, the 4×4 complex matrix is first decomposed to an 8×8 real matrix. Having the complex channel matrix $\mathbf{H}$, the real matrix can be written as:

$$\mathbf{A} = \begin{bmatrix} real(\mathbf{H}) & -image(\mathbf{H}) \\ image(\mathbf{H}) & real(\mathbf{H}) \end{bmatrix}, \tag{3.1}$$

where $real(\mathbf{H})$ and $image(\mathbf{H})$ are the real and imaginary parts of complex channel matrix $\mathbf{H}$, respectively.

Now, inversion of matrix $\mathbf{A}$ is carried out with following three steps:

1. QR decomposition of matrix $\mathbb{A}$, such that:

$$\mathbf{A} = \mathbf{QR}, \tag{3.2}$$

   where $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{R}$ is an upper triangular matrix. For orthogonal matrix $\mathbf{Q}$, we have:

$$\mathbf{Q}^T\mathbf{Q} = 1. \tag{3.3}$$

2. Calculating $\mathbf{R}^{-1}$, having $\mathbf{Q}^T = \mathbf{Q}^{-1}$. Thus, $\mathbf{A}^{-1}$ can be calculated by:

$$\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^T. \tag{3.4}$$

3. multiplication of $\mathbf{R}^{-1}$ and $\mathbf{Q}^T$

A column-wise implementation of QR decomposition using modified Gram-Schmidt is shown in Algorithm 2, where $\mathbf{v}_i$ is a temporary vector, $\mathbf{a}_i$ is a vector containing elements from the ith column of matrix $\mathbf{A}$, r_{ji} is the element from row j and column i of matrix $\mathbf{R}$, and $\mathbf{q}_i$ is the ith column of matrix $\mathbf{Q}$. Furthermore, $\mathbf{v}.\mathbf{q}$ represents the inner product of $\mathbf{v}$ and $\mathbf{q}$, and $\|\mathbf{x}\|_2$ is the L_2 norm of $\mathbf{x}$.

Algorithm 2 QR decomposition with modified Gram-Schmidt

1: **for** $i = 1 : n$ **do**
2: $\quad\mathbf{v}_i = \mathbf{a}_i$
3: $\quad$**for** $j = 1 : i - 1$ **do**
4: $\quad\quad r_{ji} = \mathbf{v}_i.\mathbf{q}_j$
5: $\quad\quad \mathbf{v}_i = \mathbf{v}_i - r_{ji}\mathbf{q}_i$
6: $\quad$**end for**
7: $\quad\mathbf{q}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$
8: $\quad r_{ii} = \|\mathbf{v}_i\|_2$
9: **end for**

Now that we have the $\mathbf{Q}$ and $\mathbf{R}$ matrices, $\mathbf{R}^{-1}$ can be calculated using Algorithm 3, where r_{ji}^{inv} represents the element in row j and column i of matrix $\mathbf{R}^{-1}$.

Fig 3.2 depicts how the dot product is calculated in step four of Algorithm 2, using LUTs and simple additions and subtractions. In this figure, inputs and outputs are in linear domain.

Fig. 3.3 illustrates calculation of $\mathbf{v}_i = \mathbf{v}_i - r_{ji}\mathbf{q}_i$ in step five of algorithm 2, using LUTs. In this figure, all inputs are in linear domain from which r_{ji} and $\mathbf{q}_i$ have to be first converted to log_2 domain.

Using designs similar to the ones shown in Fig. 3.2 and 3.3, the matrix inversion can be completely carried out using solely additions and subtractions. As a result, the generally very complex matrix inversion process can be speeded up to a great extent using the accelerator for BBE32. Detailed results on the achieved performance enhancement can be found in section 3.2.3.

Algorithm 3 Calculation of inverse matrix for upper triangular matrix $\mathbf{R}^{-1}$

1: **for** $i = 1 : n$ **do**
2: **for** $j = 1 : i - 1$ **do**
3: $r_{ji}^{inv} = r^{inv}(j, (1 : i - 1)) \times r((1 : i - 1), j)$
4: **end for**
5: $r_{1:i-1,i}^{inv} = \frac{-r_{(1:i-1),i}^{inv}}{r_{ji}}$
6: $r_{ii}^{inv} = \frac{1}{r_{ii}}$
7: **end for**

Implementation Fig. 3.4 shows the pipeline schedule for the matrix inversion accelerator for a 4×4 complex matrix. If we assume a register after each LUT for pipelining, the pipeline will have an overall 64 clock cycles delay plus a few cycles to buffer the data coming and going to BBE32.

The logic elements required for the overall implementation include: 1622 adders (mostly 16 bit), 877 x^2 LUTs (256×15 bits), 281 log_2 LUTs (64×14 bits). Having in mind that BBE32 can only read and write half a 4×4 complex matrix in each clock cycle, half of the adders and LUTs can be reused to process one complex 4×4 matrix in every two clock cycles.

As presented in Table 3.4 of section 3.2.3, the calculation of the detector coefficients, with the help of the matrix inversion accelerator, consumes 548 clock cycles overall. This is below the 600 clock cycles budget for this operation.

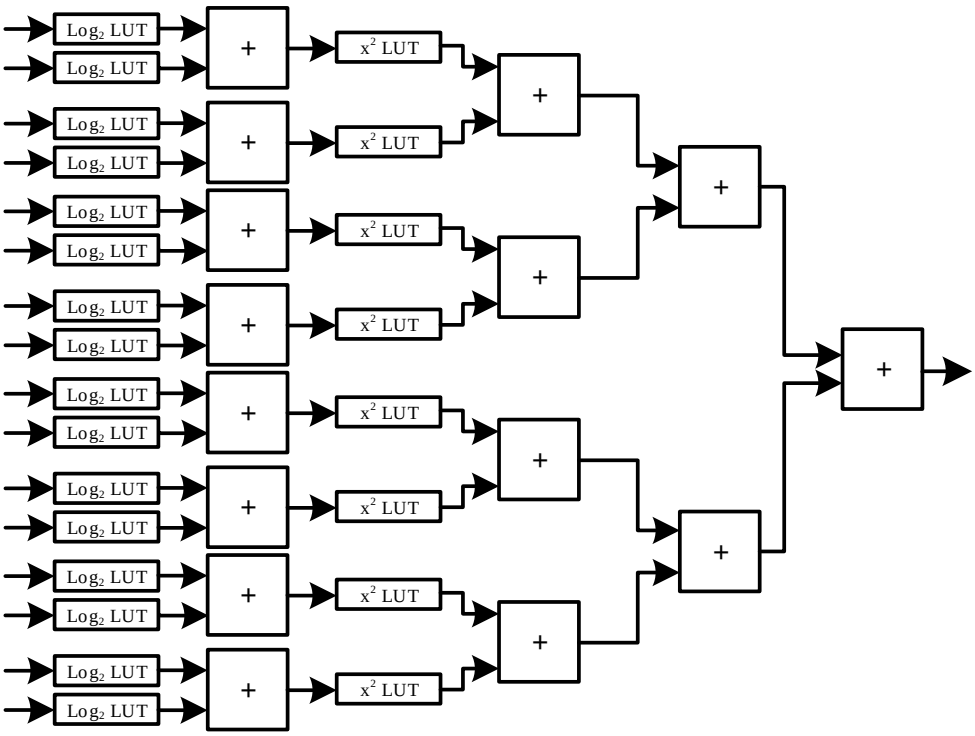


Figure 3.2: Calculating $\mathbf{v}_i.\mathbf{q}_j$ using LUTs

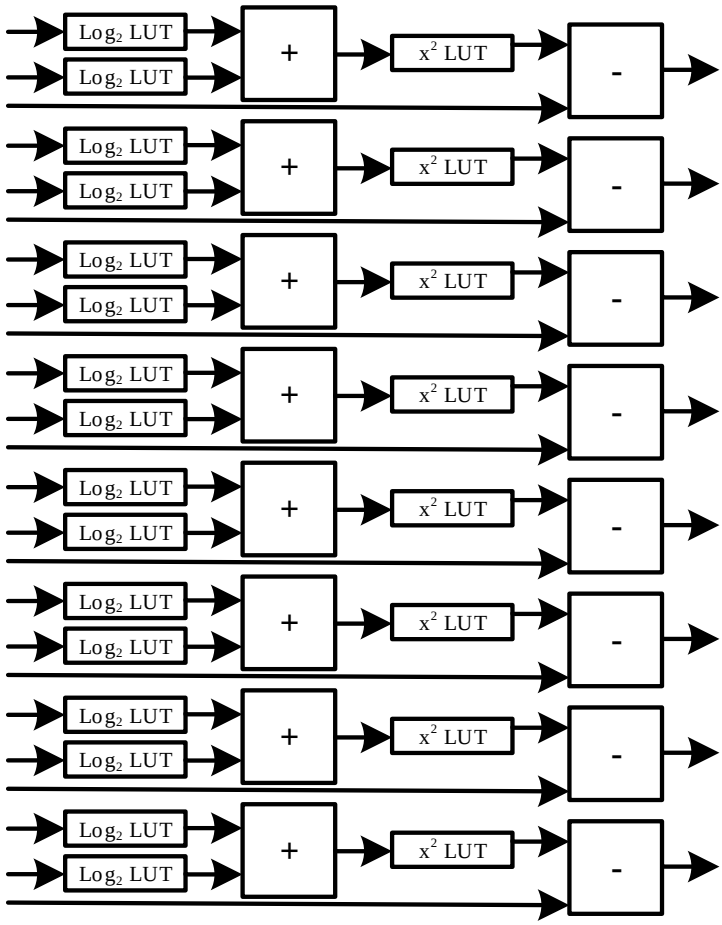


Figure 3.3: Calculating $\mathbf{v}_i = \mathbf{v}_i - r_{ji}\mathbf{q}_i$ using LUTs

Furthermore, looking at Fig. 3.4 reveals that it is possible to reuse half of resources in slots where they are inactive. Thus, a quarter of the elements mentioned earlier are sufficient for this implementation (417 adders, 220 x^2 LUTs, and 71 log_2 LUTs).

As it will be demonstrated with the results in the following section, this matrix inversion accelerator greatly helps to speedup the computations. As a result, real-time implementation of the transceiver becomes feasible having lower clock frequencies for the processing core.

3.2.3 Results

To investigate the possibility of a real-time implementation, we have measured the number of clock cycles each implemented block consumes. The results from the transmitter side processing are presented in Table 3.2.

The preparation block, mentioned in Table 3.2, does not carry out a functionality specified by the IEEE 802.1ac, but merely re-arranges the bits in the stream. This simplifies the computations performed in the upcoming blocks.

Furthermore, to facilitate the operation, the ordering of the transmitter blocks is modified. As an

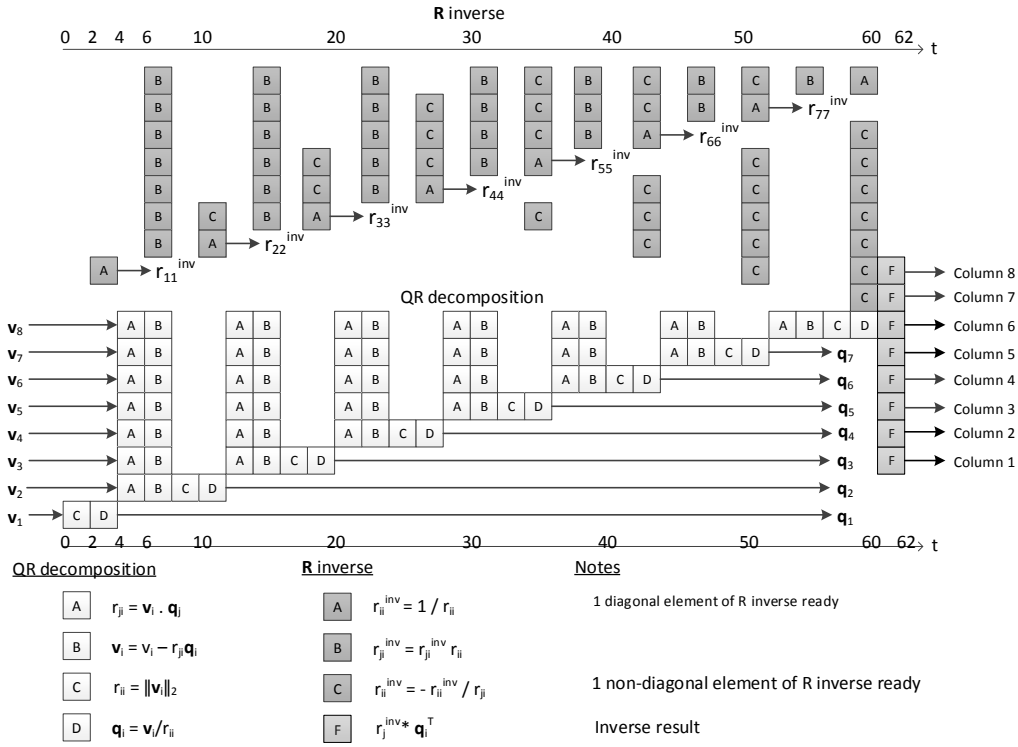


Figure 3.4: Pipeline schedule for modified Gram-Schmidt QR matrix inversion for 4×4 complex-valued matrices, where the light gray boxes show the computations from Algorithm 2, dark gray boxes show the computations from Algorithm 3, and boxes marked with "F" show the calculation of the final inverted results.

example, the STBC coding is now performed on the data on bit level, before they are mapped to complex numbers in the constellation mapping block. Additionally, the functionality of some operations are merged together in order to decrease the consumed number of clock cycles.

In Table 3.2, "Pilots" refers to the insertion of pilot subcarriers, and phase rotation is the rotation of tone, which in case of an 80 MHz bandwidth is basically a multiplication by one or minus one for some subcarriers.

Fig. 2.2 indicates that the duration of a DATA symbol with short GI is 3.6 µs. For a real-time implementation, the overall transmitter processing for one OFDM symbol should be carried out in this time. According to Table 3.2, the most complicated case takes less than 800 clock

Table 3.2: The number of clock cycles needed for the processing of a DATA symbol in the transmitter (cases described in Table 3.1):

Functional Blocks	Case A	Case B	Case C	Case D
Preparation + STBC	53	-	111	68
Tone Mapper	159	159	159	159
Stream Parser + Constellation Mapper	153	197	153	300
Pilots + CSD + Phase Rotation + Spatial Mapping	130	210	136	256
Total Number of Cycles	495	616	559	783

Table 3.3: The number of clock cycles needed for the processing of a DATA symbol in the receiver.

Functional Blocks	Case A	Case B	Case C	Case D
Pilot Removal	140	336	140	336
Symbol Detection	468	652	208	919
Tone De-mapper	324	672	324	672
Stream Deparser	74	254	-	148
Soft Bit Detection	3193	6504	3193	6504
Frequency Error Tracking	255	331	255	331
Total Number of Cycles	4639	9087	4305	9275

cycles overall. If we assume a 500 MHz clock frequency for the processing platform, it takes approximately 1.56 µs ($\frac{783}{500\ MHz} = 1.56\mu s$) for a DATA symbol to go through the implemented processing in the transmitter. Thus, a real-time processing of DATA symbols can be assumed feasible in all considered transmission scenarios in the transmitter. The same conclusion can be drawn for the preamble symbols, as they require less complex computations than the DATA symbols.

Table 3.3 presents the number of clock cycles required for the processing of a received DATA symbol. RCPI measurement is carried out using the active non-pilot subcarriers in a DATA symbols, so it is included in the "Pilot removal" block. Pilot removal separates the pilot subcarriers from the tones carrying data.

Looking at the total number of clock cycles in Table 3.3 for cases A and C, which only take advantage of a two-antenna configuration, we can see that clock frequencies just above 1 GHz are required for real-time processing. However, cases with four antennas require approximately twice more time than cases A and C, meaning that the processing platform for cases B and D require higher than 2 GHz clock frequency.

In addition to the DATA field, the VHT-LTF requires some high complexity processing on the receiver side. The number of clock cycles consumed in the different operations for the VHT-LTF symbol are represented in Table 3.4.

Table 3.4 contains two sets of numbers for case B, where "case B/ACC" is the scenario which utilizes the matrix inversion accelerator, described in section 3.2.2. The complex matrix inversion process is speeded up with the help of this accelerator and the number of clock cycles has dramatically decreased to 1944 from the original 33109 cycles.

The 4 µs, defined as the duration of the VHT-LTF, sets the constraints for the required clock frequency. Taking the matrix inversion accelerator into use, the processing platforms should have a clock frequency of 500 MHz to process the VHT-LTF symbol in real-time.

Another important criterion in studying the feasibility of this SDR implementation is the power

Table 3.4: The number of clock cycles needed for the processing of a VHT-LTF symbol in the receiver.

Functional Blocks	Case A	Case B	Case B/ACC	Case C	Case D
LS Channel Estimation	281	289	289	281	2873
LMMSE Channel Estimation	1078	1107	1107	1078	1643
Detector Coefficients	2630	31713	548	851	34005
Total Number of Cycles	3989	33109	1944	2210	38521

consumption of the design. The BBE32 comes with an energy analyzer tool, which provides estimates of the consumed energy. Dividing the energy consumption estimates by each block's execution time gives an approximation of the consumed power.

To calculate the execution times, a 500 MHz clock frequency is assumed. The measurements depend on the applied memory capacity, for which we have assumed the maximum 128kB. The energy analysis is carried out in 3.6 µs time for the DATA symbol. The power consumption of the transmitter and receiver when processing a DATA symbol can be found in Tables 3.5 and 3.6, respectively.

Comparing to some reported WiFi power consumption values in mobile devices [52], the total consumed power in the transmitter and receiver show that it is allowed to employ such software-based designs in the UE side.

3.3 Digital Front-End Processing

For implementation of this digital front-end channelization concept, three different processing platforms were adopted. These are COTS products that are currently employed in some of the available devices in the market. These platforms are briefly introduced in the following.

Odroid XU3 We use the Odroid XU3 development platform, which is based on the Samsung Exynos5422 Cortex™-A15 and Cortex™-A7 CPUs [53]. This board employs the ARM® big.LITTLE™ technology [54–56]. The idea behind this technology is to couple a relatively lower performance battery-saving CPU, i.e. A7, with a more powerful core with higher power consumption, i.e. A15. This board is also equipped with the Mali™-T628 MP6 GPU. For the channelization processing, both the A7 CPU and Mali GPU are utilized.

ARM® Mali™-T628 MP6 Mali is a mobile-scale GPU and runs at a 600 MHz clock frequency [57]. This GPU can scale from one to eight cores, each of which can handle up to eight floating point operations per cycle [58]. Furthermore, Mali supports half-precision floating-point arithmetic, defined by IEEE 754 standard [59].

ARM® Cortex®-A7™ The A7 is the so called LITTLE CPU in ARM's big.LITTLE architecture technology. Thus, A7 is slower but less power-hungry compared to A15. A7 is a multi-core processor, which has between one to four cores, and can run at up to 1.5 GHz clock frequency.

Intel® Core™ i7-4800MQ Unlike the processing platforms mentioned above, the Intel Core i7 is a desktop CPU. This processor has four cores and can run at up to 3.7 GHz [60].

Table 3.5: Power consumption in mW in the transmitter for processing of a DATA symbol.

Functional Blocks	Case A	Case B	Case C	Case D
Preparation + STBC	1,8	-	3,7	2,3
Tone Mapper	5,2	5,2	5,2	5,2
Stream Parser + Constellation Mapper	5,1	6,6	5,1	9,4
Pilots + CSD + Phase Rotation + Spatial Mapping	4,8	9,6	4,8	9,6
Total Power Consumption	16,9	21,4	18,8	26,5

Table 3.6: Power consumption in mW in the receiver for processing of a DATA symbol.

Functional Blocks	Case A	Case B	Case C	Case D
RCPI Variance	3,60	6,67	3,35	6,16
Pilot Removal	4,76	10,39	4,76	9,52
Symbol Detection	13,00	23,51	6,82	34,65
Tone De-mapper	10,02	20,07	10,02	20,03
Stream Deparser	3,06	8,59	-	6,11
Soft Bit Detection	106,95	213,92	105,43	213,19
Frequency Error Tracking	4,46	7,55	4,44	7,55
Total Power Consumption	145,61	290,70	134,82	297,21

The aim is to exploit the parallelism offered by these platforms along with the offered flexibility of the OpenCL. OpenCL is a standard for general-purpose, parallel programming across different platforms, which helps to improve the speed of a wide range of applications [61].

In this work, first, both OpenCL and C implementations are carried out on the Intel CPU. The aim of this step is to determine the amount of speedup achieved with OpenCL compared to simply using C. Then, to investigate actual mobile scale, and highly parallel processing platforms, the ARM Mali GPU and A7 CPU are employed.

3.3.1 Channelization Filtering

To achieve the best performance for the IEEE 802.11ac channelization, different approaches are considered. Each OpenCL implementation is carefully designed to most optimally take advantage of the available parallelism.

Two different designs are introduced in the following. The first solution uses a halfband filter with lower number of computations and higher order. On the other hand, the second design is based on a non-halfband filter with a shorter filter length, which utilizes vectorization. Implementations are carried out for both linear and cyclic filters.

3.3.1.1 Halfband Filters

The advantage of halfband filters is that they require less computations, as every other coefficient is zero. This decreases the number of computations by a factor of two. Additionally, having symmetric coefficients helps to reduce the complexity further. This is due to the fact that, the samples with symmetric coefficients can be first subtracted and then multiplied with the coefficient. Furthermore, the highpass and lowpass outputs of the filter can be realized at the same time.

In this implementation, it is assumed that both a block of input samples corresponding to one OFDM symbol, and the coefficients are loaded to the input buffers of the kernel. With L and N denoting the number of samples in an OFDM symbol and the filter length, respectively, the work distribution among the OpenCL work groups and elements is depicted in Fig. 3.5.

We assume that $N + L - 1$ samples are stored and fed to the kernel. As shown in Fig. 3.5, the input samples are padded with $N - 1$ zeros for filtering purposes. With the above workload distribution in the kernel, $L/2$ work groups are active simultaneously to multiply the samples and coefficients, and sum the results. Thus, all work groups produce one lowpass and one highpass output sample at the same time.

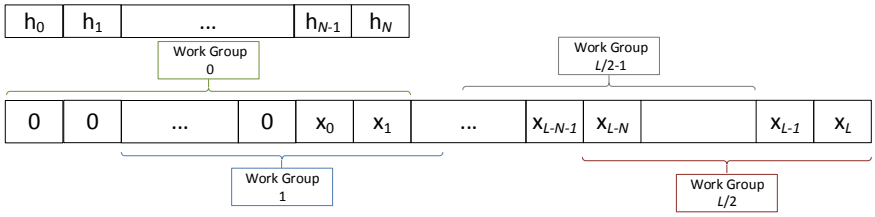


Figure 3.5: The workload distribution of implemented halfband filter in OpenCL, x denotes input samples, h is the filter coefficients, L represents the number of input samples, and N is the filter order.

3.3.1.2 Non-Halfband Filters

Although the halfband filter requires less computation, the erratic memory accesses due to the fact that only every other coefficient is used, might slow down the processing. For this reason, a non-halfband filter is also considered for channelization. In this implementation, we used the OpenCL vector operations to exploit the processing cores' support for SIMD operations. OpenCL provides support for up to 16 element vector operations. Thus, for an optimal design, filter lengths that are multiple of 16 are chosen. To avoid increasing the complexity by a factor of four using odd-order (even length) filters, we create a filter of length $16n$-1, and pad it with a zero to have a length of $16n$.

For the non-halfband filter, the computations are assigned to work elements as shown in Fig. 3.6. Here both the filter coefficients and the input samples are processed as vectors of 16 elements. The input buffer's length is S, which includes the total number of samples in one OFDM symbol, and the N padded zeros in the beginning. In this illustration, $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{S/16}$ are the vectors containing the total S input samples.

Here, each work item carries out the processing of some of the input vectors depending on the work item number. This means that each work item performs the multiplications and summation related to one output sample. Thus, one lowpass and one highpass output sample is created in each work item.

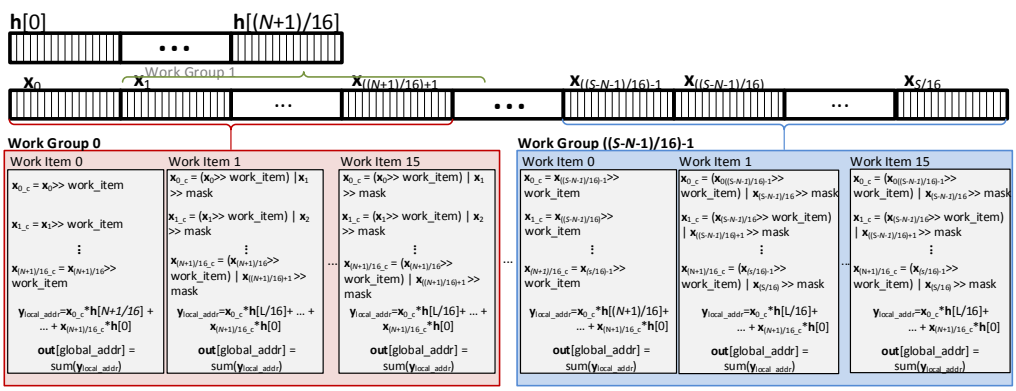


Figure 3.6: The workload distribution of implemented non-halfband filter in OpenCL, $\mathbf{x}$ denotes the input sample vectors, $\mathbf{h}$ are the vectors containing filter coefficients, N is the filter order, and S is the number of input samples plus N padded zeros.

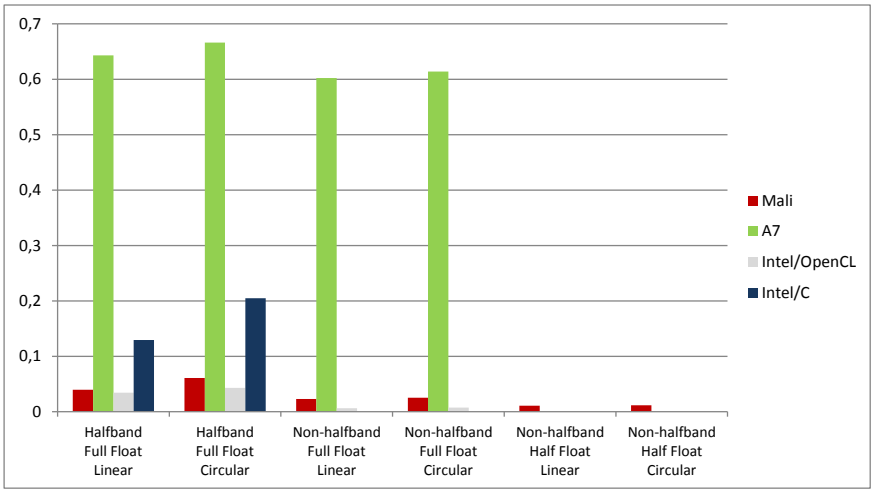


Figure 3.7: The execution times of the implemented halfband, and non-halfband, and linear, cyclic filters with half precision and full precision floating points in milliseconds on all three platforms.

3.3.2 Results

The implementations, introduced in section 3.3, are evaluated in terms of execution time, number of clock cycles, power, and energy consumption, and the results are presented in this section. Furthermore, as Mali supports half-precision floating point arithmetics, the performance results when using half and full precision are compared and analyzed. According to IEEE 754 standard [59], half-precision floating numbers are defined to have 16 bits consisting of five bits for the exponent, 10 bits for the fraction, and one bit for the sign. In all the measurements presented below, the input length is FFT length, CP length, and filter length together.

Execution Time The execution time of the linear and cyclic channelization filters with both halfand and non-halfband filters implemented on different platforms are compared in the chart in Fig. 3.7. Comparing the C and OpenCL implementations shows that managing the available parallelism using OpenCL has resulted in an approximately 80% faster execution.

Aside from the Intel desktop scale CPU, the highest performance is achieved by the Mali GPU. The Intel CPU outperforms all the other platforms due to its much higher clock frequency, i.e. up to 3.7 GHz. Although Mali has a slower clock than the A7, it carries out the channelization processing faster because of the higher number of available parallel processing elements (PEs).

Comparing the results from the single and half precision floating points shows that the execution time is approximately decreased by 55%, which surpasses the expected 50%. The reason behind this could be the lower amount of memory occupied by data, which means more cache hits, and as a result, faster execution.

The designed non-halfband circular and linear filters are of the same length. Thus, as expected, there is little difference in execution times of the two implementations. However, as the circular halfband filter has a longer length than the linear halfband filter, the execution is somewhat slower, as it can be seen from Fig. 3.7.

These implementations should fit in the timing constraints of the IEEE 802.11ac standard to be suitable for real-time applications. The requirements stem from the short inter-frame space length defined in the IEEE 802.11ac amendment [7]. The short inter-frame space time in the 5 GHz

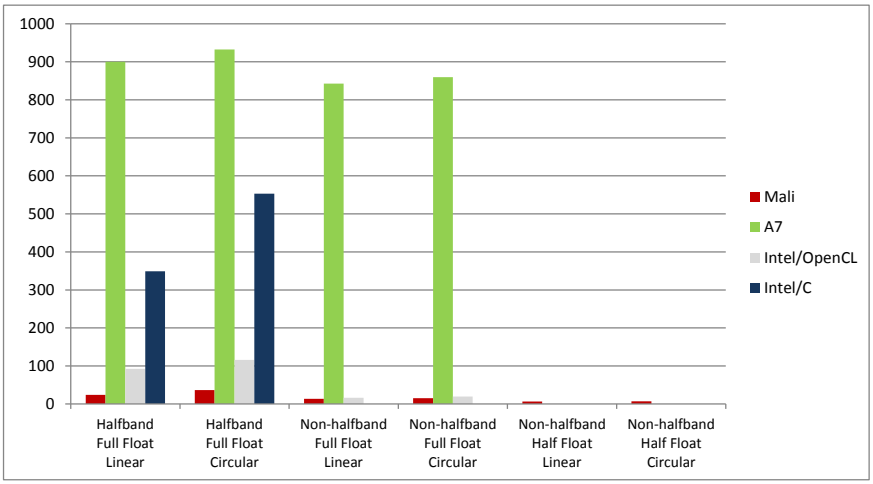


Figure 3.8: The number of clock cycles of the implemented halfband, and non-halfband, and linear, cyclic filters with half precision and full precision floating points in milliseconds on all three platforms.

band, in which the 80 and 160 MHz BWs are used, is 16 µs [7]. This time should also include the other related processing, such as MAC processing. The lowest achieved execution time on the employed platforms is 6.02 µs, which fits in the 16 µs short inter-frame space timing. To have more relaxed timing requirement for the rest of the processing, the channelization should be accelerated. This can be achieved using bigger or higher frequency GPUs, which can also be applied in an access point (AP) setup.

Number of Clock Cycles Using the measured execution times and the clock frequency of the platforms, the number of clock cycles required for the different channelization filters are calculated. Fig. 3.8 presents the number of clock cycles for different cases on all platforms.

Power Consumption The Odroid board is equipped with four sensors which measure the current going through the dynamic random-access memory (DRAM), Mali GPU, the A7, and 15 CPU. With the help of these sensors, we have measured the power consumption of our designs. The sensors are read in intervals of 100 ms, and 200 samples are taken each time. These measurements are averaged over 20 s time period.

As the kernels consume very little time, we run them in high number of iterations with the aim of keeping the cores active during the whole 20 s measurement time. However, any program running in the background, such as the operating system could partly account for the CPU/GPU power consumption. Thus, the processors' idle power, i.e., power consumption while not running any kernels, are computed and subtracted from the measured results. Only the results for Mali and A7 are presented in Fig. 3.9, as no power measurement tools were available for the Core i7 CPU. However, as reported by Intel, the thermal design power, which represents the average power, in watts, dissipated by the processor when operating at base frequency with all cores active under an Intel-defined, high-complexity workload, is approximately 47 watts [62]. This approximation shows the higher power consumption of the Core i7, compared to the reported results from Mali and A7.

The low performance, and low power A7 CPU consumes less power than the Mali GPU. However Mali's support for the half precision floating point has resulted in 33% lower power consumption,

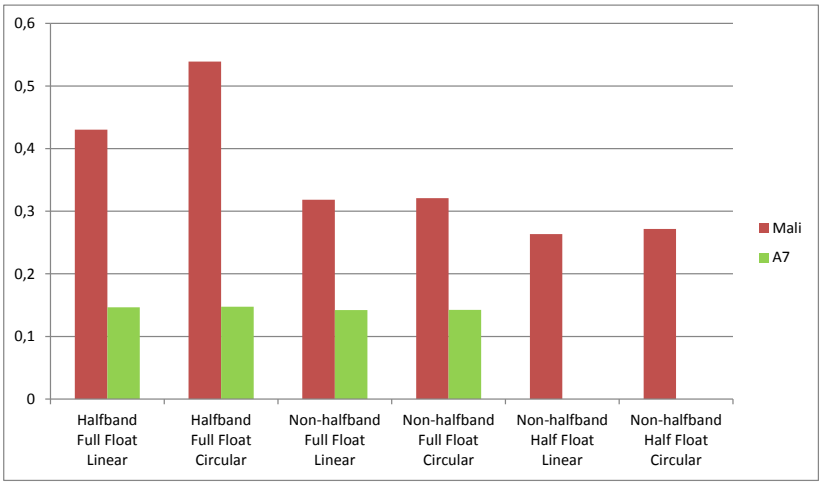


Figure 3.9: The power consumption of the implemented halfband, and non-halfband, linear, and cyclic filters with half precision and full precision floating points in watts.

which is still somewhat higher than the A7 with full precision floating points.

Energy Consumption In addition to the power consumption, it is important to evaluate the consumed energy. The energy consumption directly influences the battery life of the device. Furthermore, energy consumption comparison leads to fairer analysis compared to power, as we normalize the execution time. Thus, using the kernel execution times and power consumption, we have calculated the energy consumption of the channelization filter implemented on Mali and A7. The results are presented in Fig. 3.10

With almost twice smaller execution time, and twice less power consumption, application of half precision floating point numbers has resulted in 60 % less energy consumption compared

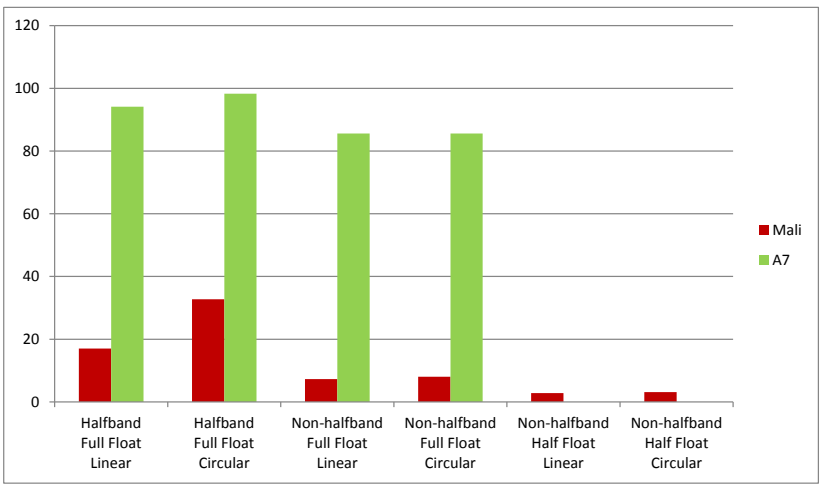


Figure 3.10: The energy consumption of the implemented halfband, and non-halfband, linear, and cyclic filters with half precision and full precision floating points in µJ.

to single precision floating points. Furthermore, the graphs in Fig. 3.10 show that even though A7 consumes less power, the energy consumption is lower in Mali. This is due to the fact that execution of the kernels takes much less time in Mali.

CHAPTER 4

SDR SOLUTIONS FOR FULL-DUPLEX COMMUNICATIONS

Similar to the DFE processing, the digital SI canceller implementations, presented in this thesis, employs COTS processing platforms, which are used in devices available in the market. Using the Odroid XU3 development board, introduced in the previous section, we take advantage of the ARM® Mali™-T628 MP6 GPU and the Cortex™-A15 CPU. Additionally, in this work, the Adreno 430, which is a powerful mobile-scale GPU is utilized as an alternative processing platform. The contents of this chapter are based on publications [P5] and [P6].

An introduction of the reference processing platform, the Intel Core-i7, as well as the Mali GPU is provided in section 3.3. Thus, only the Adreno GPU, and the A15 CPU are presented in the following.

Qualcomm® Adreno™ 430 Adreno 430 is a GPU by Qualcomm, available in the Snapdragon 810 system on chip (SoC), which is designed for mobile-scale devices. This GPU can run at 500 MHz, 600 MHz, or 650 MHz clock frequency [63]. Details about Adreno's architecture are scarcely available for public use, however it seems that it can roughly support 200 floating point operations per cycle. We used a commercial android mobile phone to run the OpenCL implementation of the digital SI canceller.

ARM® Cortex®-A15™ The A15 CPU is a part of the ARM big.LITTLE technology introduced in the previous section, where A15 is the big and A7 is the LITTLE CPU. This technology is used in the Samsung Exynos 5422 SoC found on the Odroid development board. Although compared to the LITTLE processor, A15 is more power-hungry, it is still considered a low-power processor. The high performance A15 processor has one to four cores, each equipped with NEON advanced SIMD instruction set and vector floating point units. A15 can run at up to 2.1 GHz clock frequency [55].

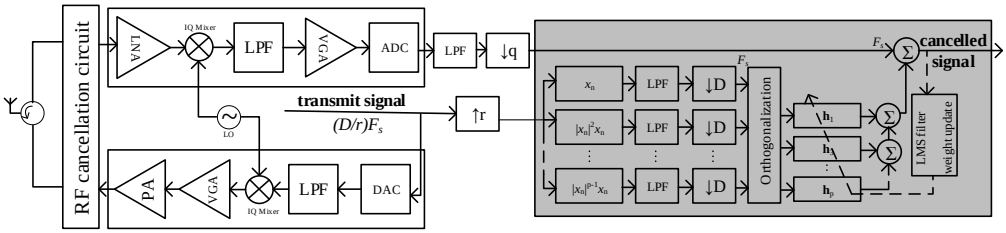


Figure 4.1: The overall structure of a full-duplex transceiver, where the grey part is implemented in software in this work.

4.1 Related Work

SI cancellation methods have been extensively researched, as it is the main challenge in realizing full-duplex communications systems. The work presented in [64] describes an RF canceller architecture, while [65] proposes an all-digital SI cancellation technique. In some works, such as [66], both analogue and digital cancellation methods are used for sufficient SI suppression. Moreover, taking advantage of several stages of cancellation such as, propagation, analogue, and digital domain cancellation, is also reported in some works available in the literature [67, 68].

Furthermore, some actual prototypes capable of full-duplex communication have been built and presented in [67, 69–71]. However, very few reports of actual hardware or software-based implementation of full-duplex systems can be found in the literature. In [72], parts of the SI cancellation methods presented in [29] are implemented on FPGA. This work reports the achieved performance in terms of SI cancellation and lacks the numerical result related to the implementation such as execution time, power, and energy consumption.

Some implementations of LMS-based adaptive filtering, which is one of the main implemented functional blocks in this work, can be found in literature. The authors of [73] have proposed an FPGA implementation for a 12-tap LMS-based adaptive filter on Xilinx DSP48. The proposed design can run at maximum 500 MHz clock frequency, and consumes approximately 158 mW static power.

Furthermore, a GPU based implementation of adaptive filtering can be found in [74], where the authors present a multichannel adaptive equalization system based on the filtered-x LMS algorithm in the context of audio processing applications. In this work, CUDA has been used as the programming language. The processing time for different input sample sizes are presented and suitability of GPUs for such applications has been demonstrated.

There are also some works reporting implementation of polyphase filters, another block in the digital SI canceller, using FPGAs and GPUs [40–42]. These works were already introduced in more details in the previous chapter.

Some of the existing works in the literature which investigate digital pre-distortion techniques use arithmetic operations similar to SI cancellation methods adopted in the work of this thesis [75–77]. In these works, parallel processing on GPUs and CPUs is utilized for better performance. The achieved performance is evaluated and presented, however, power or energy consumption of the designs are not reported.

The work in this thesis goes further in this topic by demonstrating the feasibility of real-time complete digital SI cancellation using COTS platforms while staying within the limits of power consumption of mobile devices.

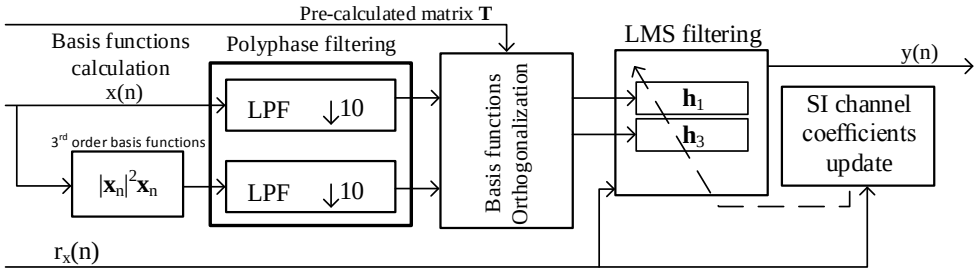


Figure 4.2: The structure of the implemented third-order digital SI canceller

4.2 Digital Self-Interference Cancellation

This section introduces the digital canceller blocks implemented in the scope of this research. The overall structure of a full-duplex transmitter is shown in Fig. 4.1, in which the nonlinear digital SI canceller is highlighted in the grey part. Furthermore, the actual third-order digital canceller implemented for this work is illustrated in Fig. 4.2.

The building blocks, depicted in Fig. 4.2, are briefly described in the following.

Basis Functions Calculation First, basis functions are generated from the nonlinear transformations of the known transmit signal. For each transmitted sample $x(n)$, the pth-order basis function is calculated using $u_p(n) = |x(n)|^{p-1}x(n)$. The highest nonlinearity order used in this work is $P = 3$.

Polyphase Filtering Looking at Fig. 4.1, it can be seen that the transmitted signal is oversampled before going through the digital canceller. Thus, the generated basis functions should be resampled to the final cancellation signal's sample rate. With a decimation factor of D, every D-th sample of the lowpass filter output is kept.

However, with this approach, many signal samples are processed which would eventually be discarded. As processing efficiency is critical in our software-based implementation, we eliminate the unnecessary computations by using polyphase filtering.

Fig. 4.3 depicts the structure of a polyphase filter with downsampling factor D. Here, $F_0, \ldots, F_{D-1}$ denote sub-filters of length G, comprising the overall polyphase filter. Thus, the total length of the filter is $G \times D$. The implemented polyphase for the digital canceller has an overall length of 20, stemming from $D = 10$ sub-filters of length $G = 2$.

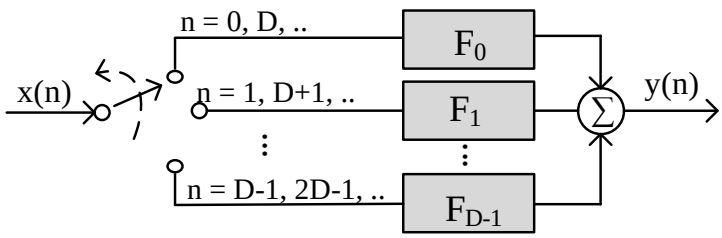


Figure 4.3: Functional structure of a polyphase filter with decimation factor D, where $y_(n)$ represents the signal samples after downsampling and filtering $x(n)$.

Various approaches are used for the OpenCL implementation with the aim of achieving the highest possible performance. Implementations based on both scalar and vector data types are carried out. Filter coefficients are re-organized to allow more efficient data loads.

One example of OpenCL kernel implementation and workload distribution for polyphase filtering is shown in Fig. 4.4. Here, the re-arranged coefficients and data samples are loaded as vectors of length four into vectors $\mathbf{p}$ and $\mathbf{x}$, respectively. Each work-item generates one output sample $\mathbf{y}[n]$ after multiplication and summation. In Fig. 4.4, k is the polyphase filter length, where $k = G \times D$. Number of work groups is denoted by n, and local size is assumed to be equal to 16 for a clearer presentation.

Computing orthogonalization matrix The transformation matrix is calculated using equations (2.30) - (2.33). This matrix only depends on the statistical properties of the transmit signal, and does not vary over time. Hence, in this work, we assume that it is precomputed to reduce the computational complexity of the design. Having nonlinearity order $P = 3$, the orthogonalization matrix $\mathbf{T}$ is a 2×2 matrix.

Basis function orthogonalization Using the precomputed transformation matrix $\mathbf{T}$, the basis functions are orthogonalized according to (2.34). This step is carried out to speed up the convergence of the learning process.

LMS filtering At this stage, the orthogonalized basis functions are filtered using the SI channel coefficients. To create a more accurate model of the SI channel memory, both pre-cursor and post-cursor taps are assumed. Thus, the overall channel memory, i.e., the filter length can be shown as $L = (L_{pre} + L_{post} + 1) \times (\frac{P+1}{2})$. The error signal $e(n)$ is calculated by subtracting the filtered basis functions from the received signal $r_x(n)$. This describes step six of the LMS learning method defined in Algorithm 1.

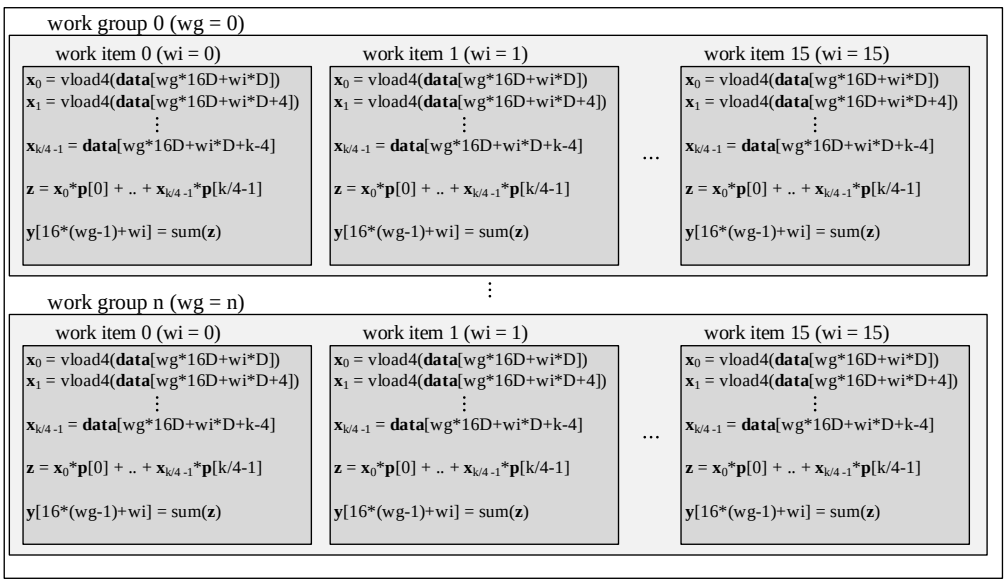


Figure 4.4: OpenCL kernel structure and workload distribution for the polyphase filter.

SI channel coefficients update The SI channel coefficients should be updated as described in steps 7-10 of Algorithm 1. To address the difference of strength between the nonlinear terms in the received signal, different step sizes should be considered for different nonlinear terms. We have selected $\boldsymbol{\mu} = 0.01$ and $\boldsymbol{\mu} = 0.001$ for the linear and third order terms, respectively.

In order to add more parallelism to the computations and reduce complexity, the SI channel coefficients are only updated after a set of N samples are processed. With this approach, the *LMS filter* kernel would stall less frequently while waiting for the updated coefficients. Thus, the *LMS filter* and *SI channel coefficients update* kernels will have less dependency, which increases the parallelism by processing larger sets of data in the *LMS filter* kernel.

4.3 Results

This section provides implementation results and analysis for the proposed digital canceller. To demonstrate the performance of our implemented canceller, we first present the effectiveness and efficiency of our solution in SI suppression. For this purpose, we use the data from an actual full-duplex prototype system, built in Laboratory of Electronics and Communications Engineering of Tampere University of Technology. More details on this full-duplex system prototype can be found in [29] and [78].

Secondly, the software-based digital canceller implementation is evaluated in terms of execution time, power, and energy consumption. The results are used to investigate the feasibility of the selected COTS platforms for this software-based implementation.

Software Development The digital canceller OpenCL kernels are optimized for each platform. Selecting a suitable workload distribution among the OpenCL work-items, using scalar or vector based implementation, and employing different vector lengths can affect the performance of the processors in each task.

With the Mali GPU, employing different kernel designs showed that the best results can be achieved when vectors of four elements are used. On the A15 CPU, using different vector lengths and even scalar data types yielded similar results. However, organizing the workload in two work groups resulted in the lowest execution time. The Core i7 CPU has the best performance when using vectors of length 16, and in most cases dividing the processing among eight work groups provides better results. Similar to Mali, the Adreno GPU performs best when having vectors of length four. Moreover, using four work groups has shown to result in faster execution on Adreno.

The performance results of the different processing platforms, presented in the following sections, are achieved when running the most efficient kernel design setups.

4.3.1 Digital Self-Interference Canceller Performance

The software-based implementation uses the sample data, i.e., transmit signal and received signal, from the real full-duplex prototype system and runs the digital canceller kernels. The resulting cancelled signal on the Adreno GPU is plotted in Fig. 4.5, which shows the instantaneous power of the cancelled signal over time, when input buffers of 10, 1280, and 2560 samples are used. This means that after decimation by a factor of 10 and orthogonalization, the input samples are processed as a single sample, or in blocks of 128, and 256, before the SI channel coefficients are updated.

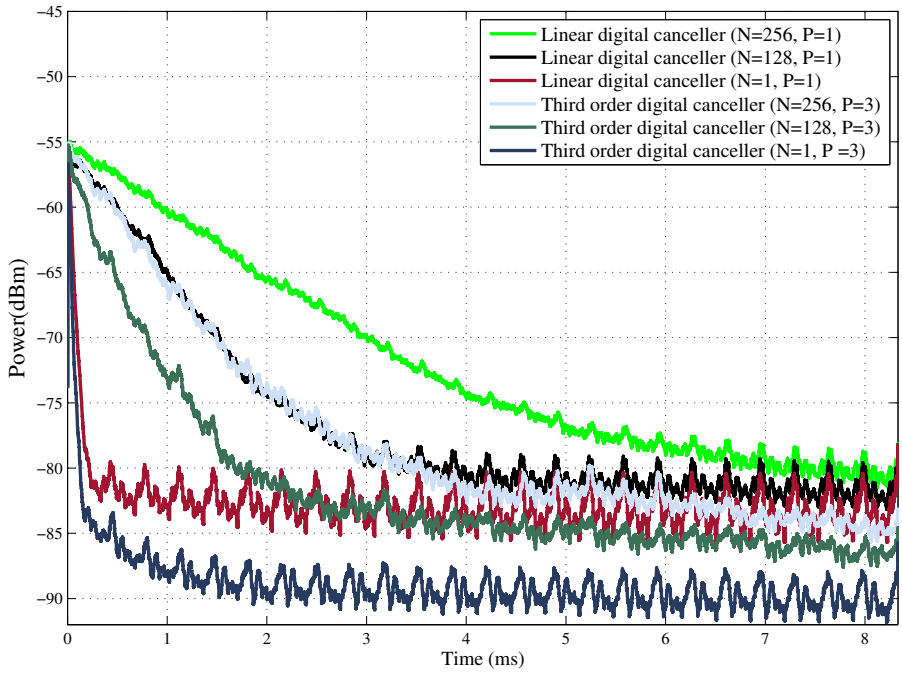


Figure 4.5: The instantaneous power of the SI signal, averaged over 1000 samples, of linear ($P = 1$) and third order ($P = 3$) digital canceller output signal, implemented on the Adreno 430, with respect to time, for $N = 1$, $N = 128$ and $N = 256$.

In this implementation, $L_{pre} = 7$ and $L_{post} = 8$. Thus, the overall SI channel memory L, $L = (L_{pre} + L_{post} + 1) \times (\frac{P+1}{2})$, is considered to be 16 for the linear canceller, and 32 for the third order canceller.

It can be seen from Fig. 4.5 that the SI signal is sufficiently suppressed, reaching the receiver noise floor (-90 dBm). When the learning algorithm is given more time to converge, almost perfect SI cancellation can be achieved. For every N, the performance of the third order canceller is shown to be superior to the linear canceller.

Fig. 4.5 also shows that higher N, i.e. less frequent updating of the SI channel coefficients, results in slower convergence of the LMS learning algorithm. However, this can be neglected as there is relatively small difference, especially after the initial learning phase. Thus, we can consider using higher N as a feasible approach for reducing the computational complexity of the digital canceller.

4.3.2 Execution Time

This section presents the measured execution times for different building blocks of the digital SI canceller. The results include the measured times on all four processing platforms, introduced in section 3.3 and in the beginning of this chapter.

The main advantage of using OpenCL on multicore platforms with SIMD or single program, multiple data (SPMD) optimized hardware is being able to better utilize the available parallel resources in order to exploit the existing data-level parallelism. When PEs of the processor are

Table 4.1: Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Adreno 430* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	1,89	-	1,50	-	1,37	-	1,21
Polyphase	23,00	44,10	16,00	30,50	13,75	26,50	12,21	22,75
Orthogonalization	11,00	18,00	5,50	11,50	2,75	5,75	2,25	4,75
LMS filter	23,00	32,76	17,00	23,28	14,25	20,05	12,75	18,32
Weight update	11,00	11,00	5,50	5,50	2,75	2,85	1,38	1,27
Total [ns]	68,00	107,75	44,00	72,28	33,50	56,52	28,59	48,30
Rate [MHz]	14,71	9,29	22,73	13,84	29,85	17,69	34,98	20,70

used efficiently and workload is distributed properly among the resources, high performance can be achieved.

We increase the amount of data processed in each kernel call to further add to the inherent parallelism of the algorithm. Thus, processing time for each signal sample is reduced. Furthermore, the implementation of each block is tailored on each platform for better execution efficiency.

Tables 4.1 - 4.4 present the execution times of each digital canceller block on the four platforms. It should be noted that the data transfer times are not included in the reported times, as the SoC can be designed so that the processor sees the same memory as the radio hardware. In each table, the execution times for both the linear and third order cancellers in case of buffer lengths of 2560, 5120, 10240, 20480 are presented. The input buffer sizes are powers of two numbers multiplied by the decimation factor, which is equal to 10.

It can be seen that the processing time for a single signal sample decreases as the buffer sizes increase. In most scenarios, the execution time is halved when the buffer size is doubled. This can be clearly observed in the "orthogonalization" and "weight update" kernels. However the achieved speedup is smaller for the two kernels which perform filtering, i.e. "polyphase" and "LMS filter". This is due to the inherent lack of parallelism stemming from the summation step of the convolution operation.

The achieved speedup for the "basis functions" kernel is not increasing linearly as the buffer sizes grow. The reason behind this could be the 10 times bigger input buffers of this kernel compared to the other kernels, executed after downsampling. The bigger input buffers could result in saturation

Table 4.2: Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Cortex A15* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	34,76	-	17,96	-	10,25	-	5,85
Polyphase	312,50	622,50	207,03	411,52	122,07	242,49	77,63	152,94
Orthogonalization	320,31	328,12	164,02	164,06	82,03	84,96	44,92	45,41
LMS filter	398,43	476,56	222,65	306,64	134,76	214,84	92,77	167,96
Weight update	265,62	242,18	142,57	125,00	83,00	81,05	42,96	40,52
Total [ns]	1296,80	1704,10	736,27	1025,20	421,86	633,59	258,28	412,68
Rate [MHz]	0,77	0,59	1,36	0,97	2,37	1,57	3,87	2,41

Table 4.3: Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Core i7* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	1,92	-	0,66	-	0,55	-	0,48
Polyphase	20,78	39,86	12,02	22,34	9,64	17,97	6,12	11,38
Orthogonalization	5,93	7,42	3,71	4,45	2,22	2,59	1,29	2,22
LMS filter	23,75	22,26	12,00	12,90	9,64	10,02	6,30	6,49
Weight update	5,93	7,42	3,71	3,71	1,85	1,85	0,92	0,92
Total [ns]	56,39	78,88	31,44	44,06	23,35	32,98	14,63	21,49
Rate [MHz]	17,73	12,67	31,80	22,69	42,82	30,32	68,35	46,53

of the available PEs on the processing platforms, and thus lowering the speedup.

The achieved sample production rates with respect to the input buffer lengths is depicted in Fig. 4.6. It can be seen that the growth seems to slow down with very long buffers as a result of the saturation of the PEs.

The presented results reveal that the implemented digital canceller on Mali and A15 cannot reach production rates close to 20 MHz, even when using big input buffers. However, having input buffers of 5120 samples, both the Adreno 430 GPU and the Core-i7 CPU can perform linear digital SI cancellation at rates over 20 MHz. When it comes to third order cancellation, input buffers of 5120, and 20480 samples are required for the Core-i7 and the Adreno 430, respectively, to carry out the related processing for above 20 MHz sample rates.

For a third order digital canceller, two polyphase filters are employed, one for the linear terms, and one for the third order basis functions. Consequently, as it can be observed from Tables 4.1 - 4.4, this processing stage takes twice more time for the nonlinear canceller compared to the linear one. Furthermore, the linear canceller is not using the "basis functions" kernel, which calculates the third order basis functions. The execution time for the rest of the third order canceller blocks is either equal or slightly different than the linear one. This is due to the fact that there are only minor differences in required number of arithmetic computations.

Table 4.4: Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Mali-T628* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	3,41	-	2,49	-	2,21	-	2,02
Polyphase	46,14	90,11	39,66	77,68	36,08	71,09	33,20	64,85
Orthogonalization	19,33	40,97	10,36	21,62	5,43	12,31	2,46	6,70
LMS filter	65,79	87,16	44,52	72,32	37,83	55,51	28,48	50,41
Weight update	30,51	25,19	16,08	16,84	6,15	8,44	3,43	4,54
Total [ns]	161,77	246,84	110,62	190,95	85,49	149,56	67,57	128,52
Rate [MHz]	6,18	4,05	9,04	5,23	11,70	6,68	14,80	7,78

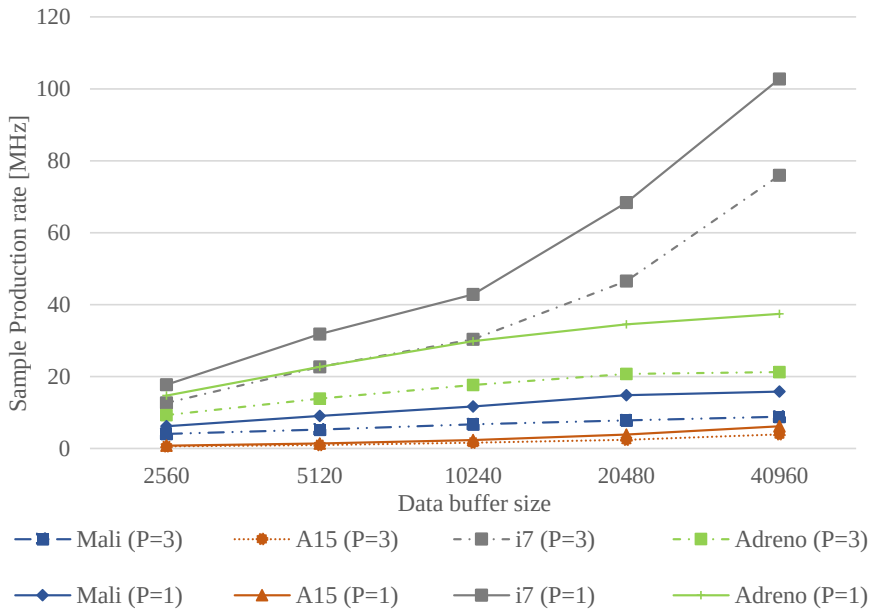


Figure 4.6: Sample production rate increase with regards to buffer size on the four platforms for both linear and third order cancellers.

4.3.3 Delay

As extensively discussed before, we increase the number of samples processed in each kernel call to add to the existing parallelism, which would be utilized by the available PEs. However, there is a downside to this approach, as having bigger input buffers translates to longer delays for the system. Thus, a balance should be achieved in the delay and sample production rate trade-off in a real application. The overall delay for the implemented canceller is calculated as:

$$\begin{aligned} overall\ delay = & T_{basisfunctions} \times buffer\ size \\ & + T_{polyphase} \times \frac{buffer\ size}{D} \\ & + T_{orthogonalization} \times \frac{buffer\ size}{D} \\ & + T_{LMS\ filter} \times \frac{buffer\ size}{D} \\ & + T_{weight\ update} \times \frac{buffer\ size}{D}, \end{aligned} \tag{4.1}$$

where T_{kernel} is the execution time of one sample for "kernel", and D is the decimation factor. Table 4.5 lists the produced overall delays using different buffer sizes on the selected platforms.

Having input buffer sizes of 5120 and 10240 for the third order canceller implemented on Core i7 and Adreno 430, respectively, results in delays of $25,6\ \mu$s and $70,5\ \mu$s. The inherent receiver processing latency of LTE UE is, at least, 1 ms due to the downlink reference symbol structure, the adopted codeword mapping, and interleaving processing. Furthermore, according to 3GPP specification [79], an additional processing time of 3 ms is allowed for sending downlink hybrid

Table 4.5: Overall delay in microseconds for different buffer lengths on all four platforms.

Buffer length	2560		5120		10240		20480	
Nonlinearity order	P=1	P=3	P=1	P=3	P=1	P=3	P=1	P=3
Mali	41,41	71,04	56,63	109,24	87,54	173,51	42,08	300,44
A15	331,99	516,34	376,97	607,65	431,98	743,26	42,08	952,07
i7	14,43	24,61	16,09	25,60	23,91	38,83	42,08	52,85
Adreno	17,40	31,93	22,52	43,91	34,30	70,50	59,38	121,22

automatic repeat request (HARQ) acknowledgement within uplink control signaling. Thus, the aforementioned delays can be considered more than reasonable.

4.3.4 Power Consumption

The same tools and approaches, described in section 3.3 for measuring the consumed power on the Odroid board, were used in this work to estimate the power consumed by the Mali GPU and A15 CPU. However, similar to the Core-i7, power measurement on the Adreno GPU is not possible due to unavailability of relative tools.

The average consumed power, presented in Fig. 4.7 is measured when the input buffer size is set to 5120 signal samples for the kernels, since the average power varies with buffer size only slightly, if any. Furthermore, the figure shows that there is insignificant or no difference between the power consumption of the implemented linear and third order SI cancellers.

The basis function calculation is not performed in the linear case, thus it is not included here. The overall average power consumption is also measured when the whole digital canceller is run on the cores, which is shown in the chart labelled as “total”. The total power consumption is somewhat higher than the average power of all implemented blocks.

Now comparing the power consumption of the two platforms reveals that the A15 CPU uses

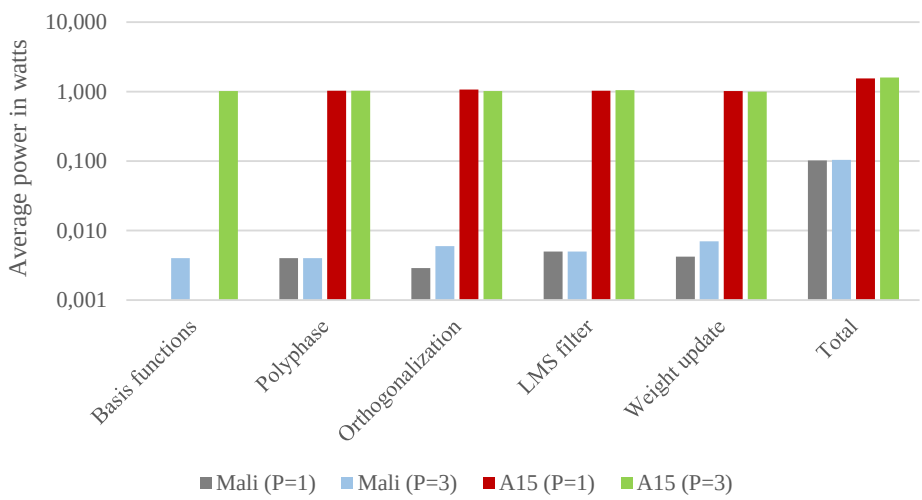


Figure 4.7: Consumed power by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

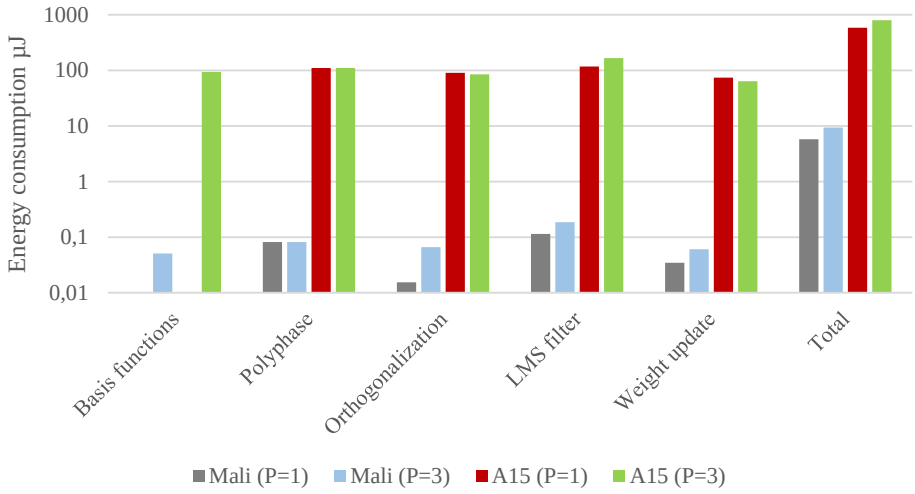


Figure 4.8: Consumed energy by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

approximately 20 times more power than the Mali GPU while running the same kernels. This could stem from the higher clock frequency of the CPU, which is 2.1 GHz, compared to 600 MHz 0f the Mali GPU. Another reason is that the CPU is equipped with extra hardware for more general-purpose computing, which results in higher power consumption.

Power consumption can be reduced by increasing parallelism to reduce the required clock frequency for achieving the same throughput. In other words, reducing switching activity, and more importantly, the voltage which has quadratic effect on power, reduces the consumed power [80].

Having input buffer size of 5120 samples, the power consumption of Mali is roughly 104 mW when running the third order digital canceller kernels. According to [81], the overall power consumption of an LTE UE receiver is close to few watts. Thus, with regards to power consumption, Mali can be considered an eligible candidate for hand-held devices.

4.3.5 Energy Consumption

While power consumption is important for heating matters of the device, evaluating the consumed energy is essential, as it translates to the battery life of the device. Thus, energy consumption is a key criterion, especially in hand-held devices.

Using the measured average power consumption and the delays corresponding to processing of 5120 samples, we calculated the consumed energy for each kernel. The results can be found in Fig 4.8.

As power consumption remains relatively constant with bigger input buffer size, and the execution time increases, it can be concluded that the energy consumption also rises with longer buffers.

It can be seen from Fig 4.8 that A15 has a higher energy consumption compared to Mali. This could be explained by higher power consumption and execution time of the kernels in A15. The total energy consumed when running the complete canceller is shown in the chart labelled as "total". The third order digital canceller consumes approximately 9 μJ when processing 5120 signal samples on the Mali GPU.

CHAPTER 5

CONCLUSION

This chapter summarizes the main results and findings of the Thesis. Furthermore, some open issues to be considered as the continuation of this work are laid out.

5.1 Summary and Main Results

SDR solutions for the IEEE 802.11ac transceiver baseband processing were proposed in publications [P1], [P2], and [P3]. Targeting very high throughputs, the 802.11ac amendment imposes very strict requirements for the processing platform, making a software-based implementation more challenging. A Tensilica DSP core was used to implement four MIMO transmission scenarios. Then, to investigate the feasibility of real-time processing for the DATA and VHT-LTF symbols, the solution was evaluated in terms of number of clock cycles and power consumption. To have real time processing of the symbols, the processing should take less than the symbol duration, which is 4μs for the header part and 3.6μs for the data part when short Guard Interval (GI) is used. The obtained results suggest that a real-time processing of the transmitter baseband operations can be achieved with clock frequencies as low as 500 MHz on the Tensilica BBE32 core. On the receiver side, however, clock frequencies higher than 1 GHz and 2 GHz are required for the two and four antenna MIMO configurations, respectively. Furthermore, the estimated power consumption indicates the feasibility of deploying the developed SDR solution in hand-held devices, as it is much lower than the reported power consumption of WiFi in mobile devices.

Following the findings on the IEEE 802.11ac baseband processing, [P4] presented the results from the implemented DFE channelization concept for the same standard. Different approaches were used to divide the IEEE 802.11ac 80 MHz bandwidth, into two 40 MHz waveforms to enable parallel processing of the two signals. The proposed solutions for channelization filtering and decimation were implemented using COTS multi-core CPUs and GPUs. The execution time, power, and energy consumption were measured on selected platforms. The results demonstrated the performance enhancement achieved by optimally utilizing the available parallel resources. Furthermore, a comparison among the employed platforms was carried out, and it showed that some of the solutions could fulfill the strict timing requirements of the IEEE 802.11ac standard.

Publications [P5] and [P6] presented the software-based implementation of an adaptive nonlinear digital SI canceller for future IBFD systems. General-purpose low-cost COTS processing

platforms, suitable for hand-held devices were selected to demonstrate the feasibility of a true SDR solution. The software was tailored to efficiently take advantage of the existing parallel resources of the multi-core processors. Delay, execution time, power and energy consumption were measured on the platforms. The proposed solution was shown to be capable of cancelling the SI at the required sample rate for a 20 MHz LTE carrier bandwidth. Furthermore, the results from power and energy consumption indicated the feasibility of a mobile-scale deployment, when compared to the estimated power consumption of an LTE UE receiver reported in the literature.

5.2 Future Work

While the results obtained in this thesis work show the feasibility of SDR based implementations of different computationally intensive algorithms, many steps still need to be taken towards achieving fully software defined radios. Extending the implementations to cover all the baseband and DFE functionalities, and also including the MAC layer processing is one issue that should be considered in the continuation of this work.

Furthermore, to realize the vision of multi-standard SDR systems, the possibility of employing a single platform to operate different standards, e.g., 5G from the 3GPP and IEEE 802.11ac from the WLAN family should be investigated. The different application areas within one standard could also raise some issues when adopting a single SDR platform. As an example, 5G targets three main dimensions for performance improvement, namely enhanced mobile broadband, ultra-reliable low-latency communications, and massive machine type communications. These impose different constraints on the processing platform, which add to the complexity of providing a single SDR solution. These issues need to be addressed as part of the future SDR research.

Although great advances in performance of processing units have been made, the processing power of these platforms is still somewhat limited. Therefore, the issue of scaling the proposed solutions, e.g., for processing of massive MIMO communication systems, remains an open research topic.

From the implementation point of view, other SDR platforms with different capabilities need to be studied. As an example, in this thesis the parallelization was only carried out by optimally dividing the available work among the parallel units of a CPU or GPU. To achieve higher performance, platforms should be employed which allow the distribution of the workload between the CPU, and one or more GPUs.

Further reduction in energy consumption has also been identified as an important topic for future work. To be able to draw more precise conclusions from the energy consumption point of view, a model of how the energy is spent can be developed to help proposing changes in the correct direction for the implementation.

BIBLIOGRAPHY

[1] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.

[2] E. Grayver, *Implementing Software Defined Radio*. New York, NY, USA: Springer-Verlag, 2013.

[3] W. H. Tuttlebee, Ed., *Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations*. Chichester, England: Wiley, 2003.

[4] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados, "Addressing next-generation wireless challenges with commercial software-defined radio platforms," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 59–67, January 2016.

[5] H. Ishikawa, "Software defined radio technology for highly reliable wireless communications," *Wireless Personal Communications*, vol. 64, no. 3, pp. 461–472, Jun 2012.

[6] M. Gast, *802.11ac: A Survival Guide Wi-Fi at Gigabit and Beyonds*. O'Reilly Media, 2013.

[7] I. P802.11ac/D5.0 2013, "IEEE standard for information technology– telecommunications and information exchange between systemslocal and metropolitan area networks– specific requirements–part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 GHz," IEEE, Piscataway, NJ, Standard, January 2013.

[8] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637–1652, Sept 2014.

[9] D. W. Bliss, P. A. Parker, and A. R. Margetts, "Simultaneous transmission and reception for improved wireless network performance," in *IEEE/SP Workshop on Statistical Signal Processing*, Madison, WI, USA, USA, Aug 2007, pp. 478–482.

[10] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single channel, full duplex wireless communication," in *Proceedings of the Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10, Chicago, IL, 2010, pp. 1–12.

[11] Z. Zhang, K. Long, A. V. Vasilakos, and L. Hanzo, "Full-duplex wireless communications: Challenges, solutions, and future research directions," *Proceedings of the IEEE*, vol. 104, no. 7, pp. 1369–1409, July 2016.

[12] D. Korpi, "Full-duplex wireless: Self-interference modeling, digital cancellation, and system studies," Ph.D. dissertation, Tampere University of Technology, Finland, 2017.

[13] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the future of parallel computing," *IEEE Micro*, vol. 31, no. 5, pp. 7–17, Sept 2011.

[14] I. S. 802.11-2012, "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE, Piscataway, NJ, Standard, 2012.

[15] E. Perahia and R. Stacey, *Next Generation Wireless LANs Throughput, Robustness, and Reliability in 802.11n*. New York, NY, USA: Cambridge University Press, 2013.

[16] "802.11ac: The fifth generation of Wi-Fi," Cisco, White paper, May 2017, last accessed 18.06.2017. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-3600-series/white_paper_c11-713103.html

[17] "IEEE 802.11ac: The next evolution of Wi-Fi™ standards," May 2012, last accessed 18.06.2017. [Online]. Available: https://www.qualcomm.com/media/documents/files/ieee802-11ac-the-next-evolution-of-wi-fi.pdf

[18] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct 1998.

[19] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 107–111, 03 1951.

[20] S. Kumawat, R. Shrestha, N. Daga, and R. Paily, "High-throughput LDPC-decoder architecture using efficient comparison techniques & dynamic multi-frame processing schedule," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 5, pp. 1421–1430, May 2015.

[21] Q. Xie, Q. He, X. Peng, Y. Cui, Z. Chen, D. Zhou, and S. Goto, "A high parallel macro block level layered LDPC decoding architecture based on dedicated matrix reordering," in *2011 IEEE Workshop on Signal Processing Systems*, Beirut, Lebanon, Oct 2011, pp. 122–127.

[22] S. Huang, D. Bao, B. Xiang, Y. Chen, and X. Zeng, "A flexible LDPC decoder architecture supporting two decoding algorithms," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, Paris, France, May 2010, pp. 3929–3932.

[23] S. K. Mitra and J. F. Kaiser, Eds., *Handbook for Digital Signal Processing*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[24] J. Yli-Kaakinen, T. Levanen, M. Aghababaeetafreshi, M. Renfors, and M. Valkama, "Optimization of parallel processing intensive digital front-end for IEEE 802.11ac receiver," in *European Signal Processing Conference*, Budapest, Hungary, Aug 2016, pp. 637–641.

[25] *Fundamentals of 5G Mobile Networks*, 1st ed. Wiley Publishing, 2015.

[26] Q. C. Li, H. Niu, A. T. Papathanassiou, and G. Wu, "5G network capacity: Key elements and technologies," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 71–78, March 2014.

[27] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge University Press, 2005.

[28] T. Riihonen, S. Werner, and R. Wichman, "Mitigation of loopback self-interference in full-duplex MIMO relays," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 5983–5993, Dec 2011.

[29] D. Korpi, Y. S. Choi, T. Huusari, L. Anttila, S. Talwar, and M. Valkama, "Adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio: Algorithms and RF measurements," in *IEEE Global Communications Conference*, San Diego, CA, USA, 6-10 Dec 2015, pp. 1–7.

[30] M. Nagaraju and M. Rakesh, "High-speed and low-power ASIC implementation of OFDM transceiver based on WLAN (IEEE 802.11a)," in *International Conference on Devices, Circuits and Systems*, Coimbatore, India, March 2012, pp. 436–439.

[31] J. Son, I. G. Lee, and S. K. Lee, "ASIC implementation and verification of MIMO-OFDM transceiver for wireless LAN," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Athens, Greece, Sept 2007, pp. 1–5.

[32] S. Yoshizawa and Y. Miyanaga, "VLSI implementation of a 4x4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," in *IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 1743–1746.

[33] P. Wang, J. McAllister, and Y. Wu, "Software defined FFT architecture for IEEE 802.11ac," in *2013 IEEE Global Conference on Signal and Information Processing*, Austin, TX, USA, Dec 2013, pp. 1246–1249.

[34] N. Yoshida, L. Lanante, Y. Nagao, M. Kurosaki, and H. Ochi, "A hybrid HW/SW 802.11ac/ax system design platform with ASIP implementation," in *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Xiamen, China, Nov 2017, pp. 827–831.

[35] D. Liu, "Baseband ASIP design for SDR," *China Communications*, vol. 12, no. 7, pp. 60–72, July 2015.

[36] H. Yang, J. Shim, J. Bang, and Y. Lee, "Software-based giga-bit WLAN platform," in *IEEE International Conference on Consumer Electronics*, Las Vegas, NV, USA, Jan 2014, pp. 478–479.

[37] M. Li, A. Amin, R. Appeltans, A. Folens, U. Ahmad, H. Cappelle, P. Debacker, L. Hollevoet, A. Bourdoux, P. Raghavan, A. Dejonghe, and L. V. D. Perre, "A C-programmable baseband processor with inner modem implementations for LTE Cat-4/5/7 and Gbps 80mhz 4x4 802.11ac (invited)," in *IEEE Global Conference on Signal and Information Processing*, Austin, TX, Dec 2013, pp. 1222–1225.

[38] Y. H. Park, K. Prasad, Y. Lee, K. Bae, and H. Yang, "Scalable radio processor architecture for modern wireless communications," in *International Conference on Field-Programmable Technology*, Shanghai, China, Dec 2014, pp. 310–313.

[39] B. Mei, A. Lambrechts, J. Y. Mignolet, D. Verkest, and R. Lauwereins, "Architecture exploration for a reconfigurable architecture template," *IEEE Design Test of Computers*, vol. 22, no. 2, pp. 90–101, March 2005.

[40] V. Mocanu, C. Anghel, and A. A. Enescu, "FPGA implementation of a digital front end block for a multi-carrier multi-antenna system," in *2009 International Semiconductor Conference*, vol. 2, Sinaia, Romania, Oct 2009, pp. 431–434.

[41] P. Fiala and R. Linhart, "High performance polyphase FIR filter structures in VHDL language for software defined radio based on FPGA," in *2014 International Conference on Applied Electronics*, Pilsen, Czech Republic, Sept 2014, pp. 83–86.

[42] A. Al-safi and B. Bazuin, "GPU based implementation of a 64-channel polyphase channalizer," in *2015 IEEE Dallas Circuits and Systems Conference (DCAS)*, Dallas, TX, USA, Oct 2015, pp. 1–4.

[43] "CUDA C programming guide version 9.2," NVIDIA Corporation, Tech. Rep., August 2018.

[44] F. T. Gebreyohannes, A. Frappé, and A. Kaiser, "A configurable transmitter architecture for IEEE 802.11ac and 802.11ad standards," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 1, pp. 9–13, Jan 2016.

[45] L. Fei-yu, Q. Wei-ming, Z. Jian-chuan, N. Gang-yang, L. Wei-bin, and M. Wei-yu, "Programmable digital front-end design for software defined radio," in *International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 1, Wuhan, Hubei, China, April 2010, pp. 321–324.

[46] G. Hueber, R. Stuhlberger, A. Holm, and A. Springer, "Multi-mode receiver design for wireless terminals," in *European Conference on Wireless Technologies*, Munich, Germany, Oct 2007, pp. 126–129.

[47] "Connx BBE32 DSP, user's guide," Tensilica Incorporated, 2012.

[48] C. Tang, C. Liu, L. Yuan, and Z. Xing, "High precision low complexity matrix inversion based on newton iteration for data detection in the massive MIMO," *IEEE Communications Letters*, vol. 20, no. 3, pp. 490–493, March 2016.

[49] J. A. Zhang, X. Huang, H. Suzuki, and Z. Chen, "Gaussian approximation based interpolation for channel matrix inversion in MIMO-OFDM systems," *IEEE Transactions on Wireless Communications*, vol. 12, no. 3, pp. 1407–1417, March 2013.

[50] J. E. Gentle, *Numerical Linear Algebra for Applications in Statistics*. New York, NY: Springer, 1998.

[51] C. K. Singh, S. H. Prasad, and P. T. Balsara, "VLSI architecture for matrix inversion using modified gram-schmidt based QR decomposition," in *Proceedings of International Conference on VLSI Design*, ser. VLSID '07, Bangalore, India, 2007, pp. 836–841.

[52] R. Friedman, A. Kogan, and Y. Krivolapov, "On power and throughput tradeoffs of WiFi and Bluetooth in smartphones," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1363–1376, July 2013.

[53] *ODROID-XU3.*, Hardkernel co., Ltd., 2013, last accessed 08.04.2017. [Online]. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127

[54] "big.LITTLE technology: The future of mobile," ARM, Tech. Rep., 2013, last accessed 30.07.2017. [Online]. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf

[55] *ARM® Cortex® -A15 MPCore™ Processor*, ARM Ltd., 2013, last accessed 30.07.2017. [Online]. Available: https://static.docs.arm.com/ddi0438/i/DDI0438.pdf

[56] *Cortex®-A7 MPCore™*, ARM Ltd., 2013, last accessed 30.07.2017. [Online]. Available: https://static.docs.arm.com/ddi0464/f/DDI0464.pdf

[57] *The ARM® Mali™ Family of Graphics Processors*, ARM Ltd., February 2013, last accessed 08.04.2017. [Online]. Available: http://malideveloper.arm.com/downloads/events/2013/GDC/0319-11%20Mali%20Minibook_TB.pdf

[58] P. Harris, "The Mali GPU: An abstract machine," 2014, last accessed 08.04.2017. [Online]. Available: https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core

[59] I. S. 754-2008, "IEEE standard for floating-point arithmetic," IEEE, Piscataway, NJ, Standard, Aug 2008.

[60] *Intel® Core™ i7 Processor Family for LGA2011 Socket*, Intel Corporation, May 2014, last accessed 08.04.2017. [Online]. Available: http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-i7-lga2011-datasheet-vol-1.html

[61] "The OpenCL specification version 2.0," Khronos Group, Tech. Rep., July 2015.

[62] *Intel® Core™ i7-4800MQ Processor*, Intel Corporation, 2018, last accessed 15.08.2018. [Online]. Available: https://ark.intel.com/products/75128/Intel-Core-i7-4800MQ-Processor-6M-Cache-up-to-3_70-GHz

[63] *Snapdragon 810 processor product brief*, Qualcomm Technologies, February 2015, last accessed 08.04.2017. [Online]. Available: https://www.qualcomm.com/media/documents/files/snapdragon-810-processor-product-brief.pdf

[64] K. E. Kolodziej, J. G. McMichael, and B. T. Perry, "Multitap RF canceller for in-band full-duplex wireless communications," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 4321–4334, June 2016.

[65] E. Ahmed and A. M. Eltawil, "All-digital self-interference cancellation technique for full-duplex systems," *IEEE Transactions on Wireless Communications*, vol. 14, no. 7, pp. 3519–3532, July 2015.

[66] M. Duarte and A. Sabharwal, "Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results," in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, Nov 2010, pp. 1558–1562.

[67] M. Heino, D. Korpi, T. Huusari, E. Antonio-Rodriguez, S. Venkatasubramanian, T. Riihonen, L. Anttila, C. Icheln, K. Haneda, R. Wichman, and M. Valkama, "Recent advances in antenna design and interference cancellation algorithms for in-band full duplex relays," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 91–101, May 2015.

[68] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637–1652, Sept 2014.

[69] M. Duarte, A. Sabharwal, V. Aggarwal, R. Jana, K. K. Ramakrishnan, C. W. Rice, and N. K. Shankaranarayanan, "Design and characterization of a full-duplex multiantenna system for WiFi networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1160–1177, March 2014.

[70] M. Duarte, C. Dick, and A. Sabharwal, "Experiment-driven characterization of full-duplex wireless systems," *IEEE Transactions on Wireless Communications*, vol. 11, no. 12, pp. 4296–4307, December 2012.

[71] M. Mikhael, B. van Liempd, J. Craninckx, R. Guindi, and B. Debaillie, "An in-band full-duplex transceiver prototype with an in-system automated tuning for RF self-interference cancellation," in *International Conference on 5G for Ubiquitous Connectivity*, Levi, Finland, Nov 2014, pp. 110–115.

[72] D. Korpi, M. AghababaeeTafreshi, M. Piilila, L. Anttila, and M. Valkama, "Advanced architectures for self-interference cancellation in full-duplex radios: Algorithms and measurements," in *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov 2016, pp. 1553–1557.

[73] C. Safarian, T. Ogunfunmi, W. J. Kozacky, and B. K. Mohanty, "FPGA implementation of LMS-based FIR adaptive filter for real time digital signal processing applications," in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, Singapore, Singapore, July 2015, pp. 1251–1255.

[74] J. Lorente, M. Ferrer, M. de Diego, and A. Gonzalez, "GPU based implementation of multichannel adaptive room equalization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 7535–7539.

[75] A. Ghazi, J. Boutellier, L. Anttila, M. Juntti, and M. Valkama, "Data-parallel implementation of reconfigurable digital predistortion on a mobile GPU," in *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov 2015, pp. 186–191.

[76] K. Li, A. Ghazi, J. Boutellier, M. Abdelaziz, L. Anttila, M. Juntti, M. Valkama, and J. R. Cavallaro, "Mobile GPU accelerated digital predistortion on a software-defined mobile transmitter," in *IEEE Global Conference on Signal and Information Processing*, Orlando, FL, Dec 2015, pp. 756–760.

[77] K. Li, A. Ghazi, C. Tarver, J. Boutellier, M. Abdelaziz, L. Anttila, M. Juntti, M. Valkama, and J. R. Cavallaro, "Parallel digital predistortion design on mobile GPU and embedded multicore CPU for mobile transmitters," *Journal of Signal Processing Systems*, vol. 89, no. 3, pp. 417–430, Dec 2017.

[78] D. Korpi, J. Tamminen, M. Turunen, T. Huusari, Y. S. Choi, L. Anttila, S. Talwar, and M. Valkama, "Full-duplex mobile device: pushing the limits," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 80–87, September 2016.

[79] 3rd Generation Partnership Project, *Technical Specification Group Radio Access Network; Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced) (Release 14)*, March 2017, last accessed 19.08.2017. [Online]. Available: http://www.3gpp.org/ftp//Specs/archive/36_series/36.913/36913-e00.zip

[80] "CMOS power consumption and C_{pd} calculation," Texas Instruments, 1997, last accessed 08.04.2017. [Online]. Available: http://www.ti.com/lit/an/scaa035b/scaa035b.pdf

[81] A. R. Jensen, M. Lauridsen, P. Mogensen, T. B. Sørensen, and P. Jensen, "LTE UE power consumption model: For system level energy and performance optimization," in *IEEE Vehicular Technology Conference (VTC Fall)*, Quebec City, QC, Canada, Sept 2012, pp. 1–5.

Publications

PUBLICATION 1

M. Aghababaeetafreshi, L. Lehtonen, M. Soleimani, M. Valkama and J. Takala, "IEEE 802.11AC MIMO transmitter baseband processing on customized VLIW processor," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, May 4-9, 2014, pp. 7500-7504, DOI: 10.1109/ICASSP.2014.6855058

IEEE 802.11AC MIMO TRANSMITTER BASEBAND PROCESSING ON CUSTOMIZED VLIW PROCESSOR

Mona Aghababaeetafreshi[1], Lasse Lehtonen[2], Maliheh Soleimani[1], Mikko Valkama[1], and Jarmo Takala[2]

[1]Department of Electronics and Communications Engineering

[2]Department of Pervasive Computing

Tampere University of Technology, Korkeakoulunkatu 1, FI-33720 Tampere, Finland

Email: mona.aghababaeetafreshi@tut.fi

ABSTRACT

This paper presents a software-based implementation for the MIMO transmitter baseband processing conforming to the IEEE802.11ac standard on a DSP core with vector extensions. The transmitter is implemented in four different transmission scenarios, which include 2×2 and 4×4 MIMO configurations, yielding beyond 1Gbps transmit bit rate. The implementation is done for the frequency-domain processing and real-time operation has been achieved when running at a clock frequency of 500MHz. The proposed software solution is evaluated in terms of power consumption, number of clock cycles and memory usage. This SDR based implementation provides improved flexibility and reduced design effort compared to conventional approaches while maintaining energy consumption close to fixed-function hardware solutions.

Index Terms— OFDM, MIMO, WLAN, Software Defined Radio, Parallel Processing

1. INTRODUCTION

Due to the rapid growth and popularity of wireless handheld devices with efficient support for rich multimedia functionalities and broadband Internet access, both mobile cellular radio networks and Wireless Local Area Networks (WLAN) are evolving rapidly. While broadband wireless access is typically the driving priority, security, low power, low cost and reliability are also seen as very important aspects. Considering in particular the wireless connectivity in indoor environments, WLAN/WiFi solutions with optimized local area access for physical (PHY) and medium access control (MAC) layers are of increasing interest. This is also the main focus area of this article.

Currently, the clear majority of wireless local area connectivity is provided by IEEE WLAN/WiFi solutions whose flag-ship technology is IEEE 802.11ac [1]. In this standard, the throughput enhancements compared to legacy systems are obtained mainly through the deployment of advanced PHY layer innovations such as considerably wider transmission bandwidth through carrier aggregation, improved modulation and coding schemes and advanced deployment of multiantenna/MIMO transmission schemes. The standard utilizes transmission bandwidths up to 80MHz (mandatory) and 160MHz (optional), which is substantial improvement compared to 802.11n legacy system. Moreover, the flexibility of RF spectrum use is improved through allowing non-contiguous carrier aggregation where the total RF bandwidth can be composed of non-contiguous channels. Furthermore, multiantenna support up to 8×8 MIMO with eight spatial streams is specified, including also multiuser MIMO. The IEEE 802.11ac amendment also allows modulation orders up to 256QAM to further increase the highest achievable throughput. Overall, the instantaneous peak throughputs can reach 1Gbps [2].

In the existing literature, a clear majority of local area connectivity device implementations, in particular 802.11ac related, are fixed-function hardware based solutions. In [3], a VLSI implementation of a 4×4 MIMO-OFDM transceiver with 80MHz transmission bandwidth is described, and tailored to a single transmission scenario. In recent reports, some contributions have also been made towards the software defined radio concept. Design and implementation of the IEEE802.11 MAC layer processing on general-purpose DSP and additional accelerator systems is reported in [4]. In [5], a software defined radio implementation of 802.11 MAC with emphasis on cross-layer communications and networking is proposed and evaluated. However, as it can be seen also in [6]-[9], only selected parts of PHY or MAC layer are typically targeted while other processing still relies on dedicated hardware.

In this paper, we address the feasibility of software based implementation using VLIW processor for the real-time operation of IEEE802.11ac transmitter full PHY layer baseband processing in four different transmission scenarios which include 2×2 and 4×4 MIMO configurations. As the processing platform, stemming from the requirements for very fast processing of huge amounts of data with transmission bit rates in the order of 1Gbps, the customized VLIW processor with vector processing capabilities is used. Such a software based implementation, if found feasible, can offer more flexibility, much faster time-to-market, and highly improved possibilities to bringing in new transmission features and enhancements.

The rest of the article is organized as follows. In Section II, a detailed description of the selected transmission scenarios of 802.11ac standard is given. Then, in Section III, the employed processor and some of its main features are described. Furthermore, the software development environment and some of

This work was supported by the Finnish Funding Agency for Technology and Innovation (Tekes) under the Parallel Acceleration (ParallaX) project, and Tampere University of Technology graduate school.

the employed optimization approaches are introduced. The implementation results and analysis are then provided in Section IV. Finally, the conclusions are drawn in Section V.

2. TRANSMISSION SCENARIOS

In this work, we mainly focus on the PHY layer implementation of the IEEE802.11ac standard compatible multi antenna transmitter with selected transmission modes. According to the 802.11ac standard draft version, the VHT PHY comprises of two functional entities: the PHY function and the physical layer management functions. The PHY is consisting of PHY header part and data part; the header part itself is further divided into multiple fields where L-STF, L-LTF and L-SIG are the legacy portions and VHT-SIG-A, VHT-SIG-B, VHT-STF and VHT-LTF are the very high throughput fields. For more details, refer to [1].

In order to obtain very high throughput, IEEE 802.11ac defines various core functionalities and parameters, which increase the data rate considerably. Modulation and Coding Scheme (MCS) improvements and spatial multiplexing based MIMO transmission allow VHT performance achievement. Furthermore, other solutions such as wider bandwidth, shorter GI, and higher number of spatial streams are also introduced in the amendment. Additionally, the PHY implementation plays an important role in the VHT scenario, for instance the optional usage of Low-Density Parity-Check (LDPC) encoder and Space Time Block Coding (STBC) enhance the error protection and diversity characteristics. As a result, the performance characteristics are improved compared to legacy systems; thus helping to achieve VHT targets.

Fig. 1 depicts the main structure of the implemented data part processing at the transmitter side where depending on the transmission scenario some blocks may be obsolete. The minimum time for IFFT and frequency domain processing for a single OFDM symbol is 4µs for the header part and 3.6µs for the data part when short Guard Interval (GI) is used.

In our work, we cover the implementation of four operation points (transmission modes) of the IEEE 802.11ac standard. These four operation points have some of the implementation parameters in common such as channel bandwidth and modulation scheme. The channel bandwidth is set to 80MHz, which implies 256 subcarriers (234data+14null+8pilot subcarriers). Also in all cases, 256-QAM modulation scheme is employed to map a block of eight bits into one constellation point. In this work, the operational blocks from the stream parser to IFFT are implemented in all four scenarios. It should be noted that we have assumed all the incoming bits from the LDPC encoder are already stored in the local memory, hence the required time for data transfer to the local memory is not considered. The implementation procedure of the transmitter blocks in each four scenarios will be shortly discussed in the following.

2.1. Case A: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding, Short GI, 2×2 SU-MIMO

This case uses a 2×2 antenna configuration and two spatial streams, which are directly mapped to the space time streams hence removing the need for STBC coding. As defined in the IEEE 802.11ac standard, the bits received from the LDPC encoder should be directly fed to the stream parser block to be rearranged and parsed [1]. However, as tone mapping can be done more efficiently at bit-level rather than with complex numbers, in this work, this operation is placed before the stream parser and modulation blocks. Furthermore, the bits from the channel encoder are first fed to a preparation block to be rearranged for faster tone mapping and modulation. Preparation block combines the real and imaginary parts of each subcarrier, in such a way that the first outcoming 16-bit block has the real parts of the two streams, and the second 16-bit block has the imaginary parts (8 bits in each 16-bit parts are zeros). Then, the prepared streams are fed to the LDPC tone mapper, which shuffles the data subcarriers of both streams simultaneously. Next is the stream parser block after which each stream will have 234×8 bits in the following form; from left to right, the first and second 4-bit blocks present the real part of the first and second streams (first subcarrier). Respectively the third and fourth 4-bit blocks show the imaginary parts (first subcarrier) of the first and second streams and so on.

In order to obtain the most efficient performance, the stream parser and constellation mapper are merged into one function; thus the LDPC tone mapped complex numbers are parsed and mapped into the constellation points in the same function. Basically, the stream parser parses 234×8×2 coded bits per symbol into two spatial streams, i.e., each stream has 234×8 coded bits per symbol. Afterwards, a block of eight bits is mapped into one 256-QAM constellation point.

As rest of the operations, namely pilot insertion, cyclic shift diversity, spatial mapping and phase rotation are based on multiplication between the data subcarriers and coefficients, all these operations can be done at the same time. All of the possible combinations of the mentioned operations are pre-calculated and stored in a look-up table so that the operations can be executed with a single multiplication per subcarrier.

2.2. Case B: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding, Short GI, 4×4 SU-MIMO

This transmission mode has four spatial streams to be mapped into a 4×4 antenna configuration and thus due to the equal number of space time and spatial streams, STBC is obsolete. As mentioned in the previous subsection, the incoming bit streams should be rearranged for faster implementation. But in this case, as four spatial streams are used, the first incoming 16 bits already feature the real part of the first subcarrier of each stream, and the second 16 bits are the imaginary part. Therefore, the bits are arranged in the desired 16-bit format. Thus, no preparation is needed and the bit streams can be directly fed into the LDPC tone mapper.

After shuffling the data bits, the stream parser rearranges the bits and allocates them to the four streams and then every 8 bit block is mapped into one constellation point in the same function. Pilot insertion, cyclic shift diversity, spatial mapping, and phase rotation, similar to the previous case, are performed by multiplication using the look up table values as coefficients.

2.3. Case C: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding and STBC, Short GI, 2×2 antenna configuration with 1×1 SU-SISO transmission

As the title indicates, in this case, the number of the spatial streams is less than the number of space time streams which means STBC implementation is needed in addition to the other blocks employed in the previous cases. As defined in the IEEE 802.11ac standard, producing the even numbered space time streams includes conjugation and negation of the symbols in the odd numbered space time streams. Since conjugation is basically negating the imaginary part of a complex number, this operation can be easily done at bit level by simply inverting the sign bit. Therefore, STBC block is also moved to the preparation block in this work.

In the next phase, the STBC encoded bits are fed to the LDPC tone mapper to be shuffled. Then the stream parsing and constellation mapping are applied to both space time streams, and finally in the last stage, pilot insertion, cyclic shift, spatial mapping and phase rotation are done at once.

2.4. Case D: 80MHz TX/RX bandwidth, 256QAM with 3/4 coding and STBC, Short GI, 4×4 antenna configuration with 2×2 SU-MIMO transmission

In the last transmission mode, there are two spatial streams and a 4×4 antenna configuration, which means STBC shall be applied. Similar to the previous scenario, the STBC creates four space time streams from two spatial streams in the preparation block. Afterwards, the bit streams will be LDPC tone mapped in the next block and then go through stream parser/constellation mapper block simultaneously. As described in the previous scenarios, the final block performs pilot insertion, cyclic shift, spatial mapping and phase rotation.

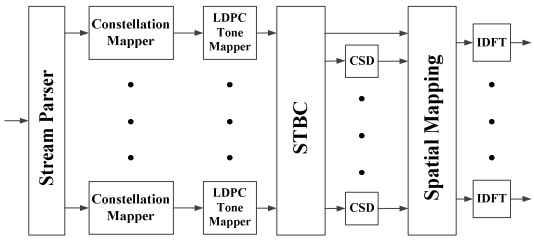


Fig. 1. Principal block diagram of transmitter baseband processing.

3. VLIW ARCHITECTURE AND IMPLEMENTATION

Due to huge amount of information processed, a customized VLIW processor with vector extensions is used as the platform to implement the transmitter baseband processing functions in this work. The adopted DSP core, Tensilica ConnX BBE32 [10], is a high performance, very small size and ultra-low power consumption DSP core that has been specifically designed for use in the cost and power sensitive baseband modem systems [10]. This DSP core is a 4-issue VLIW processor and has support for vector operations with the aid of a 16-way SIMD ALU engine and 32-way MAC SIMD engine. In addition, the processors can access wide data chunks from memory in blocks of 256 bits. Fig. 2 illustrates the general architecture of the ConnX BBE32 core.

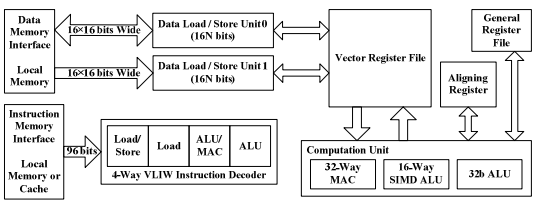


Fig. 2. ConnX BBE32 architecture.

Additionally, this DSP engine is equipped with dedicated hardware accelerator blocks to off-load computationally intensive operations such as FFT/IFFT [11]. The processor is configurable and special function units can be added to speed up the computations. We used two different processor configurations of the ones provided by the vendor: Low-Power (LP) and Performance-Maximize (PM) configurations. The LP is the baseline configuration and the PM configuration provides instruction extensions for accelerating various functions, e.g., FFT, FIR filtering, bit mapping etc. The processor has Harvard architecture with one instruction memory and two data memories.

Tensilica uses an Eclipse-based software development environment (Xtensa Xplorer), which provides a comprehensive collection of code generation and analysis tools. This tool enables software development to be carried out using C programming. However, for optimization purposes, nearly all of the software implementation in this work is done by heavily using the provided processor intrinsics. In spite of the automatic vectorization capability of the compiler, the code was vectorized manually for better performance.

As mentioned earlier, correct configuration and programming play an important role in the efficiency of the implementation, thus some optimization approaches have been applied in this work to fasten the processing. One very effective optimization approach was merging and combining the functions as much as possible. Specifically the operation blocks whose functionality involves multiplication with a constant (such as phase rotation, spatial mapping, pilot insertion and cyclic shift diversity) can be easily merged. Moreover, since it is easier to deal with bits rather than complex numbers, as many operations as possible have been implemented before the constellation mapping. For instance, although in the standard and as shown in Fig. 1, the STBC and LDPC tone mapper blocks are defined to be employed after the constellation mapping, it has been observed that such operations can be implemented more efficiently when the data is still in bits and not yet modulated to symbols.

4. RESULTS AND ANALYSIS

The described software based implementation was profiled and analyzed with the aid of the tools provided by the vendor. The results related to number of clock cycles, power, and memory usage are presented in this section.

The numbers of clock cycles were obtained with the instruction set simulator and profiling tools. In Fig. 3, the numbers of clock cycles needed to process one OFDM symbol in LP configuration are presented. PM configuration requires larger number of cycles in comparison with the LP model but the difference is only 1-2 %.

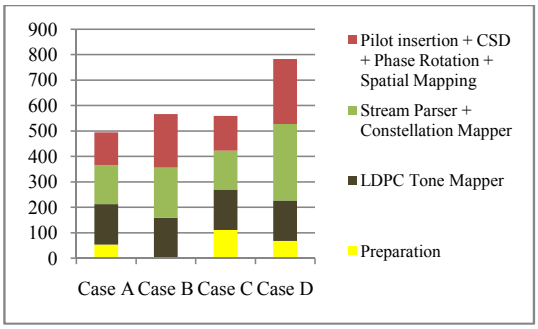


Fig. 3. LP model clock cycles results for all the cases.

As mentioned earlier, in different transmission scenarios, different blocks may operate; therefore the results are given for each block in each transmission scenario. It should be noted that for the Cases C) and D), the preparation block also includes the STBC coding operation.

As previously mentioned, the duration of an OFDM symbol is 4µs for the header part and 3.6µs for the data part when short Guard Interval (GI) is used. Thus to achieve real-time operation in the transmitter, all the processing needed to create one OFDM symbol should not take more than 3.6µs. Assuming a 500 MHz operating frequency, 3.6µs can accommodate 1800 clock cycles. Looking at the total number of clock cycles for each transmission scenario from Fig. 3, it can be concluded that the system operations can be computed in real-time in this implementation.

One of the most important evaluation criteria for the implementation is the power consumption, which is directly dependent on the memory configuration/capacity. As the vendor provides Energy Xplorer for energy consumption estimation, first energy usage is profiled and then power consumption is calculated by dividing the energy values by time. The time for each block is defined by the number of clock cycles. We have considered two common cases, which are the maximum (128k) and half (64k) of the memory capacity. The energy consumption was estimated by exploiting technology libraries for a 40nm low-power IC technology provided by the tool vendor. We also assume clock frequency of 500MHz. The monitoring time for the energy analysis is 3.6µs, and it includes both leakage and dynamic parts. Fig. 4 reveals the results related to the power consumption for the LP model, in case of full and half memory usage. The power consumption results for the PM model are not presented as there is no significant difference with the LP model. In general, the power consumption estimates are found feasible to mobile terminal scale devices.

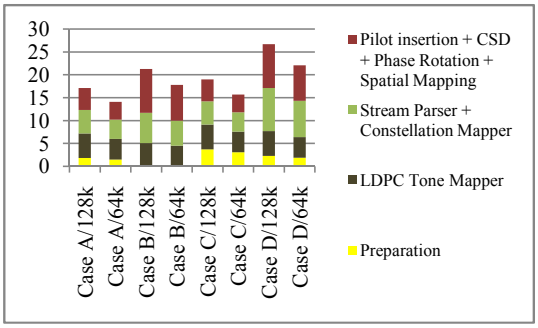


Fig. 4. Power consumption in mW for 128k and 64k memory capacities.

In order to avoid stalls and keep the pipeline full, loop unrolling was heavily exploited in some of the functional blocks such as LDPC tone mapper and the block including pilot insertion, spatial mapping, CSD and phase rotation to exploit parallelism among the instruction. As loop unrolling increases the program code size, to evaluate how much memory is needed for the developed software, instruction memory usage for each operation in each transmission scenario was measured and is presented in Table 1. Since there is no difference between PM and LP model from the memory usage point of view, only the results when using the PM model are presented. In order to get more informative realization on the effect of loop unrolling on the program code size, the code density was also calculated for all the functional blocks. The average code density over all blocks is 52.95%.

In addition to the instruction memory usage, data memory usage was evaluated and is presented in Table 1. It should be noted that the amount of used data memory does not depend on the transmission scenario except for the input buffer usage.

Table 1. Memory usage in bytes

	Case A	Case B	Case C	Case D
Instruction memory				
Preparation	184	---	376	240
LDPC Tone Mapper	1808	1808	1808	1808
Stream Parser + Constellation Mapper	432	200	428	528
Pilot insertion + CSD + Phase Rotation + Spatial Mapping	720	720	720	720
Total	3612	3664	3800	4232
Data Memory				
Local Data RAM #1	4.8 K			
Local Data RAM #2	5.128 K			
Input Buffer	468	936	468	936
Total	10.396K	10.864K	10.396K	10.864K

To achieve higher performance and faster processing, the numerical values of the cyclic shift diversity for different streams, spatial mapping and phase rotation operations were calculated and stored in a look-up table. This look-up table takes 128 and 800 bytes from the local data RAM #1 and local data RAM #2 memories, respectively.

4. CONCLUSIONS

In this paper, we developed software-based implementation of the IEEE 802.11ac transmitter full frequency-domain PHY layer baseband processing for four different multi-antenna transmission scenarios. We have evaluated the solution by profiling and analyzing the implementation using the tools provided by the vendor. We have presented the results with regards to number of clock cycles, power consumption, and memory usage. The analysis of the performance numbers clearly shows that the developed software based implementation on a DSP core can achieve real-time operation for the transmitter baseband processing assuming 500 MHz clock frequency. Furthermore, the implementation resulted in realistic power consumption and memory usage, despite of massive amount of data processing yielding beyond 1Gbps transmission bit rate in the most ambitious transmission scenario. The future work will focus on implementing the corresponding receiver chain PHY processing, which includes more complex functions such as channel state estimation and detection.

5. REFERENCES

[1] IEEE P802.11ac™ Draft Standard, version 5, January 2013.

[2] E. Perahia and R. Stacey, *Next Generation Wireless LANs*, Cambridge, NY, 2013.

[3] S. Yoshizawa and Y. Miyanaga, "VLSI Implementation of a 4×4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," in *Proc. IEEE ISCAS*, Taipei, Taiwan, 24-27 May 2009, pp. 1743-1746.

[4] S. Samadi, A. Golomohammadi, A. Jannesari, M.R. Movahedi, B. Khalaj, and S. Ghammanghami, "A Novel

Implementation of the IEEE802.11 Medium Access Control," in *Proc. Int. Symp. Intelligent Signal Process. Commun.*, Yonago, Japan, 12-15 Dec. 2006, pp.489-492.

[5] J.R. Gutierrez-Agullo, B. Coll-Perales, and J. Gozalvez, "An IEEE 802.11 MAC Software Defined Radio implementation for experimental wireless communications and networking research," in *Proc. IFIP Wireless Days*, Venice, Italy, 20-22 Oct. 2010, pp.1-5.

[6] K. Rounioja, and K. Puusaari, "Implementation of an HSDPA Receiver with a Customized Vector Processor," in *Proc. Int. Symp. System-on-Chip*, Tampere, Finland, 13-16 Nov. 2006, pp.1-4.

[7] W. Xu, M. Richter, M. Sauermann, F. Capar, and C. Grassmann,, "Efficient baseband implementation on an SDR platform," in *Proc. Int. Conf. ITS Telecommunications*, St. Petersburg , Russia, 23-25 Aug. 2011, pp.794,799.

[8] J. Janhunen, T. Pitkänen, M. Juntti, and O. Silvén, "Energy-efficient programmable processor implementation of LTE compliant MIMO-OFDM detector," in *Proc. IEEE ICASSP*, Kyoto, Japan, 25-30 March 2012, 3276 – 3279.

[9] S. Eberli, A. Burg, and W. Fichtner, "Implementation of a 2×2 MIMO-OFDM receiver on an application specific processor," *Microelectronics Journal*, vol. 40, no. 11, pp. 1642-1649, November 2009.

[10] Tensilica Inc., *ConnX BBE32 DSP User Guide*, USA, 2012.

[11] Tensilica Inc., *ConnX BBE32 DSP Core for Baseband Processing,* USA, 2013.

PUBLICATION 2

M. Aghababaee Tafreshi, L. Lehtonen, T. Levanen, M. Valkama and J. Takala, "IEEE 802.11ac MIMO receiver baseband processing on customized VLIW processor," in *IEEE Workshop on Signal Processing Systems,* Belfast, UK, Oct. 22-24, 2014, pp. 1-6, DOI: 10.1109/SiPS.2014.6986092

IEEE 802.11ac MIMO Receiver Baseband Processing on Customized VLIW Processor

Mona AghababaeeTafreshi, Lasse Lehtonen, Toni Levanen, Mikko Valkama, and Jarmo Takala
Tampere University of Technology, Tampere, Finland
mona.aghababaeetafreshi@tut.fi

***Abstract*— In this paper, a software-based implementation for the Multiple Input and Multiple Output (MIMO) receiver baseband processing conforming to the IEEE 802.11ac standard on a DSP core with vector extensions is presented. The implementation is carried out for different operation points including 2×2 and 4×4 MIMO configurations, yielding beyond 1Gbps transmission bit rate. This implementation mainly focuses on the frequency domain processing of the receiver. The presented solution is evaluated in terms of number of clock cycles and power consumption and the feasibility of a real-time operation is then addressed and analyzed. If found feasible, such Software Defined Radio based solutions offer more flexibility and reduced time-to-market-cycles compared to the conventional solutions using fixed-function hardware platforms.**

Keywords—OFDM; MIMO; WLAN; Software Defined Radio; parallel processing

I. INTRODUCTION

As wireless standards continue to evolve rapidly, the need for adaptable devices supporting different air interfaces grows. Currently, most wireless devices are implemented based on application specific fixed-function hardware platforms where most of the physical (PHY) and Medium Access Control (MAC) layer processing is still done using dedicated hardware, most notably Application-Specific Integrated Circuits (ASIC) [1]. Being implemented in silicon, such devices can offer only limited programmability and flexibility. Furthermore, adding more support for different specifications in these devices requires a larger die size and will consequently result in more power hungry devices. On the contrary, a software-based solution can offer high flexibility by employing programmable and reconfigurable platforms. In addition to the lack of flexibility of ASIC implementations, the complexity and parameterization of future systems is so high that HW optimization is extremely difficult and error-prone. Using a software defined radio platform, on the other hand, the functionality can be changed by modifying the software while still maintaining good energy-efficiency compared to fixed-function hardware implementations. Such software-based implementations will enable fast scalability at the radio layer, to improve the efficiency and flexibility of RF spectrum use. Having less costs and design efforts during development, testing, and maintenance, such solutions will also clearly reduce the time-to-market cycle [1].

The extraordinary growth in number of applications with high bandwidth requirements such as video streaming along with the increasing number of users has created an evolutionary demand to enhance the capacity of wireless networks. As a result, both mobile cellular radio networks and Wireless Local Area Networks (WLAN) are evolving rapidly to meet the high demands. Considering in particular the wireless connectivity in indoor environments, the IEEE WLAN family provides one important technology component, in parallel to cellular mobile radio evolution. The emerging flagship amendment to IEEE 802.11^{TM} WLAN standard with beyond 1Gbps bit rates is the IEEE 802.11ac [2].

This new amendment to IEEE 802.11™ WLAN standard is intended to meet the evolving needs for higher transmission data rates in range of gigabits per second and to help enable new generations of data-intensive wireless applications. The IEEE 802.11ac enables multi-gigabit data throughput at 5 GHz band [2]. The IEEE 802.11ac specification adds support for 80MHz and 160MHz channel bandwidths. The 160MHz channel may be contiguous or non-contiguous, where the non-contiguous allocation provides more flexible channel assignment. Additionally, it adds higher order modulation in the form of 256 Quadrature Amplitude Modulation (QAM) which results in improved peak data rate [2]. Furthermore, by advanced deployment of multi-antenna techniques, a further increase in data rates is achieved. The Very High Throughput (VHT) physical (PHY) layer defined in [2] allows increasing the number of spatial streams up to eight streams [2]. This amendment also introduces a new technique to allow multiple users to be served simultaneously on downlink. This technique is referred to as Multi-User (MU) MIMO. MU-MIMO enables higher system capacity, more efficient spectrum use, and reduced latency [2].

The IEEE 802.11ac physical layer packet consists of PHY header part and data part. The PHY header part is divided into multiple fields where Non-HT Short Training Field (L-STF), Non-HT Long Training Field (L-LTF), Non-HT SIGNAL Field (L-SIG) are legacy portion and VHT Signal A field (VHT-SIG-A), VHT Short Training Field (VHT-STF), VHT Long Training Field (VHT-LTF) and VHT Signal B field (VHT-SIG-B) are the VHT specific fields. Fig. 1 illustrates the VHT physical layer data packet structure and the challenging timing requirements, assuming that the short guard interval (GI) is used for data symbols.

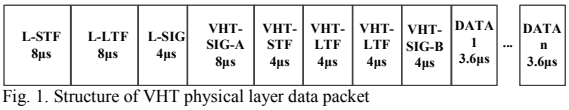


Fig. 1. Structure of VHT physical layer data packet

The majority of the previous works carried out regarding the implementation of wireless connectivity devices have focused on fixed function hardware based implementations. An example can be found in [3], where a VLSI implementation for a 4x4 MIMO-OFDM transceiver is described which is fixed to a single operating point using 80MHz transmission bandwidth. In [4], implementation of the complete baseband processing of

This work was supported by the Finnish Funding Agency for Technology and Innovation (TEKES) under the Parallel Acceleration (ParallaX) project, and Tampere University of Technology graduate school.

an IEEE 802.11a receiver on an application specific processor is described. Some contributions have been also made towards software-based solutions. However, in these works typically only parts of the PHY or MAC layer processing have been addressed. In [5], a software-defined FFT/IFFT architecture for IEEE 802.11ac is proposed based on customized soft stream processor on Field-Programmable Gate Array (FPGA). In [6], a fully programmable Software Defined Radio implementation of the IEEE 802.11 MAC that can be fully modified to develop advanced cross-layer communications and networking techniques, is presented.

In this paper, we address the feasibility of achieving a real-time operation for the IEEE 802.11ac receiver PHY layer baseband processing using a software-based implementation on a customized Very Long Instruction Word (VLIW) processor. The implementation is carried out for different transmission scenarios including 2×2 and 4×4 MIMO antenna configurations. The implemented scenarios can reach data bit rates in the order of 1Gbps. Originating from the requirements for fast processing of large amounts of data for such high data rates, a customized VLIW processor with vector processing capabilities is selected as the implementation platform. The work presented in this paper is the continuation of the transmitter implementation of the IEEE 802.11ac PHY layer baseband processing using the same platform presented in [7].

The rest of the article is organized as follows. In Section II, the implemented receiver functionalities and employed algorithms are introduced. Then, in Section III, a short description of the different implemented scenarios of the IEEE 802.11ac standard is given. In Section IV, the implementation platform and the used architecture are described. In Section V, the implementation results are presented in terms of number of clock cycles and power consumption. Finally, in Section VI, the conclusions are drawn.

II. Receiver Processing

In this section, a brief overview of the implemented algorithms for the different functional blocks in the receiver is given.

A. SINR Estimation

To improve the link quality and performance, the SINR estimation needs to be done in the receiver. SINR estimation can be used to optimize the transmit power level and dynamically adapting the data rate. In the current implementation, we calculate the Received Channel Power Indicator (RCPI), Average Noise Power Indicator (ANPI), and the Received Signal to Noise Indicator (RSNI). The RSNI value is reported for the transmitting entity for Modulation and Coding Scheme (MCS) adaptation. The averaging of measured values is done for better stability of the system. The averaged measurements should be obtained closely in time for high correlation.

1) RCPI measurement: RCPI is calculated as the average power over all received R_x, x=1,2, .., N_{RX}, receiver antennas. When receiving a Null Data Packet (NDP) for RSNI update, we calculate the RCPI over VHT-LTF symbols and VHT-SIG-B symbol. If RCPI is updated over a data packet, we calculate the RCPI over DATA symbols. The RCPI is evaluated as the average power over all non-pilot active subcarriers. RCPI updated over a DATA packet can be written as:

$$RCPI = \frac{1}{N_s \times N_{R_x} \times N_D} \times \sum_{R_x} \sum_{t} \sum_{i} Y_{DATA,t,R_x}(i) \qquad (1)$$

where N_s is the number of non-pilot active subcarriers, N_D is the number of data symbols, $i \in I_{\text{non-pilot, active subcarriers}}$, and t, t=1,2,.., N_t, is the symbol index. RCPI can be also averaged over several data packets inside a desired time window to further improve the reliability.

2) ANPI measurement: In the standard [8], the ANPI measurement is defined to be done during idle periods. However, as we use this value for the symbol detection, we estimate ANPI in the receiver based on the average power in the null carriers, except DC, in the STF symbols. L-STF and VHT-STF symbols are suitable for noise estimation, because they contain several zeros in the frequency domain presentation in addition to non-active carriers. Thus any changes in the subcarriers containing zeros can be considered noise. We assume that the time and frequency synchronization accuracy while detecting STF symbols is sufficient for us to measure only noise power in the zero-valued carriers. This can be done also as a post processing step, after properly synchronizing to the received signal. ANPI can be written as:

$$ANPI_{N_{Rx}} = \frac{1}{N_s \times N_{R_x}} \times \sum_{R_x} \sum_{i} Y_{L-STF/VHT-STF,R_x}(i) \qquad (2)$$

where N_s is the number of active zero-valued pilot subcarriers and $i \in I_{\text{active,zero-valued pilots}}$.

3) RSNI Measurement: Having calculated the values for RCPI and ANPI, the RSNI can be calculated according to the following:

$$RSNI = 10 \times \log_{10}\left(\left(RCPI - ANPI\right) / ANPI\right) \qquad (3)$$

In the above formula, the RCPI and ANPI are the power values in linear scale.

B. Channel Estimation

For detecting a WLAN 802.11ac packet, the receiver has to calculate two different channel estimates, one for the non-precoded (non-VHT) part and one for the possibly precoded part (VHT-part). First channel estimate is obtained from L-LTF symbols for detecting L-SIG and VHT-SIG-A fields. Second channel estimate is obtained, after detecting VHT-SIG-A, from VHT-LTF fields.

1) Channel estimator for the legacy part: After time and frequency synchronization, Cyclic Prefix (CP) removal and FFT operation, the received signal for L-LTF symbol per symbol index t, t = [1, 2], per subcarrier index k, $k \in I_{\text{active,non-pilot L-LTF subcarriers}}$, can be written as (4) where $\mathbf{H}_k$ is a ($N_{Rx} \times N_{Tx}$) complex channel matrix, $\mathbf{H}_{eff,k}$ is the ($N_{Rx} \times 1$) effective sum channel for legacy part, $\mathbf{X}_{L\text{-}LTF,k}$ is an $(N_{Tx} \times 1)$ real vector containing only the training symbol $\mathbf{x}_{L\text{-}LTF,k}$ (ones and minus ones) and $\mathbf{N}_{t,k}$ is an $(N_{Rx} \times 1)$ complex Gaussian noise vector.

$$\mathbf{Y}_{k,t} = \mathbf{H}_k \cdot \mathbf{X}_{L-LTF,K} + \mathbf{N}_{t,k} = \mathbf{x}_{L-LTF,k} \cdot \begin{bmatrix} \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} \mathbf{h}_{1,j} \\ \vdots \\ \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} \mathbf{h}_{N_{Rx},j} \end{bmatrix} + \mathbf{N}_k = \mathbf{x}_{L-LTF,k} \mathbf{H}_{eff,k} + \mathbf{N}_{t,k} \tag{4}$$

Now, given that we receive two L-LTF symbols in the preamble, the Least Squares (LS) channel estimator is given as:

$$\mathbf{H}_{eff,k,LS} = \frac{\mathbf{x}_{L-LTF,k}}{2} \sum_{t=1}^{2} \mathbf{Y}_{k,t} \tag{5}$$

where index t, t = [1, 2], indicates L-LTF symbol index over which the received signal is averaged before channel estimation. The channel estimate is obtained by directly multiplying with $\mathbf{x}_{\text{L-LTF},k}$ because it can only have values [-1, +1]. With $\mathbf{H}^{\text{H}}$ representing the Hermitian transpose of Matrix $\mathbf{H}$, the Linear Minimum Mean Square Error (LMMSE) estimator based on LS estimate can be written as:

$$\hat{\mathbf{H}}_{eff,k,LMMSE} = \hat{\mathbf{H}}_{eff,k,LS} \hat{\mathbf{H}}^H_{eff,k,LS} \cdot \left(\hat{\mathbf{H}}_{eff,k,LS} \hat{\mathbf{H}}^H_{eff,k,LS} + \frac{\sigma_n^2}{2} \mathbf{I}_{N_{Rx}} \right)^{-1} \hat{\mathbf{H}}_{eff,k,LS} \tag{6}$$

2) Channel estimator for VHT part: After detecting VHT-SIG-A, the receiver knows how many VHT-LTF symbols it should collect for VHT channel estimation. The VHT-LTF preamble differs from the legacy part in two main ways. First, the VHT-LTF subcarriers k, $k \in \mathrm{I}_{\text{active,non-pilot VHT-LTF subcarriers}}$, are precoded by VHT-LTF mapping matrix $\mathbf{P}$ (defined in [2]) of size ($N_{STS} \times N_{VHT\text{-}LTF}$). Secondly, the VHT-LTF symbols may be precoded by the precoder matrix $\mathbf{Q}_j$, $j \in \mathrm{I}_{\text{active,VHT-LTF subcarriers}}$. After synchronization, CP removal and FFT operation, the received signal for VHT-LTF symbol per symbol index t, t = [1,…,$N_{VHT\text{-}LTF}$], per subcarrier index k, $k \in \mathrm{I}_{\text{active, non-pilot VHT-LTF subcarriers}}$, can be written as:

$$\mathbf{Y}_{k,t} = \mathbf{H}_\mathbf{k} \mathbf{Q}_\mathbf{k} \mathbf{P}(:,t) \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_{k,t} = \mathbf{H}_{eff,k} \mathbf{P}(:,t) \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_{k,t} \tag{7}$$

Now, in the receiver, to get effective channel estimates per Space Time Stream (STS), the received VHT-LTF symbols are weighted with the rows of P matrix and averaged over all VHT-LTF symbols. For presentation clarity, let us stack the received samples per subcarrier k, over all Rx antennas and VHT-LTF symbols into single column vector, given as (8) where □ represents Kronecker tensor product.

$$\mathbf{Y}_k = \begin{bmatrix} \mathbf{Y}_{k,1} \\ \vdots \\ \mathbf{Y}_{k,N_{VHT-LTF}} \end{bmatrix} = \left(\mathbf{P} \otimes \mathbf{I}_{N_{Rx}} \right)^T \begin{bmatrix} \mathbf{h}_{eff,k}(:,1) \\ \vdots \\ \mathbf{h}_{eff,k}(:,N_{VHT-LTF}) \end{bmatrix} \mathbf{x}_{VHT-LTF,k} + \mathbf{N}_k \tag{8}$$

Now, the received training VHT-LTF training signal after decoding diversity coding is given as:

$$\tilde{\mathbf{Y}}_k = \frac{1}{N_{VHT-LTF}} \left(\mathbf{P} \otimes \mathbf{I}_{N_{Rx}} \right) \begin{bmatrix} \mathbf{Y}_{k,1} \\ \vdots \\ \mathbf{Y}_{k,N_{VHT-LTF}} \end{bmatrix} = \mathbf{x}_{VHT-LTF,k} \begin{bmatrix} \mathbf{h}_{eff,k}(:,1) \\ \vdots \\ \mathbf{h}_{eff,k}(:,N_{STS}) \end{bmatrix} + \mathbf{W}_k \tag{9}$$

where $\mathbf{W}_\mathrm{k} \in \mathrm{CN}\left(0, \frac{\sigma_\mathrm{n}^2}{\mathrm{N_{VHT\text{-}LTF}}}\right)$. Then, the LS channel estimate can be written as:

$$\hat{\tilde{\mathbf{H}}}_{eff,k,LS} = \mathbf{x}_{VHT\_LTF,k} \tilde{\mathbf{Y}}_k \tag{10}$$

Note that now the columns of the original channel matrix are stacked on top of each other. After obtaining the LS channel estimate, it is used to calculate the LMMSE channel estimation using (6) by replacing $\frac{\sigma_n^2}{2} \times \mathbf{I}_{N_{Rx}}$ with $\sigma_n^2 \mathbf{I}_{N_{Rx} N_{VHT-LTF}}$.

During the receiver implementation, LMMSE channel estimation proved to be one of the very time consuming operations as it involves the inversion of a 4×4 matrix according to (6). If (6) is written assuming a two antenna configuration, (11) is derived as:

$$\hat{\tilde{\mathbf{H}}}_{eff,k,LMMSE} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a^* & b^* & c^* & d^* \end{bmatrix} \times \left(\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a^* & b^* & c^* & d^* \end{bmatrix} + \begin{bmatrix} \sigma_n^2 I_{N_{Rx} N_{VHT-LTF}} & 0 & 0 & 0 \\ 0 & \sigma_n^2 I_{N_{Rx} N_{VHT-LTF}} & 0 & 0 \\ 0 & 0 & \sigma_n^2 I_{N_{Rx} N_{VHT-LTF}} & 0 \\ 0 & 0 & 0 & \sigma_n^2 I_{N_{Rx} N_{VHT-LTF}} \end{bmatrix} \right)^{-1} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{11}$$

where a, b, c, d are calculated according to (10), $\mathbf{H}_{eff,k,LS} = \begin{bmatrix} a & b & c & d \end{bmatrix}$, and the superscript * represents the complex conjugate of the corresponding element.

Now, using the Sherman-Morrison law [9], (11) can be simplified to:

$$\mathbf{H}_{eff,k,LMMSE} = \begin{bmatrix} \dfrac{a(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \dfrac{b(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \dfrac{c(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \\ \dfrac{d(aa^* + bb^* + cc^* + dd^*)}{(\sigma_n^2 + aa^* + bb^* + cc^* + dd^*)} \end{bmatrix} = \frac{\mathbf{H}_{LS} \times \mathbf{H}_{LS}^T \times conj(\mathbf{H}_{LS})}{\sigma_n^2 + \mathbf{H}_{LS}^T \times conj(\mathbf{H}_{LS})} \tag{12}$$

where conj(H) represents the element wise complex conjugate of matrix H.

Using this approach, we can avoid the complexity due to the matrix inversion in the channel estimation process. The same approach can be used in the 4×4 antenna configuration. Simplifying the computation of the matrix inversion reduced the complexity and the number of clock cycles to a great extent.

C. Pilot Based Fine Frequency Error Estimation and Correction

In order to compensate the effects of the frequency error on the received symbols, first the frequency error should be estimated. The frequency error is measured using the phase angle difference per symbol index t can be given as:

$$\hat{\Theta}_{t+1} = \frac{1}{N_p} \sum_{i \in \Omega P} \angle P_{i,t+1} - \angle P_{i,t} \tag{13}$$

where $i \in I_{\text{pilot, DATA subcarriers.}}$ Having the phase angle differences, the frequency error can be calculated as:

$$\hat{F}_{error,t+1} = \hat{F}_{error,t} + \frac{F_s}{N_s + N_{GI}} \hat{\Theta}_{t+1} \tag{14}$$

where F_s is the sampling frequency, N_s is the number of subcarriers, and N_{GI} is the number of samples in the Guard Interval (GI). Once the frequency error is calculated, the received DATA symbols can be corrected using:

$$\mathbf{y} = \mathbf{y}_{received} \times e^{-j\pi \hat{F}_{error,t}} \tag{15}$$

D. LDPC Tone Demapping

When Low Density Parity Check (LDPC) encoder is used as the Forward Error Correction (FEC) method, LDPC tone mapping should be employed in the transmitter, whereas in case of Binary Convolutional Codes (BCC), BCC interleaver shall be employed. LDPC tone mapper was introduced in 802.11ac to achieve full frequency diversity from 80MHz and 160MHz bandwidths. The LDPC tone mapper maps consecutive symbols to non-consecutive subcarriers inside one OFDM symbol. In other words, the LDPC tone mapper shuffles the data subcarriers in each OFDM symbol in each spatial stream. Thus in the receiver, the LDPC tone demapper rearranges the shuffled subcarriers into their original places.

E. Stream De-parser

Stream parsing is the operation done in the transmitter to re-arrange and divide the coded bits into N_{ss} spatial streams. The left-hand side in Fig. 3 illustrates how stream parsing is done for an unknown number of streams. In the receiver the N_{ss} streams are then de-parsed to form one bit stream as shown in the right-hand side of Fig. 2.

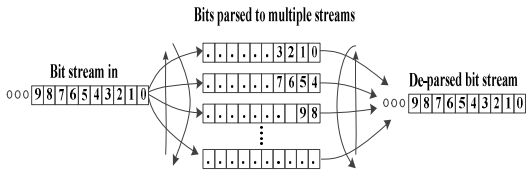


Fig. 2. Stream parsing and de-parsing process

F. Symbol Detection

To detect the symbols at the receiver, LMMSE detection is employed. The detector coefficients can be calculated using the LMMSE channel estimation derived in (6). The detector coefficients can be calculated as:

$$\mathbf{D}_{coeff} = \left(\mathbf{H}_{LMMSE}^H \mathbf{H}_{LMMSE} + \sigma_n^2 \mathbf{I}_{N_{STS}} \right)^{-1} \mathbf{H}_{LMMSE}^H \tag{16}$$

where the $\mathbf{H}_{LMMSE}^H$ is a ($N_{\text{Rx}} \times N_{\text{STS}}$) matrix. Once the coefficients are calculated, the received symbols can be detected using:

$$\hat{\mathbf{X}} = \mathbf{D}_{coeff,fixed} \mathbf{Y} \tag{17}$$

where $\mathbf{Y}$ is a ($N_{\text{Rx}} \times 1$) matrix containing the received symbols.

G. Soft Bit Detection

In soft bit detection, for each bit position, the difference of distances to the nearest zero and one bit on the constellation is calculated. This operation is illustrated in Fig. 3. This is a sub-optimal method for reducing the complexity of the soft bit detection implementation where instead of calculating the distance to all constellations, only the nearest ones are considered.

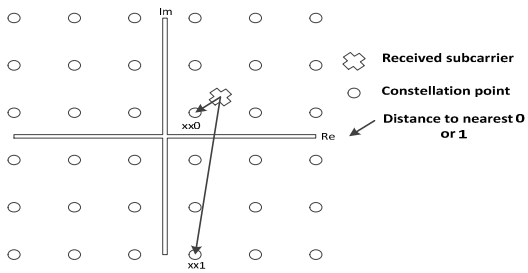


Fig. 3. Soft bit detection process

III. TRANSMISSION SCENARIOS

In this work, we have covered four different operation points (transmission scenarios) of the IEEE 802.11ac. The transmitter implementation for these cases was discussed in [7]. In all cases the channel bandwidth is set to 80MHz, which implies that each OFDM symbol contains 256 subcarriers including 234 data, 14 null, and 8 pilot subcarriers. Furthermore, in all cases 256QAM is selected as the modulation scheme mapping a block of 8 coded bits into one constellation point. In this implementation short GI is used implying that the duration of each OFDM DATA symbol is equal to 3.6μs.

Fig. 4 and Fig. 5 depict main structure of the implemented processing at the receiver. Some blocks may be obsolete in some cases depending on the scenario. It is also assumed that the incoming symbols are stored in a local memory and consequently the time required for the transfer of the data to the local memory is not considered.

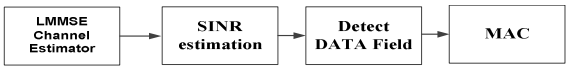


Fig. 4. Overall block diagram of frequency domain receiver processing

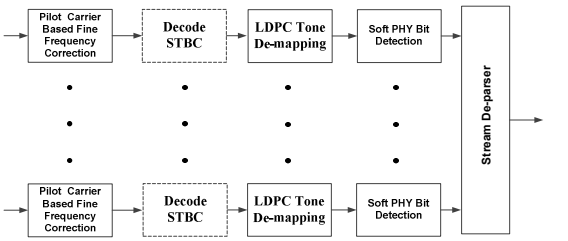


Fig. 5. Block diagram of the receiver DATA field baseband processing

Table I briefly describes the different transmission scenarios implemented in this work and highlights the common parameters and differences in these four operation points.

TABLE I. THE SPECIFICATIONS FOR THE IMPLEMENTED TRANSMISSION SCENARIOS

	Common Parameters	Number of antennas	Number of spatial streams	STBC coding
CASE A	80MHz TX/RX bandwidth, 256QAM, ¾ coding rate, Short GI	2	2	No
CASE B		4	4	No
CASE C		2	1	Yes
CASE D		4	2	Yes

IV. ARCHITECTURE AND IMPLEMENTATION

The implementation platform used in this work was selected by taking into consideration the requirement for fast processing of huge amounts of data, imposed by the IEEE 802.11ac support for very high data rates in the order of gigabits per second. As a result a VLIW processor with vector processing capabilities is chosen. More specifically, we have selected the Tensilica ConnX BBE32 DSP core as our processing platform in this work. This DSP core, which is specifically designed to be used in the next generation communication systems, is based on a high performance, ultra-low power, and very small size architecture [10]. The ConnX BBE32 meets the high computational requirements by supporting vector operations using a 16-way SIMD ALU and a 4-issue VLIW processing pipeline. Additionally, this core is equipped with 32 multiply-accumulate units and can access wide data chunks in blocks of 256 bits from the memory. The ConnX BBE32 block diagram can be found in Fig. 6. This DSP core uses a Harvard architecture having two data memories and one instruction memory. Moreover, to help offload the computationally intensive operations such as FFT/IFFT, the dedicated hardware accelerator blocks can be used, which are then controlled with custom instruction extensions. Tensilica uses an Eclipse based software development environment named Xtensa Xplorer, which provides a complete set of tools for code generation and profiling. Programming in C language is possible in this environment. However, we have manually optimized our code with the aid of the compiler intrinsics.

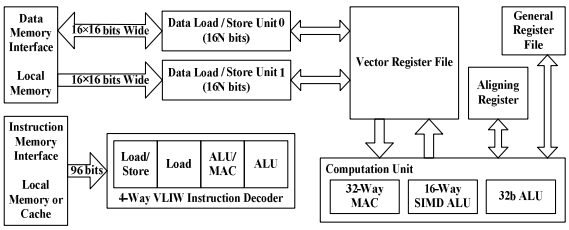


Fig. 6. Principal block diagram of ConnX BBE32

V. RESULTS AND ANALYSIS

In order to study the feasibility of the introduced solution on this platform, we have profiled and then analyzed the solution in terms of number of clock cycles and power consumption. This has been done using the profiling tools provided by the vendor. Two of the most challenging symbols to process in the IEEE 802.11ac packet structure are the VHT-LTF symbol and the DATA symbol. The reason for this is that the VHT-LTF symbol is used for channel estimation and calculating the detector coefficients which are the two most computationally intensive operations. The DATA symbol also involves heavy operations such as the soft bit detection. As a result, we have presented the power consumption and clock cycle results related to these two symbols in this section.

Table II shows the number of clock cycles needed for the different operations on one DATA symbol in the receiver. The results for the four different operation points are presented in the same table. As it can be seen from Fig. 1, the duration of one DATA symbol is 3.6μs when short GI is used. To achieve real-time operation in the receiver, all the processing should not exceed 3.6μs. Looking at the total number of cycles presented in Table II, in the first and third scenarios (scenarios using only two antennas), an operating frequency of 1GHz is required for achieving a real-time operation. However, in the second and fourth cases (where four antennas are used) the clock frequency should be doubled.

The duration of one VHT-LTF symbol is 4μs. According to the total number of clock cycles presented in Table III, to achieve a real-time operation, frequencies less than 1GHz are required for the cases using two space-time streams, whereas very high frequencies may be needed for the cases with four space-time streams due to the high number of cycles consumed by the detector co-efficient calculation function.

TABLE II. THE NUMBER OF CLOCK CYCLES NEEDED FOR THE PROCESSING OF A DATA SYMBOL

	Case A	*Case B*	*Case C*	*Case D*
RCPI Variance	185	365	185	365
RCPI, ANPI, RSNI	146	150	146	150
Remove Pilots	140	336	140	336
Symbol Detection	468	625	208	652
LDPC Tone De-Mapper	324	672	324	672
Stream Deparser	74	254	-	148
Soft Bit Detection	3193	6504	3193	6504
Frequency Error Correction	255	331	255	331
Total	4785	9237	4451	9158

TABLE III. THE NUMBER OF CLOCK CYCLES NEEDED FOR THE PROCESSING OF A VHT-LTF SYMBOL

	Case A	*Case B*	*Case C*	*Case D*
LS Channel Estimation	281	2873	281	2873
LMMSE Channel Estimation	1078	1643	1078	1643
Detector Coefficients	2630	31713	851	34005
Total	3989	36229	2210	38521

The matrix to be inversion for calculating the detector coefficients does not benefit from the special structure available for the LMMSE channel estimation. Therefore, it could not be simplified using the same methods. We continue our work to reduce the frequencies required to achieve a real-time operation by introducing instruction extensions for the bottleneck operations such as 4×4 unstructured matrix inversion in the symbol detector coefficient calculations. This is done by adding a customized inversion accelerator to the core. One important criterion, which needs to be taken into consideration during the implementation, is the power consumed by the design. As the Xtensa Xplorer tools provide the energy consumption estimation, we have calculated the power consumption using the energy numbers and dividing those by time. The time needed for each block is defined by the number of clock cycles and we have considered maximum memory capacity (128k). We have assumed a clock frequency of 500MHz and the monitoring time for the energy analysis is 3.6µs for the DATA part and 4µs for the VHT-LTF. Table IV and V present the power consumption results for different operations in all transmission cases for DATA and VHT-LTF symbols, respectively. In general the power consumption results are mostly found feasible to mobile terminal scale devices. It should be noted that the VHT-LTF operations are done only once per packet, but the data symbol operations are repeated multiple times per packet. Therefore, minimizing the power consumption per data symbol is more critical. In other words, for a VHT-LTF symbol the detector coefficient evaluation is a time critical operation and processing a data symbol is a power critical operation.

VI. CONCLUSIONS

In this paper, we have proposed a software-based implementation for the IEEE 802.11ac receiver frequency domain PHY layer frequency domain baseband processing. This implementation was carried out using a customized DSP core with vector processing capabilities. The solution was developed for four different multi-antenna transmission scenarios. The implementation has been evaluated in terms of number of clock cycles and power consumption. We presented the results for two of the symbols that require more computations, namely the DATA and VHT-LTF symbols. The analysis of the performance numbers showed that achieving a real-time operation for the IEEE 802.11ac receiver on this customized DSP platform requires very high operating frequencies. In the continuation of this work, we customize the core by adding instruction extensions for the computationally intensive operations such as matrix inversion to lower the operating frequency needed for achieving a real-time operation.

TABLE IV. POWER CONSUMPTION IN MW FOR THE PROCESSING OF A DATA SYMBOL

	Case A	*Case B*	*Case C*	*Case D*
RCPI Variance	3,36	6,67	3,35	6,16
RCPI,ANPI,RSNI	2,76	2,68	2,59	2,78
Remove Pilots	4,76	10,39	4,76	9,52
Symbol Detection	13	23,51	6,82	34,65
LDPC Tone De-Mapper	10,02	20,07	10,02	20,03
Stream Deparser	3,06	8,59	3,09	6,11
Soft Bit Detection	106,95	213,92	105,43	213,19
Frequency Error Correction	4,46	7,55	4,44	7,55
Total	148,37	293,38	140,5	299,99

TABLE V. POWER CONSUMPTION IN MW FOR TH EPROCESSING OF A VHT-LTF SYMBOL

	Case A	*Case B*	*Case C*	*Case D*
LS Channel Estimation	8,49	96,45	8,51	85,64
LMMSE Channel Estimation	31,27	51,22	31,27	51,22
Detector Coefficients	66,39	38,4	21,55	94,03
Total	106,15	186,07	61,33	230,89

REFERENCES

[1] NVIDIA, *NVIDIA SDR (Software Defined Radio) Technology*, USA, 2013.

[2] IEEE P802.11ac™ Draft Standard, version 5, January 2013.

[3] S. Yoshizawa and Y. Miyanaga, "VLSI Implementation of a 4×4 MIMO-OFDM transceiver with an 80-MHz channel bandwidth," in *Proc. IEEE ISCAS*, Taipei, Taiwan, 24-27 May 2009, pp. 1743-1746.

[4] S. Eberli, A. Burg, T. Bosch, and W. Fichtner, "An IEEE 802.11a baseband receiver implementation on an application specific processor," in *Proc. 50th Midwest Symposium on Circuits and Systems*, Montreal, Canada, 5-8 Aug. 2007, pp.1324,1327.

[5] P. Wang, J. McAllister, and Y. Wu, "Software defined FFT architecture for IEEE 802.11ac," in *Proc. Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, USA, 3-5 Dec. 2013, pp.1246-1249.

[6] J.R. Gutierrez-Agullo, B. Coll-Perales, and J. Gozalvez, "An IEEE 802.11 MAC Software Defined Radio implementation for experimental wireless communications and networking research," in *Proc. IFIP Wireless Days*, Venice, Italy, 20-22 Oct. 2010, pp.1-5.

[7] M. Aghababaeetafreshi, L. Lehtonen, M. Soleimani, M. Valkama, and J. Takala, "IEEE 802.11ac MIMO transmitter baseband processing on customized VLIW processor," in *Proc. IEEE ICASSP*, Florence, Italy, 4-9 May 2014, pp. 7550-7554.

[8] IEEE Std 802.11™ Part 11: Wireless LAN Medium Acess Control (MAC) and Physical Layer (PHY) Specifications, March 2012.

[9] M. S. Bartlett, "An inverse Matrix Adjustment Arising in Discriminant Analysis," *The Annals of Mathematical Statistics 22*, no. 1, pp. 107-111.

[10] Tensilica Inc., *ConnX BBE32 DSP User Guide*, USA, 2012.

PUBLICATION 3

M. Aghababaeetafreshi, L. Lehtonen, T. Levanen, M. Valkama and J. Takala, "IEEE 802.11ac MIMO transceiver baseband processing on a VLIW Processor", *Journal of Signal Processing Systems*, Oct 2016, 85(1), pp. 167–182, DOI: 10.1007/s11265-015-1032-2

IEEE 802.11ac MIMO Transceiver Baseband Processing on a VLIW Processor

Mona Aghababaeetafreshi · Lasse Lehtonen · Toni Levanen · Mikko Valkama · Jarmo Takala

Abstract Wireless standards are evolving rapidly due to the exponential growth in the number of portable devices along with the applications with high data rate requirements. Adaptable software based signal processing implementations for these devices can make the deployment of the constantly evolving standards faster and less expensive. The flagship technology from the IEEE WLAN family, the IEEE 802.11ac, aims at achieving very high throughputs in local area connectivity scenarios. This article presents a software based implementation for the Multiple Input and Multiple Output (MIMO) transmitter and receiver baseband processing conforming to the IEEE 802.11ac standard which can achieve transmission bit rates beyond 1Gbps. This work focuses on the Physical layer frequency domain processing. Various configurations, including 2×2 and 4×4 MIMO are considered for the implementation. To utilize the available data and instruction level parallelism, a DSP core with vector extensions is selected as the implementation platform. Then, the feasibility of the presented software-based solution is assessed by studying the number of clock cycles and power consumption of the different scenarios implemented on this core. Such Software Defined Radio based approaches can potentially offer more flexibility, high energy efficiency, reduced design efforts and thus shorter time-to-market cycles in comparison with the conventional fixed-function hardware methods.

Keywords OFDM · MIMO · WLAN · VLIW · Software Defined Radio · Parallel Processing

M. Aghababaeetafreshi (✉) · L. Lehtonen · T. Levanen · M. Valkama · J. Takala
Tampere University of Technology, Korkeakoulunkatu 1, 33720, Tampere, Finland
Tel.: +358-44-9761447
E-mail: mona.aghababaeetafreshi@tut.fi

1 Introduction

Wireless standards and protocols are evolving rapidly to meet the high demands by the growing number of users and various applications. Consequently, the idea of more flexible devices adapting to different radio interfaces is gaining more interest. This is due to the fact that such devices would considerably reduce the costs and design efforts as they would eliminate the need for a dedicated hardware for each new technology or new release of an existing standard. However, majority of the available devices still exploit application-specific fixed-function hardware platforms. In these devices, most of the processing is carried out using Application-Specific Integrated Circuits (ASIC), hard-wired to communicate in one specific protocol, thus offering very limited flexibility. These application-specific structures require smaller area and as a result have very low power consumption. However, designing structures to support all the possible standards and even the forthcoming ones in such devices significantly increases the complexity during the design and implementation processes. [1]

In order to compensate the inflexibility of ASIC designs, Software Defined Radio (SDR) solutions have been introduced. SDR solutions provide support for a wide variety of capabilities which would otherwise only be available by integrating multiple radio components. These solutions allow the modification of the functionality by modifying the software using the same hardware resources. Generally, standard programmable solutions have higher power consumption in comparison with the ASIC based implementations. However, with the use of domain-specific processors, which have been tailored to a specific application area such as communication applications, the application-specific features are expected to reduce the power consumption to a reasonable level. Moreover, with the reduced design efforts

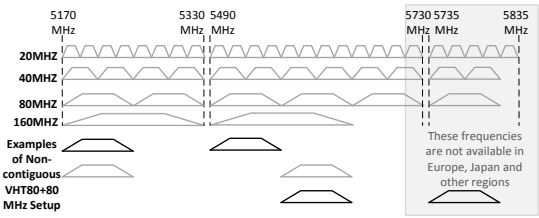


Fig. 1 IEEE 802.11ac channelization at 5 GHz

due to the elimination of silicon re-design and testing, SDR can help shorten the time-to-market cycle. [2][3]

With the increasing number of wireless devices along with the rising popularity of bandwidth intensive applications, the amount of wireless traffic is tremendously growing. As a result, the demand for enhancing the capacity of wireless networks is rising. The IEEE 802.11 WLAN family [4], currently providing the majority of wireless local area connectivity, is constantly developing to meet these high demands particularly in indoor environments. The current flagship amendment to the IEEE 802.11 WLAN standard is the IEEE 802.11ac [5].

The IEEE 802.11ac is introduced to provide improvements to reach maximum throughputs well above one Gigabit per second. Therefore, it is also referred to as the Very High Throughput (VHT) amendment. In comparison with its preceding amendment, the IEEE 802.11n (also referred to as the High Throughput (HT) amendment), this specification employs several techniques to improve the Physical (PHY) layer throughput, including the main components listed below [6].

Mandatory 5 GHz Operation: As the 2.4 GHz band is already crowded by WiFi and other unlicensed band devices and thus is more prone to interference, the VHT specification mandates operations at 5 GHz band.

Wider Bandwidth: The IEEE 802.11ac takes advantage of wider channel bandwidths, such as 80 MHz and 160 MHz. The 160 MHz channel can be formed using a 80+80 MHz non-contiguous setup which allows more flexible channel assignment [7]. The channelization for the IEEE 802.11ac at 5 GHz is shown in Fig. 1.

Denser Modulation: The VHT amendment adds support for higher modulation orders compared to the HT amendment. The highest supported order in this amendment is 256-QAM delivering an increase in data rates by 33% over 802.11n [8].

Higher order and Multi-User MIMO (MU-MIMO): The IEEE 802.11ac has increased the number of spatial streams from four streams allowed in 802.11n to eight. It is also the first amendment to introduce support for serving multiple users in the same time slot in the form of MU-MIMO.

In this paper, a software-defined radio implementation of PHY layer frequency domain baseband processing of IEEE 802.11ac transceiver is described. A vector processor code is used as the target platform where we assume that the high complexity processing, e.g., Low Density Parity Check (LDPC) coding, is carried out in a separate accelerator. The employed processor which utilizes a customized Very Long Instruction Word (VLIW) architecture with vector extensions is the Tensilica ConnX BBE32 [9]. This implementation includes some of the possible transmission scenarios from the IEEE 802.11ac specifications, including 2×2 and 4×4 MIMO. This work is a continuation of the transmitter and receiver implementation presented in [10] and [11], respectively.

The rest of the paper is organized as follows. Section 2 introduces the previous contributions related to this work. In Section 3, the physical layer packet structure and the implemented transmission scenarios are shortly described. Section 4 discusses the technical details of the transmitter and receiver processing. Section 5 describes the implementation and the architecture of the employed platform. The performance results in terms of number of clock cycles and power consumption estimates and their comparison with the existing implementations can be found in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Many studies addressing the implementation of WLAN standards exist in the literature. However, there are only a limited number of works which describe the implementation of the IEEE 802.11ac either exploiting hardware based solutions or a software based approach.

As an example, [12] presents the ASIC implementation of an OFDM transceiver based on the IEEE 802.11a which describes an OFDM design using a fixed 64-QAM modulation. Similarly, [13] describes the design and validation of the IEEE 802.11n PHY layer transceiver processing on an ASIC platform. This implementation is carried out using a fixed 40MHz bandwidth and two spatial streams and is defined to use two transmit antennas and three receive antennas.

Looking at the IEEE 802.11ac based implementations, in [14], a 4×4 MIMO-OFDM transceiver is implemented using a VLSI architecture. This study is carried out for a transceiver tailored to a 80-MHz channel bandwidth and a 4×4 MIMO.

All the above mentioned cases analyze and implement a fixed transmission scenario. However, as the processing and computing power of Digital Signal Processors (DSP) are increasing, the more programmable and

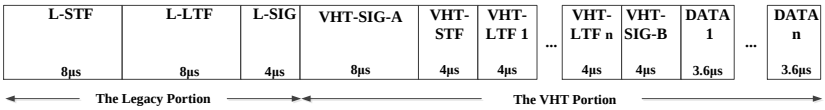


Fig. 2 The IEEE 802.11ac physical layer data packet structure assuming short GI for the data symbols

flexible approaches are gaining interest. Both [15] and [16], explore the feasibility of a SDR baseband processor for the IEEE 802.11ac.

In [15], Ho Yang et al. have covered most functions of baseband with DSP software having an 80MHz bandwidth, 4×4 MIMO, and 64-QAM modulation. The main processor used in this work is the RP-32 radio processor. This DSP features 512-bit vector processing, 256-bit data buses, and 32-lanes. Front-end filters and outer modem (LDPC, etc.) are implemented as separate units from the processor. This implementation assumes 1GHz clock frequency.

The same configuration as [15] is exploited in [16] for the inner receiver processing. However, the baseband processor is based on the ADRES template [17]. An instance of the ADRES template called BOADRES has been derived for this implementation. BOARDES has four memories with 256-bit data buses. Four vector units which support 16-lane operations allow 64 operations in parallel for 16-bit wide data. It is also equipped with 6-unit VLIW for scalar operations. This implementation excludes LDPC as well, but includes the FFT operation which consumes 25-34% of total cycles. A clock frequency of 800 MHz is assumed.

Some other contributions have also been made in software based implementations which only include parts of the PHY or MAC layer processing [18][19].

The solution presented in this paper considers very aggressive scenarios with 4×4 MIMO, 80 MHz bandwidth, 256-QAM, and STBC coding which impose strict timing constraints for a real-time operation in comparison with other existing SDR solutions. The clock frequency is assumed to be 500 MHz. In this paper, we propose a software based SDR solution in a similar fashion and similar frequency domain processing scenario as in [15] and [16]. However, in our proposal we assume lower clock frequency and exploit a dedicated matrix inversion unit.

3 Packet Structure and Considered Transmission Scenarios

3.1 Physical Layer Data Packet Structure

This paper addresses the physical layer processing related to an IEEE 802.11ac data packet. The VHT data packet is divided into a PHY header part and a data part. The header part consist of a legacy portion which ensures the backward compatibility of the IEEE 802.11ac packet format and the VHT part which is specific to this amendment. The former is intended to be received by the non-VHT stations and consists of Non-HT Short Training field (L-STF), Non-HT Long Training field (L-LTF), and Non-HT SIGNAL field (L-SIG). The VHT part is composed of VHT Signal A field (VHT-SIG-A), VHT Short Traing field (VHT-STF), VHT Log Training field (VHT-LTF), VHT Signal B field (VHT-SIG-B), and the DATA field carrying a number of data symbols [5]. Fig. 2 presents the PHY layer packet structure and the duration for different symbols.

3.2 Transmission Scenarios

Four different operation points for the IEEE 802.11ac transmitter and receiver are covered in this work. In all the transmission scenarios, the channel bandwidth is set to 80MHz which indicates 256 OFDM subcarriers including 234 data, 14 null, and 8 pilot subcarriers. Additionally, all the cases employ 256-QAM as the data modulation scheme. Table 1 shows the different implemented transmission scenarios and highlights differences in these scenarios.

Table 1 The implemented transmission scenarios. For all cases 80 MHz T_x/R_x BW, 256-QAM, LDPC coding, 3/4 coding rate and short GI are assumed.

Cases	Number of T_x/R_x antennas	Number of spatial streams	STBC coding
Case A	2	2	NO
Case B	4	4	NO
Case C	2	1	YES
Case D	4	2	YES

Fig. 3 illustrates the transmitter chain block diagram, considered in this article, for the processing of a DATA symbol. Multiple dots in the figure indicate the possibility of more spatial streams. Space-Time Block Coding (STBC) is presented with dashed lines showing that it is not implemented in all the scenarios.

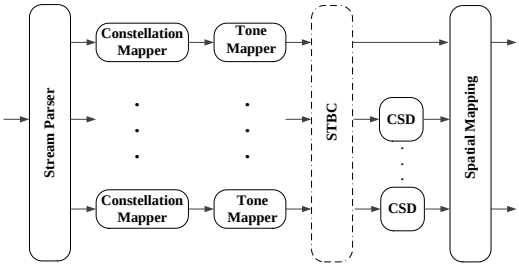


Fig. 3 Principal block diagram of the transmitter baseband processing

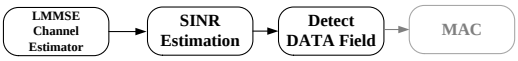


Fig. 4 Overall logical block diagram of the receiver processing

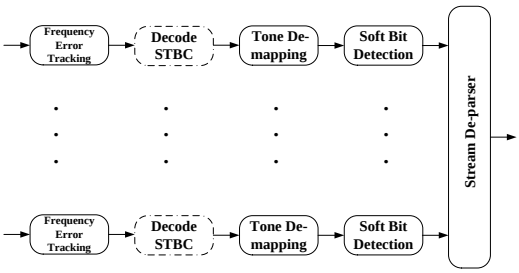


Fig. 5 Principal block diagram of the receiver baseband processing

In the receiver, processing of the DATA field relies on the processing of the preamble parts. The overall receiver logical block diagram can be seen in Fig. 4, where some of the preamble processing is also included.

Fig. 5 shows the functional blocks implemented for the processing of a DATA symbol at the receiver.

4 IEEE 802.11ac Functional Blocks

In this section an overview of the IEEE 802.11ac functional blocks considered both in the transmitter and the receiver is given.

4.1 Transmitter Processing

This section describes some of the functionalities of the transmitter implemented in this work. Since these functionalities are either redundant or less complex for the preamble symbols (shown in Fig. 2), only the processing of the DATA field is presented in this section.

It is assumed that after scrambling, the Forward Error Correction (FEC) unit encodes the bits. Binary Convolutional Codes (BCC) or LDPC codes can be used as the FEC method. Then, the encoded bits go through the blocks implemented in this work (shown in Fig. 3). Finally, the symbols are fed to a separate hardware entity to perform IFFT and then have the relevant time domain processing.

4.1.1 Stream Parsing

The incoming bits from the LDPC encoder have to be re-arranged into a new set of bit strings equal to the number of spatial streams (N_{ss}). A block of bits of size s, defined in (1), are assigned to different spatial streams in a round robin fashion [6].

$$s = max\left\{1, \frac{N_{BPSCS}}{2}\right\} \tag{1}$$

where N_{BPSCS} is the number of coded bits per single subcarrier for each spatial stream and is equivalent to the modulation order.

As an example, the stream parsing process in case of a 256-QAM modulation ($s = 4$) and $N_{ss} = 4$ divides the stream $[y_0, y_1, y_2, ..., y_i]$ into four streams of $[y_0, y_1, y_2, y_3, y_{16}, y_{17}, ..]$, $[y_4, y_5, y_6, y_7, y_{20}, y_{21}, ..]$, $[y_8, y_9, y_{10}, y_{11}, y_{24}, y_{25}, ..]$ and $[y_{12}, y_{13}, y_{14}, y_{15}, y_{28}, y_{29}, ..]$.

4.1.2 Modulation Mapping

The bit streams from the stream parser are divided into blocks of N_{BPSCS} bits and converted into complex numbers representing BPSK, 16-QAM, 64-QAM, or 256-QAM constellation points. This conversion is performed according to Gray-coded constellation mappings. Then, the output values are formed by multiplying the resulting $(I + jQ)$ value by a normalization factor K_{mod}. The normalization factor, K_{mod}, for 256-QAM, the employed modulation order in this work, is $\frac{1}{\sqrt{170}}$.[4][5]

4.1.3 LDPC Tone Mapping

LDPC tone mapping is performed when the bits are LDPC encoded. It is introduced in IEEE 802.11ac to achieve full frequency diversity from 80 and 160MHz bandwidths. LDPC tone mapping makes each two consecutively generated complex constellation numbers in an OFDM symbol to be transmitted on two data tones that are separated by a known distance from other data tones [5]. This distance is selected according to the channel bandwidth. Fig. 6 shows the tone mapping process having an 80MHz channel bandwidth.

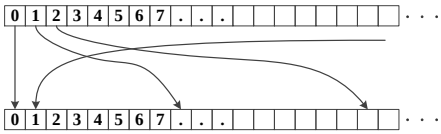


Fig. 6 Tone mapping in 80MHz wide channel

4.1.4 STBC Coding

STBC is used for transmitter diversity. The STBC coder maps a single constellation symbol into multiple radio chains. As a result, the spatial streams transform into space-time streams. This is used for improving the reliability and robustness of the data transfer.

As an example, let us assume a two antenna configuration with one spatial stream. According to Alamouti's scheme [20], at the time t, symbols x_1 and x_2 are transmitted from antenna 1 and antenna 2, respectively. Assuming that the symbols have a duration of T, at the time $T+t$, the symbols $-x_2^*$ and x_1^* are transmitted from antenna 1 and antenna 2, respectively. x^* represents the complex conjugate of symbol x.

4.1.5 Cyclic Shift Diversity

Cyclic shifts are applied to prevent unintended beamforming when correlated signals are transmitted in multiple space-time streams [5]. This is avoided by giving the signal transmitted from an antenna a large phase shift relative to the others. In the time domain, this is perceived as a delay in the signal. Different phase shift values are applied to the non-VHT and VHT fields.

4.1.6 Spatial Mapping

Spatial mapping creates the final antenna signals from the parallel streams. This may be carried out in the transmitter using different techniques such as direct mapping or spatial expansion. Here, the spatial mapper is applied to only map the space-time streams into the transmit chains, thus each stream is only scaled with a normalization factor. The normalization factor is equal to the square root of the number of space-time streams.

4.2 Receiver Processing

After the relevant time domain processing at the receiver, the symbols are fed to a dedicated hardware entity to perform FFT. Then, the symbols go through the blocks implemented in this work (shown in Fig. 4 and 5). Finally, they are fed to the LDPC decoder and then the descrambler.

In contrast with the transmitter operations, in the receiver, the preamble fields also require some high complexity processing. Hence, some of the operations for the preamble parts are also addressed in this section.

Some of the functional blocks in the receiver chain are simply just reversing the processing carried out in the transmitter. Thus, it is rather straight forward to derive their functionality based on the discussions in Section 4.1. As a result, stream de-parsing and LDPC tone de-mapping are not included in this section.

4.2.1 SINR Estimation

The receiver measures the SINR with the aim of improving the link quality. These measurements can be used for adjusting the transmit power level or dynamically adapting the modulation and coding schemes or the data rate. In this implementation, the Received Channel Power Indicator (RCPI), Average Noise Power Indicator (ANPI), and the Received Signal to Noise Indicator (RSNI) are calculated.

RCPI: RCPI is a measure of the total channel power, including signal, noise, and interference of a received frame [4]. To calculate RCPI, the average power received over all receiver antennas should be calculated. It can be measured using the VHT-LTF and VHT-SIG-B symbols in case of receiving a Null Data Packet (NDP), otherwise, the DATA symbols are considered. RCPI measured as the average power over all active non-pilot subcarriers in a DATA symbol can be written as:

$$RCPI = \frac{1}{N_{Rx} N_s N_t} \sum_{Rx} \sum_{t} \sum_{i \in I} |y_{Rx,t,i}|^2$$

where $R_x, R_x = 1, 2, .., N_{Rx}$, is the receiver antenna index, $t, t = 1, 2, .., N_t$, is the DATA symbol index, and $N_s = |\mathbf{I}_{active,non-pilot\,subcarriers}|$, in which $|I|$ represents the cardinality of the set I.

In this work, we measure RCPI for each DATA symbol, and update the average after the reception of the following symbol until the whole packet is received.

ANPI: In [4], ANPI is defined as a MAC indication of the average noise plus interference power measured during idle periods of the channel. It is also mentioned that ANPI may be calculated over any period and for any received frame and any equivalent method may be used to measure ANPI. In this work, the ANPI value is used for symbol detection. Consequently, the alternative approach used is to calculate ANPI based on the

average power over the null subcarriers except DC. As the L-STF and VHT-STF contain many zeros in the frequency domain in addition to non-active carriers, they are suitable for ANPI measurement. This is due to the fact that any change in these subcarriers can be considered as noise. The ANPI value using the received L-STF or VHT-STF symbols can be calculated as:

$$ANPI = \frac{1}{N_{Rx} N_z} \sum_{Rx} \sum_{i \in I} |y_{Rx,i}|^2 \tag{2}$$

where $N_z = |\mathbf{I}_{active, zero-valued\, pilot\, subcarriers}|$.

It is assumed that the time and frequency synchronization accuracy is sufficient so that only the noise power is measured in zero-valued subcarriers. After properly synchronizing, this can be done as a post processing step.

RSNI: According to the IEEE 802.11 standard, RSNI is an indication of the signal to noise plus interference ratio of a received frame. RSNI is measured using the calculated RCPI and ANPI and it can be obtained as:

$$RSNI = 10 \log_{10} \frac{RCPI - ANPI}{ANPI} \tag{3}$$

The RCPI and ANPI represent the power values in linear scale.

The measured values for RCPI, ANPI, and RSNI can be averaged for achieving better stability in the system. These averaging measurements should be obtained closely in time for high correlation.

4.2.2 Channel Estimation

To estimate the channel, the LTF symbols are used. As defined in [5], the data tones of each VHT-LTF symbol are precoded to enable channel estimation at the receiver, whereas the L-LTF symbols are not precoded. This means that two channel estimates have to be calculated to detect a packet at the receiver, one for the non-precoded parts and one for the precoded ones. First the L-SIG and VHT-SIG-A symbols are detected using the channel estimate obtained from the L-LTF symbol, then the second channel estimation is done based on the VHT-LTF symbols.

Channel estimation for the legacy part: After time and frequency synchronization, Cyclic Prefix (CP) removal, and FFT, the received L-LTF symbols per subcarrier $k, k \in \mathbf{I}_{active,\, non-pilot\, L-LTF\, subcarriers}$, and symbol index $t, t = [1, 2]$ can be written as:

$$\begin{aligned} \mathbf{y}_{k,t} &= \mathbf{H}_k \mathbf{x}_{L-LTF,k} + \mathbf{n}_{k,t} \\ &= x_{L-LTF,k} \begin{bmatrix} \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} h_{1,j} \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{N_{Tx}} \sum_{j=1}^{N_{Tx}} h_{N_{Rx},j} \end{bmatrix} + \mathbf{n}_{k,t} \\ &= x_{L-LTF,k} \mathbf{h}_{eff,k} + \mathbf{n}_{k,t} \end{aligned} \tag{4}$$

where N_{Tx} is the number of transmit antennas, $\mathbf{H}_k$ is a $(N_{Rx} \times N_{Tx})$ complex channel matrix, $\mathbf{x}_{L-LTF,k}$ is a $(N_{Tx} \times 1)$ real vector containing the training symbols $x_{L-LTF,k}$ (containing only ones or minus ones), $\mathbf{n}_{k,t}$ is a $(N_{Rx} \times 1)$ complex Gaussian noise vector, and $\mathbf{h}_{eff,k}$ is the $(N_{Rx} \times 1)$ effective sum channel vector for the legacy part.

Taking into consideration that $x_{L-LTF,k}$ can only have one or minus one values, and that two L-LTF ($t = [1, 2]$) symbols are received, the effective Least Square (LS) channel estimate per subcarrier k can be calculated by averaging the received symbols $\mathbf{y}_{k,t}$ over time and multiplying them with the known symbol value $x_{L-LTF,k}$ as shown in (5).

$$\hat{\mathbf{h}}_{LS,k} = \frac{x_{L-LTF,k}}{2} \sum_{t=1}^{2} \mathbf{y}_{k,t} \tag{5}$$

The LS channel estimate can be used for FFT smoothing or wiener filtering before LMMSE estimate calculation.

Now having the LS channel estimate, the effective Linear Minimum Mean Square Error (LMMSE) channel estimate per subcarrier k can be given as:

$$\begin{aligned} \hat{\mathbf{h}}_{LMMSE,k} = \hat{\mathbf{h}}_{LS,k} \hat{\mathbf{h}}_{LS,k}^H \\ \times (\hat{\mathbf{h}}_{LS,k} \hat{\mathbf{h}}_{LS,k}^H + \frac{\sigma_n^2}{2} I_{N_{Rx}})^{-1} \hat{\mathbf{h}}_{LS,k} \end{aligned} \tag{6}$$

where $\mathbf{h}^H$ represents the Hermitian transpose of vector $\mathbf{h}$ and σ_n^2 is the noise variance.

Channel estimator for the VHT part: Having the legacy channel estimate, the VHT-SIG-A symbol can be detected. Thus, the receiver knows the number of VHT-LTF symbols ($N_{VHT-LTF}$) to be collected for the VHT channel estimation. As stated before and defined in [5], the VHT-LTF symbols are precoded by matrix $\mathbf{P}$ and may also be precoded by precoder matrix $\mathbf{Q}_j$, $j \in \mathbf{I}_{active,\, VHT-LTF\, subcarriers}$. Thus, after time and frequency synchronization, CP removal and FFT, the received VHT-LTF symbols per subcarrier

$k, k \in \mathbf{I}_{active,\, non-pilot\, VHT-LTF\, subcarriers}$ and symbol index $t, t = [1, .., N_{VHT-LTF}]$ can be written as:

$$\begin{aligned}\mathbf{y}_{k,t} &= \mathbf{H}_k \mathbf{Q}_k \mathbf{P}(:,t) x_{VHT-LTF,k} + \mathbf{n}_{k,t} \\ &= \mathbf{H}_{eff,k} \mathbf{P}(:,t) x_{VHT-LTF,k} + \mathbf{n}_{k,t}\end{aligned} \tag{7}$$

For presentation clarity, the received symbols and the effective channel matrix per subcarrier k over all R_x antennas and $N_{VHT-LTF}$ symbols are stacked into a single column vector. Now the received training symbols can be given as:

$$\begin{aligned}\mathbf{y}_k &= \begin{bmatrix} \mathbf{y}_{k,1} \\ . \\ . \\ . \\ \mathbf{y}_{k,N_{Tx}} \end{bmatrix} \\ &= (\mathbf{P} \otimes \mathbf{I}_{N_{Rx}})^T \begin{bmatrix} \mathbf{h}_{eff,k,1} \\ . \\ . \\ . \\ \mathbf{h}_{eff,k,N_{Tx}} \end{bmatrix} \\ &\times x_{VHT-LTF,k} + \mathbf{n}_k\end{aligned} \tag{8}$$

where $\otimes$ is the Kronecker tensor product.

The received VHT-LTF symbols should be weighted with rows of the $\mathbf{P}$ matrix and averaged over all VHT-LTF symbols to get an effective channel estimate per Space Time Streams (STS) in the receiver. Now, the received VHT-LTF symbols after decoding diversity coding can be given as:

$$\begin{aligned}\tilde{\mathbf{y}}_k &= \frac{1}{N_{VHT-LTF}} (\mathbf{P} \otimes \mathbf{I}_{N_{Rx}}) \begin{bmatrix} \mathbf{y}_{k,1} \\ . \\ . \\ . \\ \mathbf{y}_{k,N_{Tx}} \end{bmatrix} \\ &= x_{VHT-LTF,k} \begin{bmatrix} \mathbf{h}_{eff,k,1} \\ . \\ . \\ . \\ \mathbf{h}_{eff,k,N_{Tx}} \end{bmatrix} + \mathbf{w}_k\end{aligned} \tag{9}$$

where $\mathbf{w}_k \in \mathbf{n}(0, \frac{\sigma_n^2}{N_{VHT-LTF}})$. Thus, the effective LS channel estimate per subcarrier k can be given as:

$$\hat{\mathbf{H}}_{eff,LS,k} = x_{VHT-LTF,k} \hat{\tilde{\mathbf{Y}}}_k \tag{10}$$

where $\hat{\tilde{\mathbf{Y}}}_k = \left[\tilde{\mathbf{y}}_{k,1}, .., \tilde{\mathbf{y}}_{k,N_{Tx}}\right]$, in which $\tilde{\mathbf{y}}_{k,N_{Tx}}$ is a column vector derived from $\tilde{\mathbf{y}}_k$ in (9) containing the weighted received symbols from all N_{Rx} antennas.

Then, the LMMSE channel estimate for the VHT part, using the obtained LS channel estimate in (10), can be derived by extending (6) as

$$\begin{aligned}\hat{\mathbf{H}}_{eff,LMMSE,k} &= \hat{\mathbf{H}}_{eff,LS,k} \hat{\mathbf{H}}_{eff,LS,k}^H \\ &\times (\hat{\mathbf{H}}_{eff,LS,k} \hat{\mathbf{H}}_{eff,LS,k}^H + \sigma_n^2 I_{N_{Rx}})^{-1} \hat{\mathbf{H}}_{eff,LS,k}\end{aligned} \tag{11}$$

Calculating the LMMSE channel estimate matrix includes a matrix inversion, as it can be seen in (11). This inversion results in quite complex and time consuming operations. To simplify this, let us rewrite (11) using the $(N_{Rx} \times N_{Tx})$ effective channel estimation matrix $\hat{\mathbf{H}}_{LS}$ calculated according to (10), having $N_{Rx} = N_{Tx} = 2$ and $\hat{\mathbf{H}}_{LS}$ written as $\hat{\mathbf{h}}_{LS} = [h_1, h_2, h_3, h_4]^T$:

$$\begin{aligned}\hat{\tilde{\mathbf{h}}}_{LMMSE} &= \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \left[h_1^* h_2^* h_3^* h_4^*\right] \\ &\times \left(\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \left[h_1^*\ h_2^*\ h_3^*\ h_4^*\right] + \mathbf{N} \right)^{-1} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}\end{aligned} \tag{12}$$

where $\mathbf{N} = \sigma_n^2 \mathbf{I}_{N_{Rx}}$, and the superscript h^* represents the complex conjugate of element h. As previously mentioned, here the columns of the original channel estimate matrix are stacked on top of each other in $\hat{\mathbf{h}}_{LS}$.

Now, using the Shannon-Morrison law [21], (12) can be written as:

$$\begin{aligned}\hat{\tilde{\mathbf{h}}}_{LMMSE} &= \frac{h_1 h_1^* + h_2 h_2^* + h_3 h_3^* + h_4 h_4^*}{\sigma_n^2 + h_1 h_1^* + h_2 h_2^* + h_3 h_3^* + h_4 h_4^*} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \\ &= \frac{\hat{\mathbf{h}}_{LS} \hat{\mathbf{h}}_{LS}^T \times conj(\hat{\mathbf{h}}_{LS})}{\sigma_n^2 + \hat{\mathbf{h}}_{LS}^T \times conj(\hat{\mathbf{h}}_{LS})}\end{aligned} \tag{13}$$

The same approach can be employed in the case of a 4×4 antenna configuration. This helps reduce the complexity of matrix inversion and thus the channel estimation process to a great extent.

4.2.3 Frequency Domain Pilot Symbol Based Residual Frequency Error Tracking

Moving to the operations needed for the processing of a DATA symbol, first the frequency error should be estimated. The method employed in this process uses the pilot subcarriers in two consecutive DATA symbols. Let us define a weighting vector based on the received power per pilot symbol as:

$$\mathbf{w} = \frac{1}{\sum_{i=1}^{N_p} \sigma_{I_P(i)}^2} \left[\sigma_{I_P(1)}^2 \sigma_{I_P(2)}^2 ... \sigma_{I_P(N_p)}^2\right]^T \tag{14}$$

where index i runs through available pilot indices from the pilot index set I_P and $N_p = |I_P|$ is the number of elements in the pilot index set. The weighting vector is used to limit the degradation of the phase rotation estimate due to the low power pilot symbols caused by frequency selective channel fading.

Now the phase rotation estimate, $\hat{\Theta}_t$, between two consecutive symbols at time instant t due to the frequency error F_{error} is defined as:

$$\hat{\Theta}_t = \mathbf{w}^T \left(arg\left(\mathbf{P}_{t-1}\right) - arg\left(\mathbf{P}_t\right)\right) \tag{15}$$

where $arg\left(e^{jx}\right) = x$ gives the argument of the complex number and $t = [1, ..., N_t]$ indicates the DATA symbol index. In this notation, $t = 0$ points to the VHT-SIG-B symbol and its pilot symbols.

Now, from the phase rotation estimates we can obtain the frequency error estimate at DATA symbol index t defined as:

$$\hat{\mathbf{F}}_{error,t} = \frac{1}{2\pi t\left(N_s + N_{GI}\right)} \sum_{i=1}^{t} \hat{\Theta}_i \tag{16}$$

where F_s is the sampling frequency, N_s is the number of subcarriers, and N_{GI} corresponds to the number of samples in the guard interval (GI) used for the DATA symbols. Due to the averaging of the phase rotation estimates we obtain improved frequency error estimate in the end of the DATA field. By using the phase rotation estimate $\hat{\Theta}_t$, the corrected received symbol estimates for subcarrier k at time instant t is obtained, given as:

$$\hat{\mathbf{y}}_k = exp\left(j \sum_{i=1}^{t} \hat{\Theta}_i\right) \mathbf{y}_k \tag{17}$$

4.2.4 Symbol Detection

The received symbols can be detected using the effective LMMSE channel estimation matrix. The method used in this implementation is LMMSE symbol detection where the detector coefficients per subcarrier k are calculated as:

$$\mathbf{D}_{coeff,k} = (\hat{\mathbf{H}}_k^H \hat{\mathbf{H}}_k + \sigma_n^2 \mathbf{I}_{N_{STS}})^{-1} \hat{\mathbf{H}}_k^H \tag{18}$$

where $\hat{\mathbf{H}}_k$ is the LMMSE channel estimate calculated in (11), and N_{STS} is the number of space-time streams.

Once the coefficients are calculated, the symbols can be detected using:

$$\hat{\mathbf{x}} = \mathbf{D}_{coeff,k} \mathbf{y}_k \tag{19}$$

where $\mathbf{y}_k$ is a $(N_{Rx} \times 1)$ vector containing the received symbols of subcarrier index k.

For cases C and D, the LMMSE channel estimate is defined with regards to the STBC coding. Thus, for the 2×2 case, $\hat{\mathbf{H}}_k$ is written as:

$$\hat{\mathbf{H}}_k = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ h_{12}^* & -h_{11}^* \\ h_{22}^* & -h_{21}^* \end{bmatrix} \tag{20}$$

where h_{ij} is the channel from i^{th} receiver antenna to j^{th} transmit antenna. This can be extended to the 4×4 STBC case, as well.

Additionally the received symbols per subcarrier k should be defined taking into consideration the adoption of the STBC coding. Thus, for the STBC case C, $\mathbf{y}_k$ is defined as:

$$\mathbf{y}_k = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{21}^* \\ y_{22}^* \end{bmatrix} \tag{21}$$

where y_{xy} is the the received symbol at time slot x on receive antenna y. This is easily extended for the 4×4 configuration for case D.

Calculating $\mathbf{D}_{coeff}$ includes a matrix inversion which does not benefit from the same structure as the LMMSE channel estimation. A solution for reducing the implementation complexities of this operation is presented in Section 5.2.

4.2.5 Soft Bit Detection

After tone de-mapping, the symbols arrive at the soft bit detection unit to get demodulated. First, the nearest constellation point corresponding to the received symbol is found. Then, for the soft bit detection, for each bit position, the difference of the distances to the nearest zero and one bit on the constellation point is calculated. Fig. 7 illustrates this process.

This is a sub-optimal approach to reduce the complexity of the soft detection. In this method, instead of calculating the distances to all constellation points, only the distances to the nearest ones are calculated.

4.2.6 LDPC Decoder

As already mentioned earlier, it is assumed that this high complexity operation can be implemented as a coarse-grain accelerator in the system. Such an accelerator is not tightly coupled as the matrix inversion accelerator, thus allowing the frequency domain processing to be done in parallel with the LDPC decoding. Hence, using the existing architectures in the literature, the LDPC decoder can be added to the receiver chain.

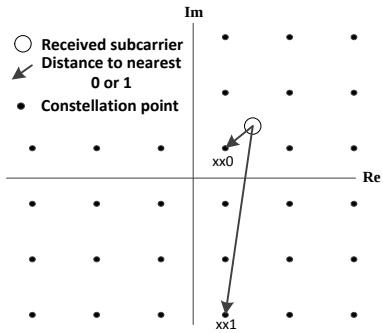


Fig. 7 Soft bit detection process

An example architecture can be found in [22], where a high-throughput LDPC decoder architecture that supports multiple code rates like 1/2, 2/3, 3/4, and 5/6 is described. Moreover, some other possible architectures are presented in [23] [24].

5 SDR Implementation

5.1 Processor Architecture

As explained in the introduction section, the 802.11ac standard focuses on achieving very high throughputs which consequently imposes very high requirements for the processing platform. The huge amount of data to be processed exposes a lot of data level parallelism which can be exploited with Single Instruction Multiple Data (SIMD) operations. Additionally, employing a Very Long Instruction Word (VLIW) processor can help to further utilize the instruction level parallelism.

The Tensilica ConnX BBE32, a VLIW processor with vector capabilities, is selected for this work. The BBE32 DSP core is specifically designed for the next generation wireless communication systems [9]. This very high performance DSP core has ultra low power consumption and a small size architecture. A 16-way SIMD ALU and a 4-issue VLIW processing pipeline makes this core a proper choice for this application. Moreover, it is equipped with 32 multiply-accumulate units and can access wide data in blocks of 256 bits from the memory. As it can be seen from Fig. 8, the BBE32 core has a Harvard architecture with one instruction memory and two data memories.

This core is configurable and special functional units can be added to the architecture for further speedup. Some computationally intensive operations such as FFT/IFFT are also available as dedicated hardware parts which help reduce the load on the processor [26]. These accelerator blocks can be controlled using custom instruction extensions.

Tensilica provides an eclipse based software development environment, namely Xtensa Xplorer. This environment is equipped with a comprehensive collection of code generation, profiling, and analysis tools. In Xtensa Xplorer, software development can be carried out in C programming language. However, the results of this work which are discussed in detail in Section 6.1 are achieved by manually optimizing the code by relying on the provided processor intrinsics.

5.2 Accelerator for Matrix Inversion

The most computationally intensive and time consuming function in the transceiver is matrix inversion in the 4×4 cases. In this section, we address the accelerator implemented for the BBE32 core to reduce the number of clock cycles required for matrix inversion. This discussion focuses on the computations needed in case B from Table 1 which are the most intensive ones.

5.2.1 Modified Gram-Schmidt Algorithm

Modified Gram-Schmidt (MGS) algorithm is one of the numerically stable algorithms for matrix inversion. To reduce the implementation complexity we used the approach from [25] and implemented the computations in $\log_2$ and x^2 domains where the multiplication, square, square root, and division of complex numbers turn into additions and subtractions. This approach requires a number of domain conversions and here these are realized with the aid of Lookup Tables (LUT). LUTs for $\log_2$ calculation of complex numbers are rather inconvenient to implement, thus requiring the decomposition of the complex matrix to a real matrix, which results in an 8×8 real matrix in case of a 4×4 complex matrix using:

$$A = \begin{bmatrix} \mathbf{R(H)} & -\mathbf{I(H)} \\ \mathbf{I(H)} & \mathbf{R(H)} \end{bmatrix} \tag{22}$$

where $\mathbf{R(H)}$ and $\mathbf{I(H)}$ are the real and imaginary parts of the complex channel matrix $\mathbf{H}$, respectively.

The inversion of matrix $\mathbf{A}$ comprises of three major steps:

- QR decomposition of the matrix into the upper triangular matrix $\mathbf{R}$ and orthogonal matrix $\mathbf{Q}$
- Calculating $\mathbf{R}^{-1}$ since $\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^H$ (knowing that $\mathbf{Q}$ is an orthogonal matrix, $\mathbf{Q}^H = \mathbf{Q}^{-1}$)
- multiplication of $\mathbf{R}^{-1}$ with $\mathbf{Q}^H$

The column-wise algorithm implementing the MGS can be given as follows [25]:

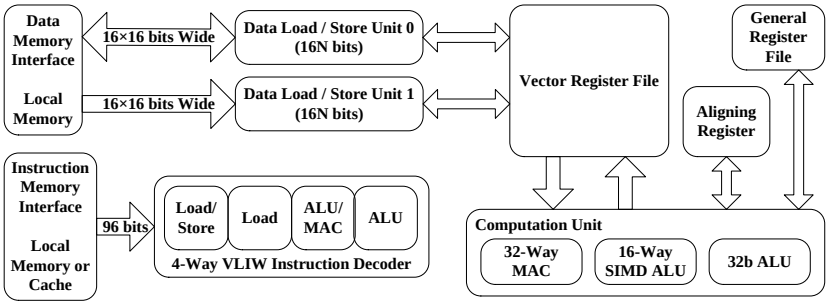


Fig. 8 Principal block diagram of Connx BBE32

for $j = 1 : n$ **do**
  $\mathbf{w}_j = \mathbf{a}_j$
  for $i = 1 : (j-1)$ **do**
    $r_{ij} =< \mathbf{w}_j, \mathbf{q}_i >$
    $\mathbf{w}_j = \mathbf{w}_j - r_{ij}\mathbf{q}_j$
  end for
  $\mathbf{q}_j = \mathbf{w}_j / \|\mathbf{w}_j\|_2$
  $r_{jj} = \|\mathbf{w}_j\|_2$
end for

where $\mathbf{a}_j$ is the jth column of matrix $\mathbf{A}$, $\mathbf{w}_j$ is a temporary vector, r_{ij} is the element from row i and column j of the $\mathbf{R}$ matrix, $\mathbf{q}_j$ is the jth column of the $\mathbf{Q}$ matrix, $< \mathbf{w}_j, \mathbf{q}_i >$ is the inner product of $\mathbf{w}_j$ and $\mathbf{q}_j$, and $\|\mathbf{w}_j\|_2$ is the Euclidean norm of vector $\mathbf{w}_j$.

Next, to calculate $\mathbf{R}^{-1}$:

for $j = 1 : n$ **do**
  for $i = 1 : (j-1)$ **do**
    $ir_{ij} = ir(i, (1 : j-1)) * r((1 : j-1), j)$
  end for
  $ir_{(1:j-1),j} = -ir_{(1:j-1),j} / r_{ij}$
  $ir_{jj} = 1 / r_{jj}$
end for

where ir_{ij} is the element from row i and column j of the inverse of the $\mathbf{R}$ matrix.

With the use of LUTs, the above mentioned algorithms result in operations similar to Fig. 9 and Fig. 10. Fig. 9 illustrates the calculations for updating $\mathbf{w}$ in $\mathbf{w}_j = \mathbf{w}_j - \mathbf{r}_{ij}\mathbf{q}_i$ where $\mathbf{w}_j$ is in linear domain, and $\mathbf{r}_{ij}$ and $\mathbf{q}_i$ are in $\log_2$ domain.

The operations regarding the calculation of the inner product $\mathbf{r}_{ij} =< \mathbf{w}_j, \mathbf{q}_i >$ using LUTs can be seen in Fig. 10 in which the inputs are in $\log_2$ domain.

Operations similar to Fig. 9 and Fig. 10 can be carried out for the other computations needed for the MGS-QR algorithm, such as $\|\mathbf{w}\|_2$ calculation using the $\log_2$ and x^2 LUTs. As a result, the complete matrix inversion process can be implemented as an accelerator for the BBE32 core using only additions, subtractions, and LUTs.

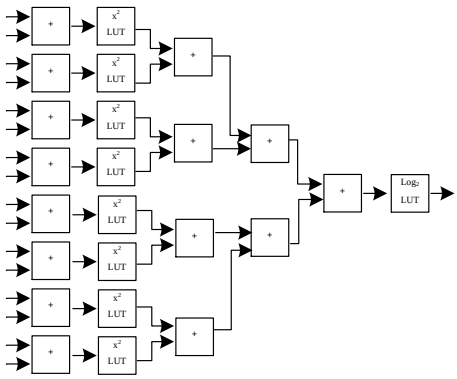


Fig. 9 Calculating $\mathbf{w}_j = \mathbf{w}_j - \mathbf{r}_{ij}\mathbf{q}_i$ using LUTs

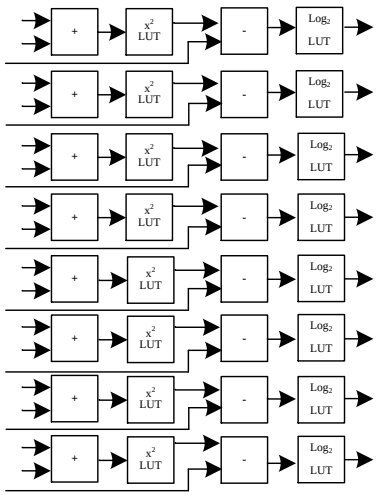


Fig. 10 Inner product calculation for $\mathbf{r}_{ij} =< \mathbf{w}_j, \mathbf{q}_i >$ using LUTs

5.2.2 Requirements for the Matrix Inversion

In order to find the cycle budget for the matrix inversion task, the timing constraints in the non-STBC 4×4 case should be analyzed. Knowing that each VHT-LTF symbol has $4\mu s$ time duration and assuming a 500MHz operating frequency, there will be 2000 cycles available in total for all the processing carried out for the VHT-LTF symbol in the receiver. The fourth VHT-LTF symbol is used for calculating the detector coefficients. After the reception of the fourth VHT-LTF symbol, the LS and LMMSE channel estimations, which are used to determine the detector coefficients, can be carried out in 1400 clock cycles (presented in Section 6.1). This leaves approximately 600 cycles, out of the total 2000 available cycles, for the channel matrix inversion.

As described in Section 3.2, each spatial stream carries 234 data subcarriers and thus for the inversion of the channel matrices, 234 complex 4×4 matrices should be inverted. BBE32 has two 256-bit interfaces to two local memories and implements the complex numbers as 16+16-bit fixed-point numbers meaning that it takes two clock cycles to read and write each 4×4 complex matrix to/from memory simultaneously. This means that just the reading/writing of the data from/to the memory takes $2 \times 234 = 468$ clock cycles. As a result only about 100 clock cycles of the total 600 will be left for matrix inversion with throughput of one matrix every other cycle.

5.2.3 Implementation

Fig. 11 presents the pipeline schedule of the developed unit for inverting complex-valued 4×4 matrices. Assuming these operations would have a register for pipelining after each LUT, the pipeline would cause approximately 64 clock cycle latency plus few clock cycles to buffer the data coming from the BBE32 and going back to it. Overall, to process all of the computations, the implementation needs 1622 adders (mostly 16-bit), 877 x^2 LUTs (256×15 bits), 281 $\log_2$ LUTS (64×14 bits). Moreover, considering the fact that the BBE32 cannot write out or read in more than half a 4×4 complex matrix per clock cycles, half of the adders and LUTs could be reused to achieve the throughput of one matrix per two clock cycles. This means a total of 468 cycles (234×2) for the inversions in each symbol. As presented in Section 6.1, for calculating the $\mathbf{D}_{coeff}$ a total of 548 cycles are consumed which is below the available 600 cycles. The empty slots shown in Fig. 11 allow reusing half of the resources so that, in total, about a quarter hardware resources are needed to calculate the operations (417 adders, 220 x^2 LUTs, 71 $\log_2$ LUTs).

The developed matrix inversion accelerator provides significant speedup for the IEEE 802.11ac processing and allows real-time operation for the frequency domain processing in the transceiver.

6 Results and Analysis

In this section, the results from the implementation of the discussed scenarios in Section 3.2 for the transmitter and receiver functional blocks are presented and analyzed. These implemented blocks are evaluated in terms of number of clock cycles and power consumption with the instruction set simulator and profiling tools. As mentioned in the previous sections, the most time consuming and complex field to process is the DATA field. Additionally, in the receiver, the VHT-LTF field includes some heavy processing due to the detector coefficient calculations and channel estimation. Thus, the results related to the DATA symbol in the transmitter and both DATA and VHT-LTF symbols in the receiver are presented.

6.1 Obtained Results

The main focus of this paper is studying the feasibility of achieving a real-time implementation on this platform. To address that, the number of clock cycles for different operations should be considered. Table 2 shows the results for the transmitter side processing of a DATA symbol.

Table 2 The number of clock cycles needed for the processing of a DATA symbol in the transmitter

Functional Blocks	Case A	Case B	Case C	Case D
Preparation + STBC	53	-	111	68
Tone Mapper	159	159	159	159
Stream Parser + Constellation Mapper	153	197	153	300
Pilots + CSD + Phase Rotation + Spatial Mapping	130	210	136	256
Total Number of Cycles	495	616	559	783

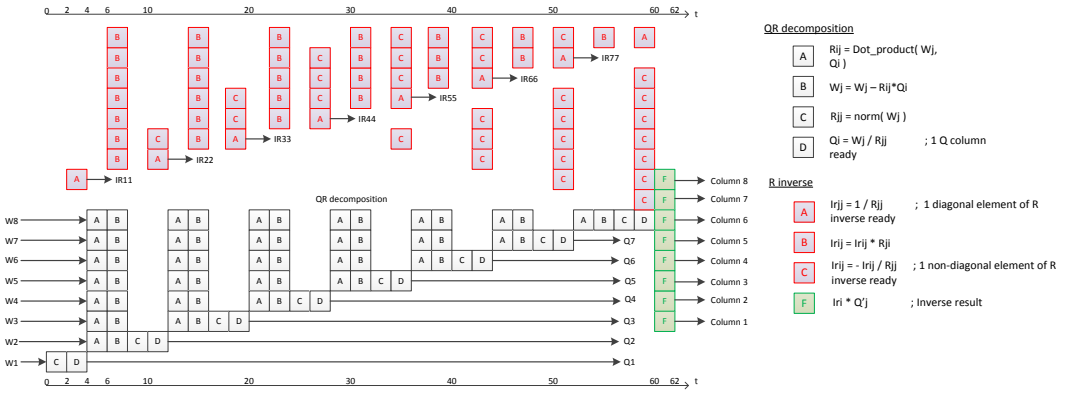


Fig. 11 Pipeline schedule for MGS-QR matrix inversion for 4x4 complex-valued matrices

The "preparation" function in Table 2 is not one of the principal blocks of the IEEE 802.11ac transmitter. However, this block was added to the chain for rearranging the bits in a way that it would accelerate the following operations. "Pilots" refers to the addition of the pilot subcarriers, and "phase rotation" indicates the rotation in tones, which for the 80 MHz bandwidth is only a multiplication with minus one for some of the subcarriers in each transmit chain. Furthermore, the original order of operations depicted in Fig. 3 is modified to speed up some functions such as STBC which is executed faster when done before modulation and at bit level. As it can be seen in Table 2, some of the blocks were merged together during the implementation to achieve a better performance.

According to Fig. 2, the duration of a DATA symbol with short GI is $3.6\mu s$. This means that all the related processing for a DATA symbol should take less than $3.6\mu s$ for the implementation to be considered real-time. Table 2 shows that the total number of cycles needed for the most complex case is less than 800 cycles. Assuming an operating frequency of 500 MHz, the last case would take $1.6\mu s$. It can be concluded that for all of these scenarios in transmitter a real-time operation is feasible at 500 MHz when using the Tensilica BBE32 core.

Table 3 presents the number of clock cycles needed for the processing of a DATA symbol at the receiver. As the RCPI measurement should be carried out during the DATA field processing, it is included in the same table here. "Pilot removal" refers to the operation where the pilot subcarriers are separated from the tones carrying data.

The total number of cycles presented in Table 3 shows that frequencies higher than 1 GHz may be needed for cases A and C for achieving a real-time operation.

Table 3 The number of clock cycles needed for the processing of a DATA symbol in the receiver

Functional Blocks	Case A	Case B	Case C	Case D
RCPI Variance	185	365	185	365
Pilot Removal	140	336	140	336
Symbol Detection	468	652	208	919
Tone De-mapper	324	672	324	672
Stream Deparser	74	254	-	148
Soft Bit Detection	3193	6504	3193	6504
Frequency Error Tracking	255	331	255	331
Total Number of Cycles	4639	9087	4305	9275

In these cases, as mentioned in Table 1, only two transmit antennas are present. However, with four antennas in cases B and D, the number of cycles consumed is approximately twice more than cases A and C. As a result, higher frequencies, in order of 2 GHz, are needed.

The block responsible for the high number of cycles is the soft bit detection. In the continuation of this work the soft bit detection implementation will be modified to accomplish better results in terms of the number clock cycles. This will be carried out by introducing instruction extensions for this bottleneck operation. This is one key topic for our future work.

The other complex field for processing in the receiver is the VHT-LTF field. The corresponding functions with the relating number of clock cycles for this field are presented in Table 4.

Table 4 The number of clock cycles needed for the processing of a VHT-LTF symbol in the receiver

Functional Blocks	Case A	Case B	Case B/ACC	Case C	Case D
LS Channel Estimation	281	289	289	281	2873
LMMSE Channel Estimation	1078	1107	1107	1078	1643
Detector Coefficients	2630	31713	548	851	34005
Total Number of Cycles	3989	33109	1944	2210	38521

The matrix inversion accelerator implemented for BBE32 presented in Section 5.2 is utilized for the inversions needed in calculating detector coefficients for case B. The results achieved when using the accelerator are presented in the column with the title Case B/ACC in Table 4. As it can be seen from this table, this accelerator has dramatically decreased the number of clock cycles in comparison with the case where the usual approach is used.

Based on Fig. 2, the duration of each VHT-LTF symbol defined in the VHT standard is $4\mu s$. Thus, with the accelerator using LUTs for the matrix inversion, a real-time operation for the VHT-LTF symbol can be achieved with frequencies near 500 MHz, whereas, higher frequencies are required without the application of this method in the implementation.

Another important aspect for evaluating the introduced software solution is the power consumed by the design. The profiling tools provided by the Xtensa Xplorer include an energy analyzer tool. This tool can be used for measuring the power consumption by dividing the energy numbers by time. The required time for each block is defined by its respective number of clock cycles. The amount of power consumed is directly dependent on the used memory capacity. For these measurements, maximum memory capacity (128k) is assumed, and the operating frequency is set to 500 MHz. The time for the energy analysis is $3.6\mu s$ and $4\mu s$ for the DATA and VHT-LTF symbols, respectively. Tables 5 and 6 represent the power consumed for a DATA symbol in the transmitter and receiver, respectively.

Table 5 Power consumption in mW for the processing of a DATA symbol in the transmitter

Functional Blocks	Case A	Case B	Case C	Case D
Preparation + STBC	1.8	-	3.7	2.3
Tone Mapper	5.2	5.2	5.2	5.2
Stream Parser + Constellation Mapper	5.1	6.6	5.1	9.4
Pilots + CSD + Phase Rotation + Spatial Mapping	4.8	9.6	4.8	9.6
Total Power Consumption	16.9	21.4	18.8	26.5

Table 6 Power consumption in mW for the processing of a DATA symbol in the receiver

Functional Blocks	Case A	Case B	Case C	Case D
RCPI Variance	3.6	6.67	3.35	6.16
Pilot Removal	4.76	10.39	4.76	9.52
Symbol Detection	13	23.51	6.82	34.65
Tone De-mapper	10.02	20.07	10.02	20.03
Stream Deparser	3.06	8.59	-	6.11
Soft Bit Detection	106.95	213.92	105.43	213.19
Frequency Error Tracking	4.46	7.55	4.44	7.55
Total Power Consumption	145.61	290.7	134.82	297.21

The total amount of power consumption presented in Tables 5 and 6 indicates, potentially, the feasibility of this software solution for mobile terminal scale devices.

6.2 Performance Comparison

As mentioned earlier, very few contributions have been made toward software based implementation of the IEEE 802.11ac. Thus, in this section, the performance results from some of the reported fixed-function solutions are presented. These numbers cannot be directly compared with the results presented in this paper, as there could be different functional blocks implemented using different algorithms. Moreover, the work presented in this article covers the frequency domain processing and not the complete transceiver.

In [12], an OFDM transceiver for the IEEE 802.11a is implemented which supports a 20 MHz bandwidth, up to 64-QAM modulation, and 3/4 coding rate. This specification does not require support for MIMO and requires less intensive computations compared to our work. The design is targeted for 180 nm TSMC technology and consumes 72 mW power in the whole transceiver.

An implementation of a MIMO-OFDM transceiver with 40 MHz bandwidth, two transmit and three receive antennas, up to 64-QAM modulation, and 3/4 coding rate is described in [13]. The ASIC employed in this work is fabricated in a TSMC 130nm CMOS process. This implementation consumes a total power of 362 mW in the transmitter and 502 mW in the receiver.

Similarly, in [14], an architecture for a MIMO-OFDM transceiver conforming to the VHT amendment is proposed. This work shares some of the specifications used in our work such as, 80 MHz bandwidth, 4×4 MIMO, and 256 OFDM subcarriers. However, it employs a 64-QAM modulation scheme rather than the 256-QAM scheme we have adopted. Synthesized on a 90 nm CMOS, this design consumes 117.7 and 496 mW power in transmitter and receiver, respectively.

7 Conclusion

This paper proposed a software based implementation for the IEEE 8802.11ac transmitter and receiver baseband processing on a DSP core. The frequency domain processing of the VHT PHY layer was implemented for four different MIMO scenarios. Then, to address the feasibility of a real-time operation on the Tensilica BBE32 core, the proposed software solution was evaluated in terms of number of clock cycles needed for the processing of a DATA and VHT-LTF symbol. The results showed the possibility of a real-time implementation with an operating frequency less than 500 MHz in the transmitter. In the receiver, by exploiting an accelerator for matrix inversion, the VHT-LTF symbol processing can be carried out in real-time with an operating frequency slightly above 500MHz. However, frequencies higher than 1 GHz for the 2 antenna configurations and higher than 2 GHz for the four antenna configurations are required to achieve a real-time processing of the DATA symbol on this platform. Moreover, the power consumption of the design was measured and the results showed the feasibility of exploiting the developed software on hand-held devices. The described solution will be improved in the future works with the aim of reducing the operating frequency required for a real-time implementation by introducing instruction extensions for the operations consuming high number of clock cycles, particularly the soft bit detection.

Acknowledgements This work was supported by the Finnish Funding Agency for Technology and Innovation (TEKES) under the Parallel Acceleration (ParallaX) project, the Graduate School of the Tampere University of Technology, and Nokia Foundation.

References

1. Tuttlebee W. (Ed.) (2004). *Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations.* 1^{st}ed. West Sussex: Wiley.
2. Grayver, E. (2013). *Implementing Software Defined Radio.* New York: Springer.
3. NVIDIA, NVIDIA SDR (Software Defined Radio) Technology, USA. (2013)
4. IEEE Std 802.11^{TM} Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. (2012).
5. IEEE P802.11ac^{TM} Draft standard, version 5. (2013).
6. Perahia, E., & Stacey, R. (2013). *Next Generation Wireless LANs.* 2^{nd}ed. Cambridge: Cambridge University Press.
7. Netgear Inc. (2012). *Next Generation Gigabit WiFi - 802.11ac*, USA.
8. QUALCOMM Inc. (2012). *IEEE 802.11ac: The Next Evolution of WiFi^{TM} Standards*, USA.
9. Tensilica Inc. (2012). *Connx BBE32 DSP User Guide*, USA.
10. Aghababaeetafreshi, M., Lehtonen, L., Soleimani, M., Valkama, M., Takala, J. (2014). IEEE 802.11ac MIMO transmitter baseband processing on customized VLIW processor. In *Proc. IEEE ICASSP*, Florence, Italy, 4-9 May (pp. 7550-7554).
11. Aghababaeetafreshi, M., Lehtonen, L., Levanen, T., Valkama, M., Takala, J. (2014). IEEE 802.11ac MIMO Receiver Baseband Processing on Customized VLIW Processor. In *IEEE Workshop on Signal Processing Systems (SiPS)*, Belfast, UK, 20-22 October (pp. 232-237).
12. Nagaraju, M., & Rakesh, M. (2012). High-speed and low-power ASIC implementation of OFDM transceiver based on WLAN (IEEE 802.11a). In *Proc. International Conference on Devices, Circuits and Systems*, Coimbatore, India, 15-16 March (pp. 436-439).
13. Son, J., Lee, I., & Lee, S. (2007). ASIC Implementation and Verification of MIMO-OFDM Transceiver for Wireless LAN. In *porc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communication*, Athens, Greece, 3-7 September (pp. 1-5).

14. Yoshizawa, S., & Miyanaga, Y. (2009). VLSI Implementation of a 4x4 MIMO-OFDM Transceiver with an 80-MHz Channel Bandwidth. In *proc. IEEE ISCAS*, Taipei, Taiwan, 24-27 May (pp. 1743-1746).
15. Yang, H., Shim, J., Bang, J., & Lee Y. (2014). Software-based giga-bit WLAN platform. In *proc. IEEE International Conference Consumer Electronics (ICCE)*, Las Vegas, USA, 10-13 January (pp. 478-479).
16. Li M., Amin A., Appeitans, R., Folens, A., Ahmad, U., Cappelle, H., Debacker, P., Hollevoet, L., Bourdoux, A., Raghavan, P., Antoine, D., & Van Der Perre, L. (2013). A C-programmable baseband processor with inner modem implementations for LTE Cat-4/5/7 and Gbps 80MHz 4x4 802.11ac. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, USA, 3-5 December (pp. 1222-1225).
17. Mei, B., Lambrechts, A., Verkest, D., Mignolet, J.Y. & Lauwereins, R. (2005). Architecture exploration for a reconfigurable architecture template. In *IEEE Design & Test of Computers* 22(2), 90-101.
18. Samadi, S., Golomohammi, A., Jannesari, A., Movahedi, M.R., Khalaj, B., & Ghammanghami, S. (2006). A Novel Implementation of the IEEE802.11 Medium Access Control. In *Proc. International Symposium on Intelligent Signal Processing and Communications (ISPACS)*, Yonago, Japan, 12-15 December (pp. 489-492).
19. Gutierrez-Agullo, J.R., Coll-Perales, B., & Gozalvez, J. (2010). An IEEE 802.11 MAC Software Defined Radio implementation for experimental wireless communications and networking research. In *Proc. IFIP Wireless Days*, Venice, Italy, 20-22 October (pp. 1-5).
20. Alamouti, S. M. (1998). A Simple Transmit Diversity Technique for Wireless Communications. *IEEE Journal on selected areas in Communication*, 16(8), 1451-1458.
21. Bartlett, M. S. (1951). An Inverse Matrix Adjustment Arising in Discriminant Analysis. *The Annals of Mathematical Statistics*, 22(1), 107-111.
22. Kumawat, S., Shrestha, R., Daga, N., Paily, R. (2015). High-Throughput LDPC-Decoder Architecture Using Efficient Comparison Techniques & Dynamic Multi-Frame Processing Schedule. In *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(5), 1421-1430.
23. Xie, Q., He, Q., Peng, X., Cui, Y., Chen, Z., Zhou D., Goto, S. (2011). A high parallel macro block level layered LDPC decoding architecture based on dedicated matrix reordering. In *IEEE Workshop on Signal Processing Systems (SiPS)*, Beirut, Lebanon, 4-7 October (pp.122-127).
24. Huang, S., Bao, D., Xiang, B., Yun, C., Zeng, X. (2010). A flexible LDPC decoder architecture supporting two decoding algorithms. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, 30 May - 2 June (pp.3929-3932).
25. Singh, C. K., Prasad, S. H., & Balsara, P. T. (2007). VLSI Architecture for MAtrix Inversion using Modified Gram-Schmidt based QR Decomposition. In *Proc. International Conference on VLSI Design*, Bangalore, India, January (836-841).
26. Tensilica Inc. (2013). *Connx BBE32 DSP Core for Baseband Processing*, USA.

PUBLICATION 4

M. Aghababaeetafreshi, J. Yli-Kaakinen, T. Levanen, V. Korhonen, P. Jääskeläinen, M. Renfors, M. Valkama and J. Takala, "Parallel processing intensive digital front-end for IEEE 802.11ac receiver," in *49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, 8-11 Nov, 2015, pp. 1619-1626, DOI: 10.1109/ACSSC.2015.7421422

Parallel Processing Intensive Digital Front-End for IEEE 802.11ac Receiver

Mona AghababaeeTafreshi, Juha Yli-Kaakinen, Toni Levanen, Ville Korhonen, Pekka Jääskeläinen, Markku Renfors, Mikko Valkama, and Jarmo Takala
Tampere University of Technology, P.O. Box 553, FI-33720 Tampere, Finland
email: mona.aghababaeetafreshi@tut.fi

***Abstract*—Modern computing platforms offer increasing levels of parallelism for fast execution of different signal processing tasks. In this paper, we develop and elaborate on a digital front-end concept for an IEEE 802.11ac receiver with 80 MHz bandwidth where parallel processing is adopted in multiple ways. First, the inherent structure of the 802.11ac waveform is utilized such that it is divided, through time-domain digital filtering and decimation, to two parallel 40 MHz signals that can be processed further in parallel using smaller-size FFTs and, e.g, legacy 802.11n digital receiver chains. This filtering task is very challenging, as the latency and the cyclic prefix budget of the receiver cannot be compromised, and because the number of unused subcarriers in the middle of the 80 MHz signal is only three, thus necessitating very narrow transition bandwidth in the deployed filters. Both linear and circular filtering based multirate channelization architectures are developed and reported, together with the corresponding filter coefficient optimization. Also, full radio link performance simulations with commonly adopted indoor WiFi channel profiles are provided, verifying that the channelization does not degrade the overall link performance. Then, both C and OpenCL software implementations of the processing are developed and simulated for comparison purposes on an Intel CPU, to demonstrate that the parallelism provided by the OpenCL will result in substantially faster realization. Furthermore, we provide complete software implementation results in terms of time, number of clock cycles, power, and energy consumption on the ARM Mali GPU with half precision floating-point arithmetic along with the ARM Cortex A7 CPU.**

Keywords*—*WLAN, IEEE 802.11ac, Multirate Filtering, Digital Front-End, Graphics Processing Units, Open Computing Language, Parallel Processing.

I. Introduction

Software-based implementations of radio transceiver digital front-end (DFE) and baseband (BB) processing stages are receiving increasing interest, due to substantially enhanced re-configurability and reduced time-to-market cycles, when compared to classical fixed-function digital hardware implementations [1][2]. Modern platforms have increased parallel computation capabilities due to the limits of improving performance by means of increasing the clock frequency. Multi-core processors and graphics processing units (GPU) along with programming standards such as OpenCL enable software developers to explicitly utilize the parallelism for faster processing [3].

In this paper, we address the DFE processing of the flagship WLAN/WiFi technology, namely IEEE 802.11ac [4], where the basic radio access is based on 80 MHz instantaneous bandwidth. Interestingly, this 80 MHz access waveform is composed by essentially aggregating two 40 MHz sub-signals [4], stemming from the legacy IEEE 802.11n access bandwidth, with three null subcarriers (approximately 1 MHz) inbetween. In the digital front-end concept proposed in this paper, this overall 80 MHz signal is divided to two 40 MHz sub-signals, through carefully optimized time-domain filtering, which in turn can then be processed forward in parallel, with two smaller-size FFTs and corresponding frequency-domain processing. This overall receiver principle, assuming also wideband I/Q downconversion from RF to baseband, is depicted at conceptual level in Fig. 1. This can be also extended to a 160 MHz signal being divided into four 40 MHz sub-signals from which each can be received by a modified 40 MHz 802.11n receiver. However, this filtering task is far from trivial, as the cyclic prefix (CP) budget of the overall wireless link, including filtering in the devices, should not be compromised, since the latency requirements of the 802.11ac receiver are very tight [4], and because the small spectral gap of around 1 MHz calls for very narrow transition bandwidth in the filter optimization. Hence, in this paper, we first address this channelization filter optimization task, and report both linear digital filtering and circular digital filtering based multirate solutions with different characteristics and tradeoffs related to latency, filtering performance, and CP budget. Essentially, the circular filtering based solution slightly increases the latency but does not compromise the CP budget at all, being implemented after the CP removal, just prior to the parallel FFT units. We also provide full radio link simulation results, with commonly adopted WiFi indoor channel models, to verify that the overall channelization filtering does not degrade the link performance.

Then, related to the actual software-based processing implementations, we have developed both C and OpenCL-based solutions on the Intel® Core™ i7-4800MQ CPU [5] to demonstrate that the explicit parallelism provided by the OpenCL framework will result in substantially faster execution. We also provide complete software implementation results using the Odroid XU3 [6]. The Odroid XU3 is based on the Samsung Exynos 5 Octa, powered by ARM Cortex™-A15 quad core and Cortex™-A7 quad core CPUs, which employs the ARM® big.LITTLE™ technology [7][8][9]. This technology creates a multicore processor which couples relatively slower processor cores with more powerful ones. The XU3 also features the ARM® Mali™-T628 MP6 GPU [10] with half precision floating-point arithmetic. Different filter designs are implemented and assessed in terms of execution time, number of clock cycles, power, and energy consumption.

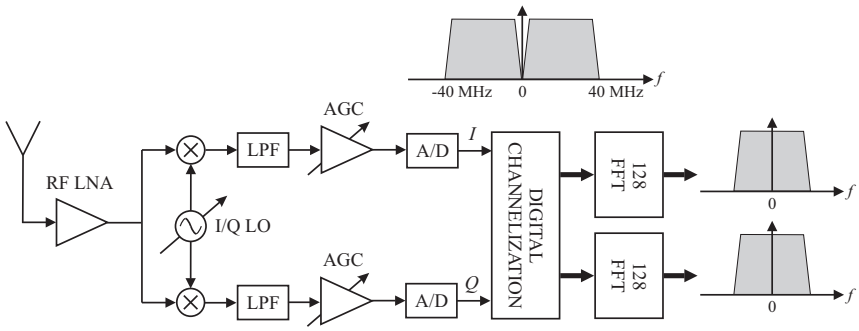


Fig. 1. The overall receiver principle with digital channelization filtering yielding two 40 MHz sub-signals.

The rest of the paper is structured as follows. First, in Section II, the channelization filtering architectures based on linear and cyclic half-band multirate filters, together with corresponding filter optimization, are described. Then, in Section III, we provide comprehensive link performance evaluations, with and without channelization filtering, to verify and demonstrate that the optimized filtering solutions reported in Section II do not essentially degrade the link performance in any way. In Section IV, the software implementation and OpenCL kernel designs are described. Finally, the results from the GPU and the two CPUs are reported in Section V, and Section VI concludes the work.

II. Channelization Filter Architectures for IEEE 802.11ac

In this work, 80 MHz access bandwidth in IEEE 802.11ac system consisting of 256 subcarriers is considered. 242 subcarriers out of the total 256 are active (234 for data and 8 for pilots). Three subcarriers around DC (subcarriers $-1, 0, 1$) are zero and both the negative and positive frequency components contain 121 transmission subcarriers (subcarriers $\pm k$ for $k = 2, 3, \ldots, 122$) [4]. In the IEEE 802.11 standards [11], the total multicarrier symbol duration is defined as 4 μs; 20 percent of this duration (800 ns) is the guard interval which carries the cyclic prefix of the signal. For FFT size of $L = 256$ this corresponds to the cyclic prefix of 64 samples. As described already in the Introduction, the goal is to divide the 80 MHz IEEE 802.11ac signal sampled at the Nyquist rate into two 40 MHz-wide signals using linear filtering such that the positive frequency components are separated into one signal and negative frequency components into a second. These are then processed further, in parallel, with two 128 point FFTs and subsequent subcarrier level processing.

A. Polyphase Halfband Filters

The problem stated above can be solved either using finite-impulse response (FIR) or infinite-impulse response (IIR) *analytical filters* [12], [13]. For FIR case, the analytical filter requiring minimum number of multiplier values can be derived with the aid of halfband filters. The transfer function of the halfband lowpass-highpass FIR filter pair can be realized efficiently as a parallel connection of *Type II* (odd-order symmetric) FIR transfer function $H(z^2)$ and a delay of M

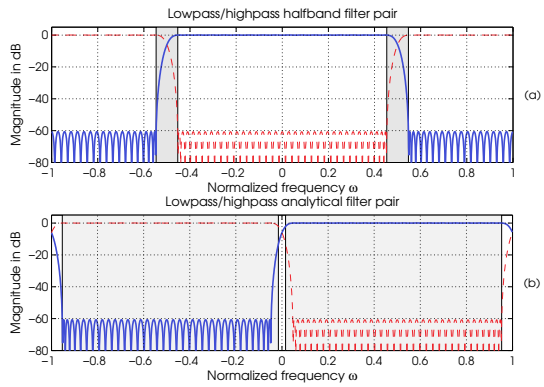


Fig. 2. Magnitude responses of the (a) halfband and (b) analytical filter pairs. In (a), the gray areas indicate the transition bands of the prototype filter pair. In (b), the gray areas indicate the 40 MHz sub-bands containing active subcarriers.

as expressed in [14].

$$G(z) = H(z^2) \pm 1/2z^{-M}. \tag{1}$$

Here, M is an odd integer such that the order of the overall transfer function $G(z)$ is $N = 2M$. The lowpass (highpass) filter is realized using the above transfer function with the plus (minus) sign.

The analytical filter is obtained from $H(z)$ by multiplying the impulse response values $h(n)$ by j^{-n}. This corresponds to shifting the frequency response of the filter by $\pi/2$. The parallel connection of the resulting Hilbert transformer (more precisely, the approximation of it) and a delay can be used for forming analytical signals, that is, for separating the positive and negative frequency components as desired. Fig. 2(a) and Fig. 2(b) show the magnitude response of the lowpass-highpass halfband and analytical filter pairs, respectively. The active subcarriers in Fig. 2(b) are denoted by the gray area. The resulting output signals can be decimated by two, if desired, by sharing the input samples into these two branch filters such that odd samples go to one branch and even samples to another. In this case, the branch filters [$H(z)$ and $1/2z^{-M/2}$] work at the output sample rate, that is, at the half of the input rate.

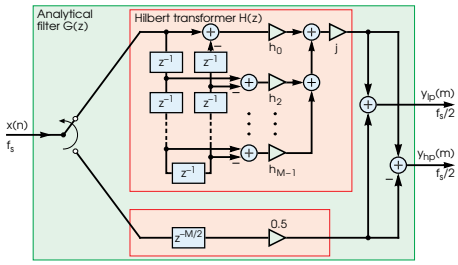


Fig. 3. Efficient processing structure of the decimating analytical filter realizing both the lowpass and highpass outputs $y_{\text{lp}}(m)$ and $y_{\text{hp}}(m)$, respectively.

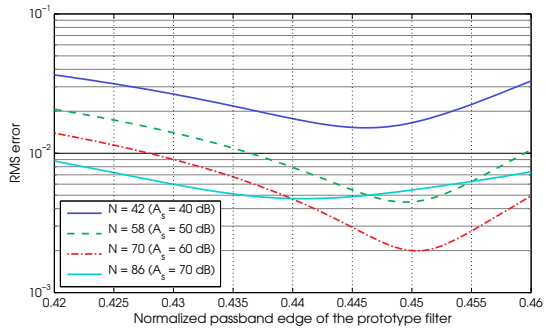


Fig. 4. RMS error between the received and the transmitted symbols as a function of passband edge for halfband FIR filters of order 42, 58, 70, and 86.

A more detailed structure of this analytical filter is shown in Fig. 3. A pair of these filters is required for filtering both the real and imaginary parts of the input signal.

When decimating the resulting lowpass and highpass filtered signals, the residue of the active negative (positive) subcarriers alias above positive (negative) subcarriers, i.e., subcarriers $-k$ for $k = 2, 3, \ldots, 122$ alias above subcarriers $128 - k$ for $k = 2, 3, \ldots, 122$. Consequently, the stopband edge of the lowpass analytical filter has to be $\omega_s = (128 - 122)/128\pi = 0.046875\pi$ to prevent aliasing into positive active subcarriers. Correspondingly, the passband and stopband edges of the prototype halfband filter are $\omega_p = 1/2\pi - (128 - 122)/128\pi = 0.453125\pi$ and $\omega_p = \pi - 0.53125\pi$ as the passband and stopband edges of the prototype halfband filter are located symmetrically around $\pi/2$ as $\omega_s = \pi - \omega_p$ for $\omega_p < \pi/2$ [12]. The gray areas in Fig. 2(a) denote the transition bands of the prototype filter pair.

The magnitude of the aliasing components is defined by the stopband attenuation of the prototype filter. Due to the properties of the prototype halfband filters, the order of the transfer function is restricted to be $N = 2 + 4k$, where k is an integer [12]. The performance of the analytical filter (parallel connection of Hilbert transformer and delay) is evaluated by measuring the root-mean-square (RMS) error of the received channelized signals as a function of the passband edge and filter length. In this simulation, the frequency response of the channelization filter is equalized per channel and 16-QAM subcarrier modulation with 1000 symbols are used. As can be seen from Fig. 4, the best RMS error performance is obtained using the filter of order $N = 70$ for which the passband edge is located at $\omega_p = 0.4505\pi$. The difference between the derived passband edge ($\omega_p = 0.453125\pi$) and the value obtained by simulation can be explained by the distribution of the zeros at the stopband. By optimizing the locations of the zeros in z-domain, their contribution to the attenuation at the exact carrier frequencies can maximized if desired. However, in real environment the difference would be negligible due to, e.g., the possible carrier-frequency offset.

B. Cyclic Polyphase Halfband Filters

The previous linear digital filtering based channelization increases the effective time dispersion of the received signal, and thus partially compromises the CP budget of the receiver. A straightforward way to tackle the increase in the overall impulse response length is to perform the channelization processing using cyclic convolution instead of linear convolution (conventional FIR filter). The basic idea is to carry out the linear convolution block-wise for the received data and then cyclically add the last N samples from the resulting N+L samples long sequence to the beginning of the block as depicted in Fig. 5. As a consequence, due to duality of cyclic convolution in time-domain and multiplication in frequency-domain, the effect of the channelization filter can be exactly equalized. Furthermore, as the cyclic convolution processing can be carried out *after removing the CP*, this solution does not contribute in any way to the effective time dispersion in the signal.

In this case, only the FFT size and the computational complexity restrict the length of the channelization filter. The computational complexity of cyclic realization is approximately 25 percent lower for the same filter order since the CP can excluded before channelization. It should be pointed out that the same polyphase filter channelization architecture can be used for both the linear and cyclic convolution.

III. 802.11AC Link Performance Evaluations

In order to verify that the overall channelization filtering does not degrade the 802.11ac link performance, extensive link simulations are carried out. Standardized WLAN/WiFi channel models D and F [15], [16], are used to simulate the link performance of the two proposed channelization architectures in the case of frequency selective fading channel. Table I shows the delay spread and cluster parameter values of these channel models. These two channel models can be considered to represent the environments with little-to-moderate frequency selectivity, as it is typically the case in indoor offices and houses (channel model D), and moderate-to-large frequency selectivity, common in large indoor spaces such as airport and conference centers (channel model F). The symbol error rate (SER) and error vector magnitude (EVM) performance of the channelization architectures are evaluated in two cases. In the first case, the performance is evaluated as a function of signal-to-noise ratio (SNR) whereas in the second case as a function of co-channel signal-to-interference ratio (SIR). For SNR simulation, the SER and EVM evaluation is carried out with both the perfect timing synchronization as well

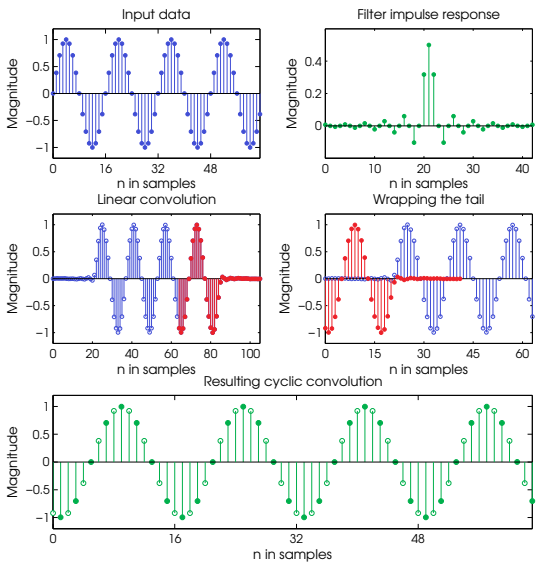


Fig. 5. Illustration of cyclic convolution using linear halfband filter.

as an example timing synchronization error of 8 samples. Three different prototype filters are used for the linear filter channelization case. The stopband attenuations of these filters are 40 dB, 50 dB, and 60 dB. For circular filter case, only one prototype filter is used with the stopband attenuation of 40 dB. In addition, SER and EVM performance is evaluated in the case with no channelization for reference purposes.

In the case of SIR simulations, the co-channel interference is a complex exponential having a random frequency inside the negative frequency band and the error functions are evaluated over the positive active subcarriers. In this case, only the perfect time-synchronization case is simulated. In all simulations, the number of random channel instances is 1000 whereas the number of 16-QAM modulated OFDM symbols is equal to 100.

The simulated SER and EVM as a function of SNR are shown in Fig. 6 whereas the corresponding SIR results are shown in 7. As can be seen from these figures, in the case of SNR simulation, the performance of the circular filter architecture is approximately the same as with no channelization. In the case of SIR simulation, the linear filter architecture slightly outperforms the circular filter with Channel model D, whereas in the case of Channel F, the circular architecture results in considerably better SER and EVM values. This is because the circular filtering based channelization architecture does not reduce the CP budget of the receiver in any way. However, as the adjacent channel rejection in IEEE 802.11 RF front-ends should be at least 40 dB, the SIR performance of the circular filter also with little-to-moderate frequency selectivity of the Channel D can be considered to clearly meet the requirements.

TABLE I. DELAY SPREADS AND CLUSTER PARAMETERS OF INDOOR TGN AND TGAC SPATIAL CHANNEL MODELS [15], [16]

Model	Scenario	RMS delay spread	Number of clusters	Taps/cluster
D	Indoor typical office	50 ns	3	16,7,4
F	Large indoor space	150 ns	6	15,12,7,3,2,2

IV. SOFTWARE IMPLEMENTATION

Three different platforms were employed for the implementation of the channelization task described in the previous sections. Firstly, C and OpenCL implementations were carried out on the Intel® Core™ i7-4800MQ CPU which has 4 cores and runs at base frequency of 2.7GHz and turbo frequency up to 3.7GHz. This step was done with the purpose of demonstrating the speedup achieved as a result of using OpenCL compared to C. Then, to take advantage of the parallel computing ability of GPUs, the implementation was additionally carried out on the ARM® Mali™-T628 MP6 GPU [10]. Mali-T628 is a part of the Samsung Exynos 5 Octa (Exynos 5422) mobile System on Chip (SoC). Mali-T628 offers scalability from one to eight cores and runs at a frequency of 600 MHz. This GPU also provides support for half precision floating point arithmetic. Half-precision floating numbers are defined by the IEEE 754 standard to have 16 bits consisting of five bits for the exponent, 10 bits for the fraction and one bit for the sign [17]. The usage of half floats could possibly reduce the execution time, power, and energy consumption to some extent. Exynos 5422 is also equipped with two CPUs using the big.LITTLE heterogeneous computing architecture [9]. The two CPUs are ARM® Cortex®- A15™ and A7™ [7][8]. A15 and A7 are quad core CPUs and can run at up to 2.1GHz and 1.5GHz, respectively. The ARM big.LITTLE architecture aims at achieving high performance while improving the power efficiency by coupling a performance driven "big" core with a power efficiency driven "LITTLE" core. Thus, these CPUs were also taken into consideration for the implementation. In this work, we have used ODROID-XU3 [6] to utilize the Samsung Exynos 5422 SoC for the implementation.

Various approaches were considered to implement the channelization filter in OpenCL, designed carefully to utilize the available parallelism. Two of the approaches which proved to be most efficient are described in the following. Different filter designs and software implementations are considered in each solution. The first solution focuses on a halfband filter design with higher length and lower number of required arithmetic operations, due to the zero coefficients, compared to the second solution. The second one, however, uses a non-halfband filter design with shorter length and utilizes vector operations. These implementations are carried out for both linear and cyclic filter designs.

A. Halfband Filter Without Vectorization

First approach implements the linear and cyclic halfband filters described in Section II. This implementation takes advantage of the fact that every other coefficient in the filter design is zero (as illustrated in Fig. 5), thus reducing the number of required multiplications by a factor of two. Moreover, having symmetric coefficients helps simplify the implementation further by first subtracting the pair of samples having the same coefficient values and then multiplying the resulting difference

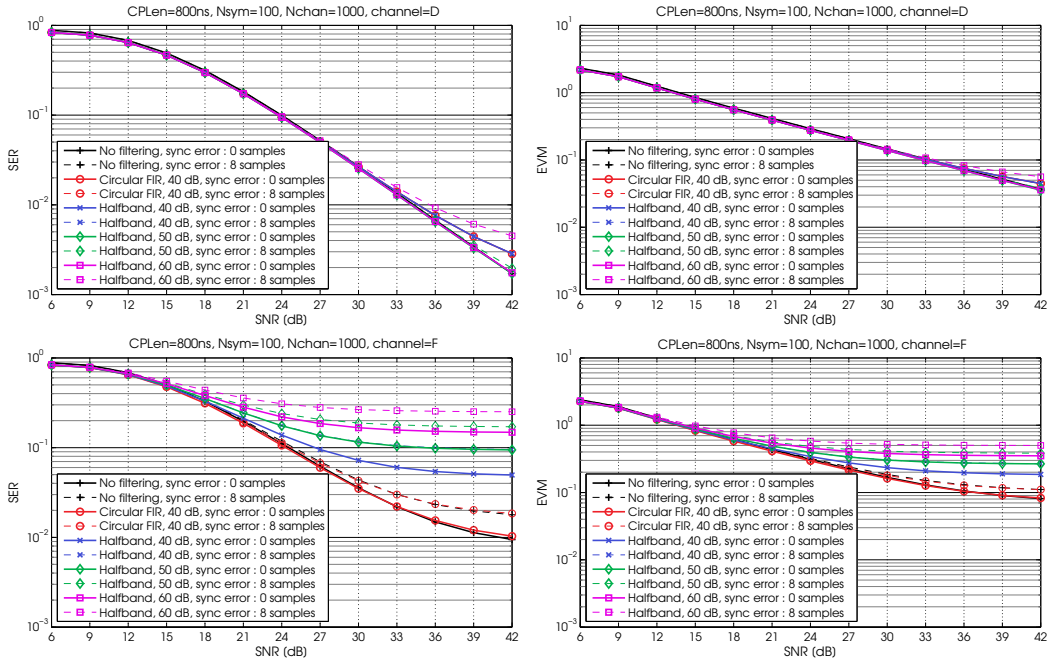


Fig. 6. SER and EVM as a function of SNR for conventional and circular halfband filters with channel models D and F.

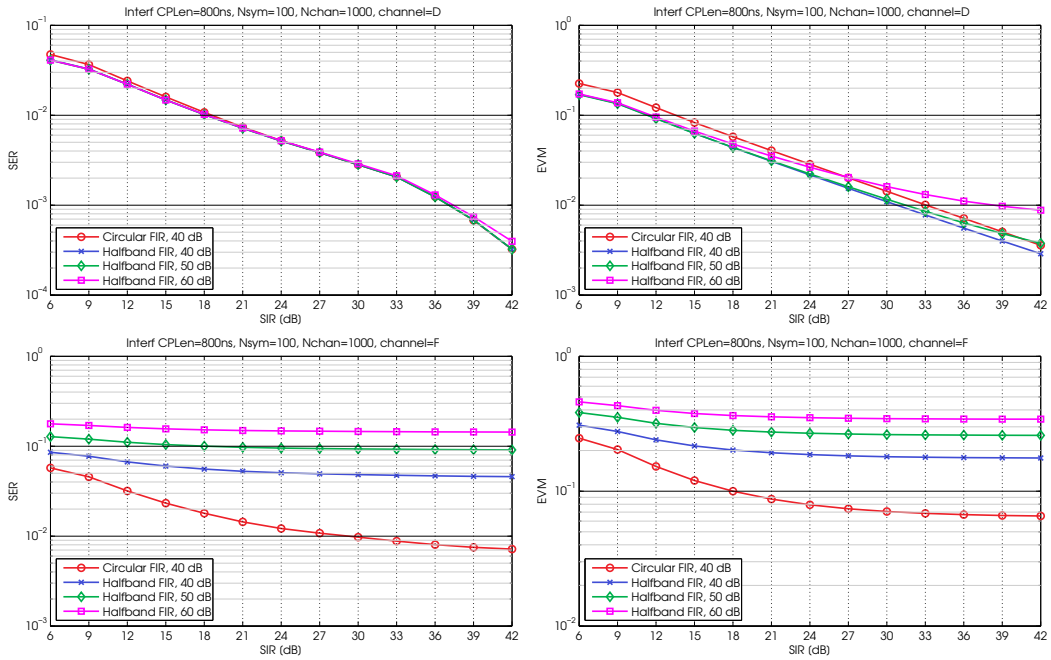


Fig. 7. SER and EVM as a function of SIR for conventional and circular halfband filters with channel models D and F.

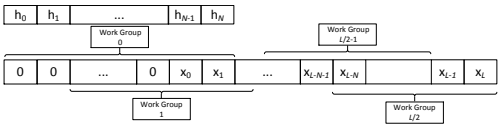


Fig. 8. The structure for the implemented halfband kernel, x denotes input samples, h denotes filter coeficients, L is the number of input samples, and N is filter order.

only once. Also both the lowpass and highpass filters can be realized at the same cost. In this implementation, it is assumed that all the input samples of one OFDM symbol and the filter coefficients are fed to the kernels input buffers. Fig. 8 illustrates the distribution of the computations among work groups, having L and N representing the number of subcarriers in one OFDM symbol and the filter order, respectively. To start the parallel computations, it is assumed that $N+L-1$ samples are stored in the input buffer, $N-1$ of which are zeros, added at the beginning of the input stream. As shown in Fig. 8, in this implementation, $L/2$ work groups work simultaneously to multiply the vector coefficients with the input samples and do the summations. Consequently, each work group produces one lowpass and one highpass output at the same time with other work groups.

B. Non-halfband Filter with Vectorization

In software based solutions, an important aspect to consider aside from the number of arithmetic operations is the memory accesses. In case of a halfband filter implementation, having zero coefficients, every other sample is skipped, which reduces the number of required multiplications. However, having these erratic memory accesses in the halfband filter could be less efficient than executing all multiplications instead of half. Thus, in the second approach, a non-halfband filter is considered for the channelization. As the cores support Single Instruction Multiple Data (SIMD) operations, an efficient implementation for the filter could be carried out using the OpenCL vector operations. The highest number of allowed vector components in OpenCL is 16. For this reason, the optimum designed filter should have a length that is multiple of 16. In general, the realization of odd-order (even length) two-channel FIR filter bank is not desirable. This is due to the reason that the resulting polyphase branch filters are non-symmetric filters, resulting in a quadruple complexity compared with the original half-band design. Therefore, the filter length is chosen to be $16n-1$. Then the length has been increased to $16n$ by padding one zero at the end of the impulse response.

In this kernel design, input samples and filter coefficients should be in the form of vectors of length 16. Fig. 9 depicts the arrangement of work groups and work items in this implementation. S is the number of subcarriers in one OFDM symbol plus N zeros added for filtering. $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{S/16}$ are vectors of length 16 containing S samples altogether. As it is illustrated in Fig. 9, each work group operates on a number of vectors. Then inside each work group, each work item, according to its work item number, carries out the processing related to a part of the vectors corresponding to that work group. This processing includes the multiplication of the data samples by coefficient values and the final summation.

V. Results and Analysis

To evaluate the performance enhancement achieved by exploiting parallelism using OpenCL, we have measured the execution time and number of clock cycles consumed when executing the filters both using C and OpenCL. This preliminary step was carried out on the Intel® Core™ i7. Then to study the advantages and disadvantages of different multicore platforms, the channelization filter was additionally implemented on the ARM® Mali™-T628 and the ARM® Cortex®- A7™ CPU. In addition to time and number of clock cycles, power and energy consumption were measured on the ARM platforms using the sensors available on the Odroid XU3. Most importantly, the performance improvements obtained by the application of half precision floating point arithmetic on Mali was investigated and is presented in this section. In all the measurements, the number of input samples is equal to the FFT size plus the CP length and the filter order, all multiplied by two as all the samples are in complex form.

A. Execution Time

Fig. 10 shows the execution times in milliseconds for running linear and circular filters using halfband and non-halfband implementations on the different platforms introduced in Section IV. Firstly, Fig. 10 shows that the halfband filter is executed approximately 80% faster when using OpenCL rather than C. Furthermore, it can be seen that among the OpenCL implementations, the Intel Core i7 consumes the least time. The second fastest platform is the Mali GPU, and the slowest is the ARM A7 CPU. This can be explained by the Intel CPU having the highest clock frequency, up to 3.7GHz which is six times higher than Mali's and two times higher than A7's. Another important observation from the implementation results is the amount of speedup gained by using half precision floats on the Mali GPU. The results show that the application of half precision floats has lowered the execution time by at least 55% which exceeds the expected linear speedup of two. This could be explained by the fact that taking up less space for the data results in more cache hits and less memory transfers, thus causing the faster execution. As it can be seen from Fig. 10, there is less difference between the linear and circular filtering solutions in non-halfband implementations as the designed non-halfband linear and circular filters have the same filter length. However, with the halfband implementation, the circular design requires a higher filter length, thus resulting in relatively slower execution.

The latency restrictions for this channelization process originate from the duration of the defined short interframe space (SIFS) in the IEEE 802.11ac amendment. As this channelization task is carried out for 80 and 160 MHz bandwidths which are only available in 5GHz carrier, the available SIFS time is equal to $16\mu s$. The lowest possible execution time realized on the platforms used in this work is $6.02\mu s$. Taking into consideration the other related required processing, such as MAC processing, the filtering can fit in the time frame. However, to have better margins for the rest of the required processing, it is beneficial to still reduce the execution time further. Mali is a small mobile GPU and employing a faster, larger GPU can result in lower execution times for the filtering that can, more easily, meet the real time requirements. Thus,

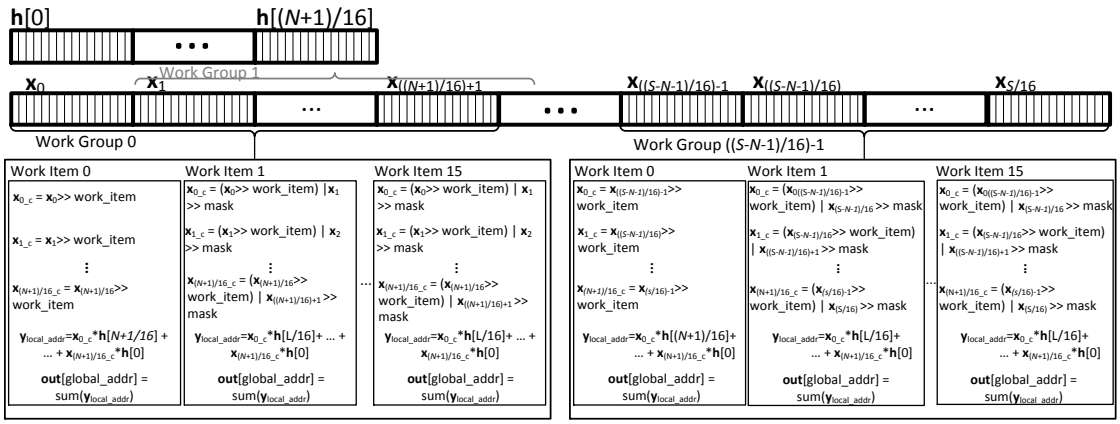


Fig. 9. The structure for the implemented non-halfband kernel, **x** denotes input sample vectors, **h** are the vectors containing filter coefficients, N is the filter order, and S is the number of input samples plus N zeros.

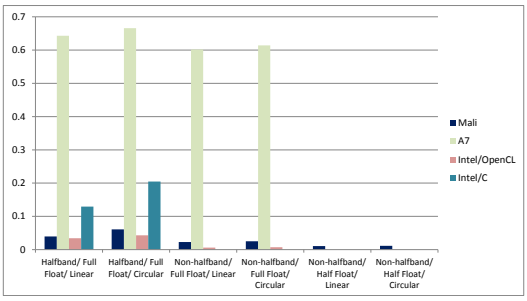


Fig. 10. Execution time in miliseconds consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

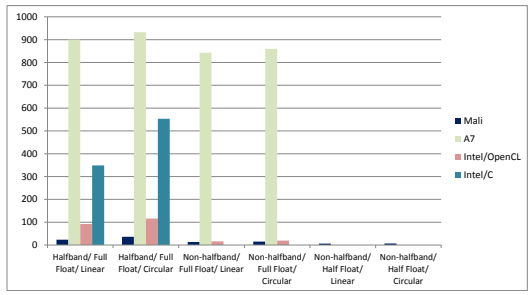


Fig. 11. Number of clock cycles consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

this could be also considered as a suitable implementation candidate e.g. in an access point setting using larger GPUs.

B. Number of Clock Cycles

To calculate the number of clock cycles required for each filter implementation, the nominal frequency of the platforms was assumed. The clock frequencies considered for the Intel CPU, ARM CPU, and the Mali GPU were 2.7GHz, 1.4GHz, and 600 MHz, respectively. The calculated number of clock cycles are presented in Fig. 11. Similar to the execution times presented in the previous section, these numbers, most importantly, verify the great advantage of using half precision floats over the full precision floats.

C. Power

The Odroid is equipped with four separated current sensors to measure the power consumption of Big CPU (A15), Little CPU (A7), GPU and DRAM in real time. In this work, the measurements are carried out in a way that 200 samples are taken from the sensors in intervals of 100ms and then averaged over a 20s time period to assess the average power consumption. To achieve more precise measurements, the kernels were run in high number of iterations to keep the cores active with kernel executions during the whole 20s. We did not have any tools available to measure the power consumption of the Intel CPU. The power consumed by Mali and A7 in different scenarios are presented in Fig. 12. It can be seen that the relatively lower power, lower performance Little CPU, A7, consumes less power than the GPU. Moreover, the application of half precision floating points has reduced the power consumption by approximately 33%.

D. Energy

While it is important to evaluate power consumption for heating matters, energy consumption, specifically in mobile applications, plays a very important role, as it translates to battery life. Fig. 13 illustrates the calculated energy consumption in different implementations by both the GPU and the CPU. As it is shown in this figure, employing half precision floating points has resulted in almost 60% reduced energy consumption in comparison with the case with full precision floating points. This is due to the reason that kernel execution with half floats is carried out in more than half of the time and with almost half power as the full floats. Although A7 is a low power CPU, the much lower kernel execution times on Mali has resulted

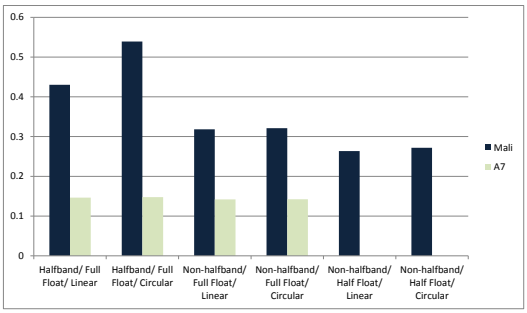


Fig. 12. Power in watts consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

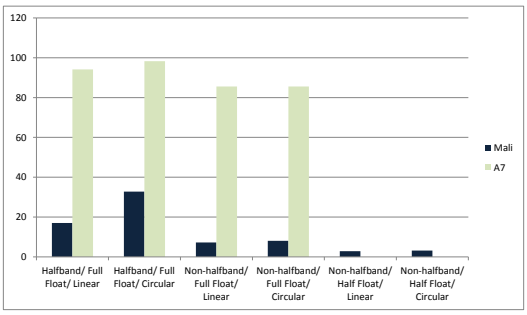


Fig. 13. Energy in µJ consumed by linear and circular digital filtering using halfband and non-halfband designs on different platfroms.

in overall lower energy consumption by the GPU.

VI. Conclusion

In this paper, we addressed the digital front-end processing of the IEEE 802.11ac receiver, targeting software-based processing implementation with substantially increased level of parallelism for fast execution. First, the overall 80 MHz received waveform is divided to two 40 MHz-wide signals through time-domain digital filtering so that the two 40 MHz signals can be then processed in parallel. We have optimized the channelization filter realizations and reported the results for both the linear and circular digital filtering. Then, the overall 802.11ac radio link was simulated, incorporating the developed channelization filter architectures, with two different WLAN/WiFi channel models. The SER and EVM performance of the channelization architectures were evaluated, showing that the link performance is not degraded by these filtering solutions. Finally, actual software implementations were carried out for linear and circular digital filtering using both halfband and non-halfband designs on different platforms, namely the Intel® Core™ i7-4800MQ CPU, ARM® Cortex®- A7™, and ARM® Mali™-T628 MP6 GPU. All filter designs were evaluated in terms of execution time and number of clock cycles on all three platforms, and power and energy consumption on the Mali GPU and A7 CPU. Comparing the OpenCL and C implementations revealed that exploiting parallelism using OpenCL yields a five times faster execution. The results also demonstrated that the high performance Intel CPU and the Mali GPU executed the filtering tasks much faster. Moreover, while the power efficient ARM A7 consumes less power than Mali, having very short execution times resulted in Mali consuming much lower energy. Taking advantage of half precision floating points on Mali reduces the execution time, number of clock cycles, power, and energy to a great extent. The measured execution times also showed that the designs can marginally meet the latency requirements for the IEEE 802.11ac. However, the filtering can more easily satisfy the restrictions by employing higher performance GPUs or CPUs.

Acknowledgment

This work was supported by the Finnish Funding Agency for Technology and Innovation (Tekes) under the Parallel Acceleration (ParallaX) project, Tampere University of Technology graduate school, and Nokia Foundation.

References

[1] W. Tuttlebee (Ed.), *Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations*. 1^{st}ed. West Sussex: Wiley, 2004.

[2] E. Grayver, *Implementing Software Defined Radio*. New York: Springer, 2013.

[3] *The OpenCL specification*, The Khronos Group Inc., 2011. [Online]. Available: https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf

[4] *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, IEEE Standard 802.11ac-2013, Dec. 2013.

[5] *Intel® Core™ i7 Processor Family for LGA2011 Socket*, Intel Corporation, 2014.

[6] Hardkernel co., Ltd. ODROID-XU3. Available: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127&tab_idx=1

[7] *Cortex-A15 Technical Reference Manual*, ARM, 2011. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0438c/DDI0438C_cortex_a15_r2p0_trm.pdf

[8] *Cortex-A7 MPCore Technical Reference Manual*, ARM, 2011, 2012. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0464d/DDI0464D_cortex_a7_mpcore_r0p3_trm.pdf

[9] *big.LITTLE Technology: The Future of Mobile, Making very high performance available in a mobile envelope without sacrificing energy efficiency*, ARM, 2013.

[10] *The ARM® Mali™ Family of Graphics Processors*, ARM, 2013.

[11] *IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-2012, 2012.

[12] H. W. Schüßler and P. Steffen, "Halfband filters and Hilbert transformers," *Circuits, Syst., Signal Process.*, vol. 17, no. 2, pp. 137–164, 1998.

[13] R. Ansari, "IIR discrete-time Hilbert transformers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1116–1119, Aug. 1987.

[14] T. Saramäki, "Finite impulse response filter design," in *Handbook for Digital Signal Processing*, S. K. Mitra and J. F. Kaiser, Eds. New York: John Wiley and Sons, 1993, ch. 4, pp. 155–277.

[15] *TGn Channel Models*, IEEE Standard 802.11-03/940r4, 2004. [Online]. Available at: https://mentor.ieee.org/802.11/dcn/03/11-03-0940-04-000n-tgn-channel-models.doc

[16] *TGac Channel Model Addendum*, IEEE Standard 802.11-09/0308r12, Dec. 2010. [Online]. Available at: https://mentor.ieee.org/802.11/dcn/09/11-09-0308-12-00ac-tgac-channel-model-addendum-document.doc

[17] *IEEE standard for floating-point arithmetic*, IEEE standard 754-2008, Aug.29, 2008.

PUBLICATION 5

M. AghababaeeTafreshi, M. Koskela, D. Korpi, P. Jääskeläinen, M. Valkama and J. Takala, "Software defined radio implementation of adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio," in *IEEE Global Conference on Signal and Information Processing*, Washington, DC, USA, 7-9 Dec, 2016, pp. 733-737, doi: 10.1109/GlobalSIP.2016.7905939

SOFTWARE DEFINED RADIO IMPLEMENTATION OF ADAPTIVE NONLINEAR DIGITAL SELF-INTERFERENCE CANCELLATION FOR MOBILE INBAND FULL-DUPLEX RADIO

Mona AghababaeeTafreshi, Matias Koskela, Dani Korpi, Pekka Jääskeläinen,
Mikko Valkama, and Jarmo Takala
Tampere University of Technology, P.O. Box 553, FI-33720 Tampere, Finland

ABSTRACT

Inband full-duplex radio transceivers offer enhanced spectral efficiency by transmitting and receiving simultaneously at the same frequency. However, deployment of such systems is challenging due to the inherent self-interference stemming from coupling of the transmit signal to the receiver. Furthermore, to track changes in the time-varying self-interference channel, the process needs to be self-adaptive. Thus, advanced solutions are required to efficiently mitigate the self-interference. With the current rise in parallel architectures due to limitations of performance enhancement by higher clock frequencies, multi-core platforms are considered as viable solutions for implementing such advanced techniques. This paper describes a programmable implementation of an adaptive nonlinear digital self-interference cancellation method for full-duplex transceivers on two mobile GPUs and a multi-core CPU. The results demonstrate the feasibility of realizing a real-time software-based implementation of digital self-interference cancellation on a mobile GPU, in case of a 20 MHz cancellation bandwidth.

***Index Terms*—** 5G, Full-duplex, self-interference cancellation, graphics processing units, open computing language

1. INTRODUCTION

Inband full-duplex communications provide a novel solution toward more spectral efficient networks. Systems utilizing such communications fully exploit the spectral and temporal resources by transmitting and receiving concurrently at the same frequency. With the expected increase in the throughput of future wireless systems, especially in the upcoming 5G networks, inband full-duplex communications can play a crucial role by improving spectral efficiency [1]. Employing such systems, throughput can be possibly increased by a factor of two, as the bandwidth can be used simultaneously for both transmission and reception [1]. However, deployment of full-duplex networks is far from trivial. This is due to the fact that simultaneous transmission and reception at the same frequency results in overlapping of the powerful transmit signal with the received signal of interest, thus producing strong self-interference (SI). This SI signal can be theoretically removed by subtracting the originally known transmitted signal from the received waveform. However, in practice, the signal will be both linearly and nonlinearly distorted while propagating to the receiver. This is a result of the nonlinear amplifiers, in-phase/quadrature (I/Q) imbalance of the transmitter and receiver, phase noise of the local oscillator, and analog-to-digital converter (ADC) quantization noise [2]. Consequently, effective cancellation of the SI signal becomes a challenging task. Aside from the aforementioned generic SI cancellation challenges, the task is even more challenging in the mobile device side compared to the base station side. Firstly, as low-cost components are more commonly used in mobile devices, nonlinear distortion becomes an especially critical issue. Secondly, due to limitations in power consumption, area, and processing complexity in mobile devices, less sophisticated and less computationally intensive methods are required on the mobile side. For this reason, in some of the earlier works, it was assumed that only the base station would be full-duplex compatible and the mobile device would work in half-duplex mode [3]. However, this would result in lower throughput compared to a system where full-duplex operation is also employed on the mobile side. Thus, in this paper we focus on the implementation of a method suitable for mobile devices using commercial off-the-shelf (COTS) low-cost components, while maintaining the required suppression of the SI signal. If proved feasible, real-time implementation on COTS components will eliminate the risky and costly custom hardware design efforts.

The proposed SI canceller implementation is based on software defined radio (SDR) solutions, which introduce flexibility compared to traditional fixed-function platforms. Although such implementations may not result in as low power and area as the conventional implementations, e.g. fixed-function hardware accelerators, they require less design efforts, and offer shorter time-to-market cycles. In addition, as increasing clock frequency for better performance is reaching its limits, parallel processing especially on graphics processing units (GPU) has gained a lot of interest. Furthermore, Open Computing Language (OpenCL) provides a framework for parallel computing on heterogeneous platforms. Thus, utilizing OpenCL and the available parallel resources in multi-core processors and GPUs, this work proposes a software implementation for SI cancellation in full-duplex systems, applicable on both the network side and the mobile station side. Here, three multicore platforms have been used and compared, namely, Intel® Core™ i7-4800MQ, Qualcomm® Adreno™ 430, and ARM® Mali™-T628 MP6 GPU [4] [5] [6]. Furthermore, the implemented algorithm is evaluated with measured signals from a true full-duplex RF test-bench, to demonstrate that it can attenuate the SI signal and fulfill the real-time constraints. There have been several contributions towards solving the SI issue in full-duplex systems in the recent years, such as the works reported in [3], [7], [8]. Additionally, several prototypes have been built to demonstrate the advances made in this regard as presented in [2], [9], and[10]. However, to the best of the authors' knowledge, no real-time hardware or software implementation for digital SI cancellation has been reported in the literature.

The rest of the paper is organized as follows. Section 2 shortly introduces the overall full-duplex transceiver model and the adaptive nonlinear digital SI cancellation algorithm. Section 3 provides a brief introduction to the selected platforms in addition to a description of the algorithm's OpenCL implementation. Then, in Section 4, real-time implementation results are presented. Finally, in Section 5, conclusions are drawn.

This work was supported by Tampere University of Technology graduate school, and the industrial research fund of Tampere University of Technology by Tuula and Yrjö Neuvo.

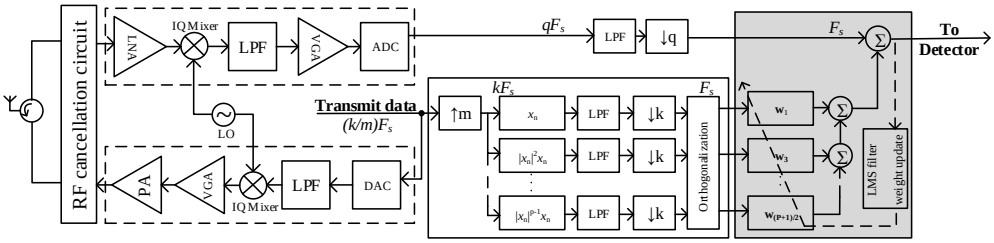


Fig. 1: Overall structure of the full-duplex transceiver, where the grey part is implemented in software

Algorithm 1 LMS-based adaptive nonlinear digital cancellation

Initialize $\mathbf{w}$ to 0, and n to L_{post}
while transmitting **do**
  $\mathbf{u}(n) = \left[\tilde{\mathbf{u}}(n+L_{pre})^{\mathrm{T}} \dots \tilde{\mathbf{u}}(n-L_{post})^{\mathrm{T}}\right]^{\mathrm{T}}$
  $e(n) = r_x(n) - \mathbf{w}(n)^{\mathrm{H}}\mathbf{u}(n)$
  if $(n \mod N == 0)$ **then**
    $\mathbf{w}(n+1) \leftarrow \mathbf{w}(n) + \boldsymbol{\mu} e^*(n)\mathbf{u}(n)$
  end if
  $n \leftarrow n+1$
end while

2. SELF-INTERFERENCE CANCELLATION IN FULL-DUPLEX SYSTEMS

To effectively mitigate the SI signal, the cancellation process is carried out in two stages, namely, RF and digital cancellation [9]. Figure 1 illustrates the overall structure of the full-duplex transceiver with the digital cancellation block, which is the focus of this paper, shown in more detail. As previously mentioned, the transmitter and receiver paths contain many non-ideal components, especially in mobile devices. However, as the transmitter power amplifier (PA) most significantly contributes to the nonlinear distortion, the SI signal can be modelled under the assumption that it is solely distorted by the PA [9]. Thus, using the well-known parallel Hammerstein model for highly nonlinear PAs, the observed SI signal, with respect to the original transmit signal, can be written as [9]:

$$r_x(n) = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} h_p(l) u_p(x(n-l)) + z(n), \qquad (1)$$

where P is the highest nonlinearity order of the modelled PA, L is the memory length of the model, $h_p(l)$ represents the overall p^{th} order effective SI channel coefficients, $x(n)$ is the baseband transmit signal, $u_p(x(n)) = |x(n)|^{p-1}x(n)$ is the pth order basis function, and $z(n)$ represents noise and possible model mismatch. The accuracy of this model depends on accurate estimation of the effective SI channel coefficients. Furthermore, as a result of the continuously changing environment around a mobile device, the channel coefficients need to be adaptively estimated. On the other hand, due to limited computational resources on a hand-held mobile device, a low complexity parameter learning and tracking algorithm is preferred. In [9], such an algorithm based on least mean squares (LMS) learning [11] is proposed. This algorithm ensures the accuracy of SI channel coefficients using a novel basis function orthogonalization procedure which is described in detail in [9]. This procedure should be performed prior to the actual LMS algorithm. The orthogonalized basis functions result in more accurate SI suppression. Now, the signal after the digital canceller can be written as:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_{p,ort}(l) \tilde{u}_p(x(n-l)) \approx z(n), \qquad (2)$$

where $\tilde{u}_p(x(n))$ contains the transformed orthogonalized basis functions, and $\hat{h}_{p,ort}(l)$ represents the corresponding SI cancellation coefficients. With a precise estimation of the coefficients, the cancellation signal should be sufficiently accurate for only $z(n)$ to remain after digital cancellation.

The low complexity LMS-based method used in [9], which adaptively estimates the SI channel coefficients, is described in Algorithm 1. This algorithm is modified to adjust the frequency of filter weight updates. Here, $L = L_{pre} + L_{post}$ is the length of the channel filter, where L_{pre} and L_{post} represent the pre-cursor and post-cursor taps, respectively. N defines how often the filter weights are updated and vector $\boldsymbol{\mu}$ contains the step sizes which are selected differently for each nonlinear term in the received signal [9]. Furthermore, $\tilde{\mathbf{u}}(n)$ represents the orthogonalized basis functions, $r_x(n)$ is the observed signal, $e(n)$ represents the cancelled signal, and $\mathbf{w}$ is defined as:

$$\mathbf{w} = \left[\hat{h}_{1,ort}(0) \dots \hat{h}_{P,ort}(0)\, \hat{h}_{1,ort}(1) \dots \hat{h}_{P,ort}(L-1)\right]^{\mathrm{T}}. \qquad (3)$$

3. ALGORITHM IMPLEMENTATION

3.1. Platforms

Three multi-core platforms have been chosen for implementing the SI cancellation algorithm. The first one is a desktop CPU, the Intel® Core™ i7-4800MQ, which has four cores [4]. This processor runs at a base frequency of 2.7 GHz and can run at up to 3.7 GHz [4]. The second platform is a mobile GPU, the Qualcomm® Adreno™ 430, which comes built in the Qualcomm® Snapdragon™ 810 system on chip (SoC) and can have a maximum clock speed of 500, 600, or 650 MHz [6]. The Snapdragon 810 is currently used in many of the hand-held devices in the market, and thus it can be a realistic candidate for GPU processing on mobile devices and provide actual, reliable results. The third one is the ARM® Mali™-T628 MP6 GPU, which is available on the Odroid XU3 board [12]. This GPU has four cores and can run at up to 600 MHz clock frequency [5]. Mali-T628 is a part of the Samsung Exynos 5 Octa (Exynos 5422) mobile SoC, which is a commercial product. Thus, Mali can also be considered as a practical candidate for mobile processing.

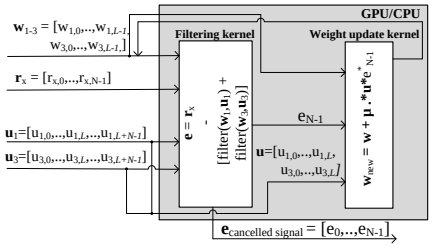


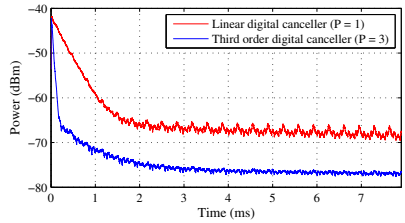
Fig. 2: Implemented kernel structure and data flow, where $P = 3$ is the highest nonlinearity order, $L = L_{pre} + L_{post} + 1$ is the channel filter length, N is number of samples processed in parallel before updating the SI channel coefficients, $\mathbf{w}_p$ contains the filter coefficients corresponding to the pth nonlinearity order, $\mathbf{r_x}$ is the vector comprised of the received signal samples, $\mathbf{u_p}$ represents the pth order orthogonalized basis function samples, $\mathbf{e}$ is a vector of produced cancelled signal samples, and $\boldsymbol{\mu}$ contains the step sizes.

3.2. Digital canceller implementation

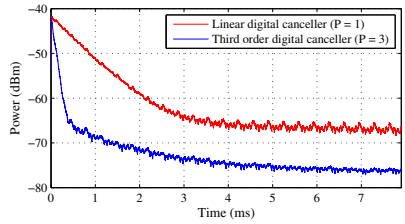
The implementation developed in this work carries out the adaptive digital self-interference cancellation in two steps using two OpenCL kernels. The structure and the data flow of the implemented kernels are illustrated in Fig. 2, where the highest nonlinearity order P is assumed to be three, $L = L_{pre} + L_{post} + 1$ is the channel filter length, N is the number of samples processed in parallel before updating the SI channel coefficients, $\mathbf{w}_p$ contains the filter coefficients corresponding to the pth nonlinearity order, $\mathbf{r_x}$ is the vector comprised of the received signal samples, $\mathbf{u_p}$ represents the pth order orthogonalized basis function vector, $\mathbf{e}$ is a vector of produced cancelled signal samples, and $\boldsymbol{\mu}$ contains the step sizes selected differently for each specific nonlinear term. The OpenCL kernels are highly flexible and the parameters can be adjusted on top level.

In the first step, the filtering kernel computes the cancelled signal. This is carried out by filtering the basis functions and then subtracting the filtered signal from the received signal to produce the cancelled output. To improve efficiency, we assume that the orthogonalized basis functions are already computed from the known transmit data. This requires simple processing and can be carried out, e.g., in separate hardware. Having filter length of L, and to filter N samples, an $L + N$ vector of each basis function is fed to the kernel. The OpenCL kernel is designed in a way that each work item (WI) in each work group (WG) produces one output sample by multiplying and accumulating the corresponding vector of basis functions with the filter coefficients vector. As a result, a vector of length N of the cancelled signal samples is produced. We have used 16-component floating point vectors which is the longest vector length allowed by OpenCL. In total, N WIs are required, and the number of WIs per WG are adjusted depending on each platform to achieve the best performance. As an example, for $N = 256$, the kernel local size, i.e. number of WIs per WGs, for the Core i7, Adreno 430, and Mali-T628 is selected as 256, 64, and 2, respectively.

As explained in Section 2, we aim to adaptively track the time-varying SI channel to use more accurate estimates of the SI channel coefficients. Thus, in the second step, one sample from the produced cancelled signal, along with L basis function samples are fed to the second kernel to update the filter weights. In this kernel, each WI is responsible for processing a 16-element vector. Thus, a total of $L/16$ WIs are required, which are distributed among WGs.



(a) $N = 16$



(b) $N = 32$

Fig. 3: The average power of digital canceller output signal, implemeneted on the Adreno 430, with respect to time, when $L = 16$ for both $P = 1$ and $P = 3$

Both kernels process multiple samples in parallel. However, the two kernels should run sequentially, as a result of the filtering kernel being dependent on the production of updated coefficients. Thus, with the aim of introducing more parallelism to the algorithm, the weight update is done in a way that the weights are only adjusted after a block of N samples are processed. As the value of N rises, more samples are processed simultaneously using the available computing units on the CPU or the GPU. Consequently, the more samples processed in parallel, the more utilized the parallel resources of the cores will be.

4. RESULTS AND ANALYSIS

To evaluate the implemented algorithm for SI cancellation, it is crucial to firstly verify its ability in mitigating the SI signal. After running the algorithm for a set of sample data obtained from an actual full-scale full-duplex radio prototype system, described in [9] and [13], the generated cancelled signal by the software implementation was written to a file. Then, these results were used to create the plots in Fig. 3, using Matlab, which show the average power of the digital canceller output signal in case of both linear ($P = 1$) and third order nonlinear ($P = 3$) digital cancellers. While measuring the reported results, parameters were selected as $L_{pre} = 8$, and $L_{post} = 7$. Figure 3(a) is using the data in the case where $N = 16$ samples are processed simultaneously, while Fig. 3(b) corresponds to the case with $N = 32$. It can be seen that, when using the implemented LMS-based canceller, the power of the cancelled signal is decreasing, and that the nonlinear (P=3) canceller is clearly outperforming the plain linear (P=1) canceller due to its ability to cancel also the third-order nonlinear SI stemming from the nonlinear PA. It can also be observed that the LMS algorithm converges somewhat slower when the SI channel coefficients are updated less often. However, the differ-

Table 1: Execution time when $L = 16$ and $N = 256$ for both the linear ($P = 1$) and third order nonlinear ($P = 3$) digital cancellers

	Core i7		Adreno		Mali	
Clock frequency	2700 MHz		600 MHz		600 MHz	
Parallel PEs	64		∼200		32	
Nonlinearity order	*P*=1	*P*=3	*P*=1	*P*=3	*P*=1	*P*=3
Filtering time for *N* samples [µs]	3.04	3.42	5.88	8.70	59.61	59.88
Time for updating filter weights [µs]	1.52	1.52	3.5	3.58	23.67	24.02
Total time for *N* samples [µs]	4.56	4.94	9.4	12.28	83.28	83.9
Total time for one sample [ns]	17.81	19.29	36.64	48	325.3	327.7

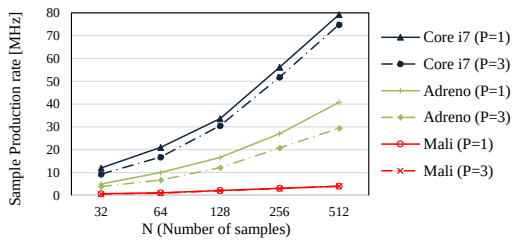


Fig. 4: Sample production rate of both the linear ($P = 1$) and third order ($P = 3$) digital cancellers, for different N, where $L = 16$ and N is the number of samples processed in parallel before updating the SI channel coefficients

ence in the convergence speeds is still rather small, and thus higher N, such as $N = 256$, can be used without extensively slowing down the convergence.

It is also essential to evaluate the feasibility of the OpenCL implementation to carry out the SI cancellation process in a real-time fashion. To be able to process a 20 MHz wide LTE or WiFi carrier, we assume a sample rate of $F_s = 24$ MHz. Thus to achieve real-time processing, the output signal should be produced at a 24 MHz rate, meaning that production of each output sample should take equal to or less than 41.66 ns ($1/24$ MHz= 41.66 ns).

Table 1 shows kernel execution times for both stages of the algorithm and the total time, using both the linear and nonlinear digital cancellers, on all three platforms. It should be noted that data transfer times are not included in the reported execution times. Table 1 also lists the clock frequency and number of parallel processing elements (PE) of the corresponding platforms for efficiency comparison. These results correspond to the case where in the LMS filtering phase, $N = 256$ samples are filtered simultaneously, and then the filter coefficients are updated by the second kernel. Comparing the execution times of the linear ($P = 1$) and nonlinear ($P = 3$) cancellers shows that the added complexity from the nonlinear canceller has resulted in slightly slower execution of the kernels. It can be seen that using a linear canceller and having the filter weights updated after processing 256 samples, both the Intel Core i7 CPU and the Qualcomm Adreno 430 meet the timing constraints. In case of a third order nonlinear canceller, while the Core i7 easily fits in the real-time processing limits, the Adreno GPU takes approximately 6 ns longer. However, by increasing N as shown in Fig. 4, and utilizing the parallel resources of the GPU, real-time nonlinear SI cancellation can also be realized using the Adreno 430.

Although Mali-T628 runs at a clock frequency close to Adreno's, results achieved by Mali show much slower performance. This can be explained by number of parallel computing units. Each of Mali-T628's four cores are capable of computing eight parallel floating point operations each cycle in their vector pipelines [14]. In contrast, Adreno 430 architecture is kept more in secret but it seems to be capable of supporting approximately 200 floating point operations per cycle. This is also supported by the presented results in Table 1 which shows Mali to be approximately six times ($200/32 = 6.25$) slower than Adreno. However there can be other details in the hardware architecture which this hypothesis overlooks.

The graph in Fig. 4 demonstrates how introducing more parallelism, by increasing the number of samples processed in parallel, affects sample production rate. In most cases, doubling the number of the input samples of the filtering kernel results in approximately the same execution time, while the time for updating filter coefficients does not increase at all. As a result, sample production rate nearly increases by a factor of two. It can be seen that, already at $N = 128$, it is possible to achieve a real-time implementation using the Intel Core i7, while the Adreno 430 is capable of real-time nonlinear cancellation with $N = 512$. In both platforms, the real-time implementation is realized without requring all the available processing resources. Although larger N is required to achieve real-time implementation, it will result in higher latency for the system. Thus, there is a trade-off between latency and sample production rate. As the signal is filtered in blocks of N samples, a latency equal to the filtering time of the first set of N samples should be considered only in the beginning of the process. For $N = 256$, this latency is equal to the filtering time reported in Table 1. The latency, using the Adreno 430 and in case of nonlinearity order $P = 3$, for $N = 512$, $N = 128$, $N = 64$, and $N = 32$ is equal to 13.82 µs, 6.91 µs, 5.89 µs, and 4.60 µs, respectively, which should be taken into consideration according to the application requirements.

5. CONCLUSIONS

In this paper, an SDR implementation of an adaptive nonlinear digital self-interference cancellation method for full-duplex transceivers, especially on the mobile side, was presented. The implemented solution was evaluated and analysed to demonstrate the performance achieved by the proposed method in addition to the feasibility of a real-time software-based implementation on multi-core platforms, especially on mobile GPUs. The results showed that using the implemented advanced digital SI canceller, the SI signal can be attenuated to a great extent. Furthermore, utilizing the Qualcomm Adreno 430 GPU on the mobile side, and the Intel Core i7 CPU on the base station side, the cancelled signal can be produced at the required rates for real-time processing, in case of, e.g., 20 MHz cancellation bandwidth. Hence, it can be concluded that, using off-the-shelf mobile GPUs, a real-time implementation of the proposed LMS-based solution for adaptive nonlinear digital SI cancellation is feasible also for mobile scale full-duplex devices. This can help in realizing the theoretical potential throughput gains provided by full-duplex communications. Moreover, taking advantage of the programmability of GPUs and CPUs, this solution provides high flexibility for possible required algorithmic reconfigurations and extensions. In the continuation of this work, we will aim at increasing the sample production rate using more advanced GPUs, while employing higher nonlinearity orders, which adds to the complexity of the implementation.

6. REFERENCES

[1] S. Hong, J. Brand, J. I. Choi, M. Jain, J. Mehlman, S. Katti, and P. Levis, "Applications of self-interference cancellation in 5G and beyond," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 114–121, February 2014.

[2] M. Heino, D. Korpi, T. Huusari, E. Antonio-Rodriguez, S. Venkatasubramanian, T. Riihonen, L. Anttila, C. Icheln, K. Haneda, R. Wichman, and M. Valkama, "Recent advances in antenna design and interference cancellation algorithms for in-band full duplex relays," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 91–101, May 2015.

[3] E. Everett, M. Duarte, C. Dick, and A. Sabharwal, "Empowering full-duplex wireless communication by exploiting directional diversity," in *Proc. of Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 6-9 Nov 2011, pp. 2002–2006.

[4] Intel Corporation, *Intel® Core™ i7 Processor Family for LGA2011 Socket*, May 2014.

[5] ARM Ltd., *The ARM® Mali™ Family of Graphics Processors*, February 2013.

[6] Qualcomm Technologies, *Snapdragon 810 processor product brief*, February 2015.

[7] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan, and R. Wichman, "In-band full-duplex wireless: Challenges and opportunities," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637–1652, Sept 2014.

[8] D. Korpi, L. Anttila, V. Syrjälä, and M. Valkama, "Widely linear digital self-interference cancellation in direct-conversion full-duplex transceiver," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1674–1687, Sept 2014.

[9] D. Korpi, Y. S. Choi, T. Huusari, L. Anttila, S. Talwar, and M. Valkama, "Adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio: Algorithms and rf measurements," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 6-10 Dec 2015, pp. 1–7.

[10] M. Duarte, C. Dick, and A. Sabharwal, "Experiment-driven characterization of full-duplex wireless systems," *IEEE Transactions on Wireless Communications*, vol. 11, no. 12, pp. 4296–4307, December 2012.

[11] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, "Stationary and nonstationary learning characteristics of the lms adaptive filter," *Proceedings of the IEEE*, vol. 64, no. 8, pp. 1151–1162, Aug 1976.

[12] Ltd. Hardkernel co., "Odroid-xu3.," 2013, Available at http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127.

[13] D. Korpi, T. Huusari, Y. S. Choi, L. Anttila, S. Talwar, and M. Valkama, "Full-duplex mobile device - pushing the limits," *IEEE Communications Magazine*, accepted. Available at http://arxiv.org/abs/1410.3191.

[14] Peter Harris, "The mali GPU: An abstract machine," 2014, Available at https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

PUBLICATION 6

M. Aghababaeetafreshi, D. Korpi, M. Koskela, P. Jääskeläinen, M. Valkama and J. Takala, "Software defined radio implementation of a digital self-interference cancellation method for inband full-duplex radio using mobile processors," *Journal of Signal Processing Systems*, Oct 2018, 90(10), pp. 1297–1309, doi: 10.1007/s11265-017-1312-0

Software Defined Radio Implementation of a Digital Self-interference Cancellation Method for Inband Full-Duplex Radio Using Mobile Processors

Mona Aghababaeetafreshi · Dani Korpi · Matias Koskela · Pekka Jääskeläinen · Mikko Valkama · Jarmo Takala

Abstract New means to improve spectral efficiency and flexibility in radio spectrum use are in high demand due to congestion of the available spectral resources. Systems deploying inband full-duplex transmission aim at providing higher spectral efficiency by concurrent transmission and reception at the same frequency. Potentially doubling system throughput, full-duplex communications is considered as an enabler technology for the upcoming 5G networks. However, system performance is degraded due to the strong self-interference (SI) caused by overlapping of high power transmit signal with the received signal of interest. Furthermore, due to commonly existing radio frequency imperfections, advanced techniques capable of mitigating nonlinear SI are required. This article presents a real-time software-defined implementation of a digital SI canceller for full-duplex transceivers, potentially applicable even in mobile-scale devices. Recently, software-defined radio has gained a lot of interest due to its higher flexibility, scalability, and shorter time-to-market cycles compared to traditional fixed-function hardware designs. Moreover, as the performance enhancements achieved by increasing the clock frequency is reaching its limits, the current trend is towards multi-core processors. Since contemporary mobile phones already contain powerful massively parallel GPUs and CPUs, feasibility of a real-time implementation on mobile processors is studied. The reported results show that by adopting the presented solution, it is possible to achieve sufficient SI cancellation under time varying coupling channel conditions. Additionally, the possibility of carrying out such advanced processing in a real-time fashion on the selected platforms is investigated, and the implementation is evaluated in terms of execution time, power, and energy consumption.

Keywords 5G · full-duplex · self-interference cancellation · GPU · OpenCL

M. Aghababaeetafreshi
Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland
E-mail: mona.aghababaeetafreshi@tut.fi

1 Introduction

In full-duplex communications, transmission and reception are carried out using the same spectral and temporal resources. Since this simultaneous use of bandwidth for both transmission and reception can theoretically increase the throughput by a factor of two, inband full-duplex communication is considered a promising enabler for the future 5G networks [17]. However, deployment of such systems is extremely challenging due to the strong self-interference (SI) produced as a result of the transmit signal coupling to the receiver. Thus, a crucial step toward achieving the promised gain in throughput by full-duplex transmission is to effectively attenuate the SI signal [3]. This is a complicated task, as it is not possible to simply subtract the known transmit signal from the received waveform to obtain the signal of interest. The reason behind this is the linear and nonlinear distortion of the signal while propagating from transmitter to the receiver due to transceiver analogue imperfections [24][16]. This makes efficient suppression of the SI signal the main obstacle in realizing full-duplex systems.

The analogue imperfections make SI cancellation even more challenging on the mobile side compared to the base station side, since typically low cost components are employed in mobile devices. Furthermore, processing resources and power consumption are limiting factors in hand-held devices. Thus, effective yet less complex solutions are required for the user equipment

side in order to exploit the full potential of full-duplex systems. An SI cancellation method designed for a mobile station should also take the continuously changing environment around the device into consideration. Thus, an adaptive solution should be developed to keep track of the time-varying SI channel.

Currently, advanced processor architectures take advantage of parallel processing to achieve higher performance, since performance enhancement through increasing the processors' operating frequency in a fixed power envelope has reached technical challenges [20][9]. Thus, we take advantage of the parallel processing capabilities of multi-core GPUs and CPUs. Moreover, to better utilize the parallel resources of the processors, Open Computing Language (OpenCL) is used. OpenCL is a programming standard for heterogeneous platforms, enabling efficient access to the available parallel resources [20].

In this work, a software based implementation for the entire digital canceller, which includes an orthogonalization procedure, together with a parameter learning algorithm is introduced. Such software defined radio (SDR) implementation provides more programmability, lower expenses, less design efforts, and thus shorter time-to-market cycles compared to traditional fixed-function approaches [30][13]. To demonstrate the feasibility of an SDR implementation of the digital canceller, we worked with commercial off-the-shelf (COTS) low-cost components, which highlight the true advantages of a software-based solution. The implementation is carried out on four multicore platforms, which are suitable for both the network and user equipment side. The employed platforms are Qualcomm® Adreno™ 430, ARM® Mali™-T628 MP6, ARM® Cortex®-A15, and Intel® Core™ i7-4800MQ. The core i7 desktop CPU was mainly used for comparison purposes, while the rest of the processors are the main target of the study.

Using the measured signals from an actual full-duplex prototype, we demonstrate that sufficient real-time suppression of the SI signal is feasible using the proposed implementation. Furthermore, the implemented canceller is evaluated in terms of execution time, delay, power, and energy consumption. This article is a continuation of the work presented in [2].

The rest of the paper is organized as follows. Section 2 introduces some of the related work, existing in the literature. Section 3 describes the digital SI cancellation method adopted in this work. Section 4 explains the implementation of digital canceller blocks, and presents the selected platforms. Then, in Section 5, the implementation results are shown and analyzed. Finally, conclusions are drawn in Section 6.

2 Related work

As mentioned in the previous section, sufficient cancellation of the SI signal is the main challenge in achieving a system operating effectively in full-duplex mode. This topic has been researched extensively and various techniques have been introduced in the literature.

In [21], a novel RF canceller architecture is described which cancels both the direct antenna coupling and multipath effects, while [3] proposes an all digital cancellation method. [7] uses three different methods for SI suppression with both analogue and digital cancellation. Like [7], the proposed methods in literature typically include different stages of cancellation such as, propagation, analogue, and digital domain cancellation [29][16]. Some studies assume that only the base station functions in full-duplex mode, while the mobile equipment remains operating in half-duplex mode. An example of which can be found in the work presented in [10].

Some contributions toward actual prototypes capable of full duplex communications can be found in the literature, such as the ones described in [6], [8], [16], and [27]. However, there are very few existing articles on real-time implementation of digital SI cancellation.The work in [22] implements parts of the digital cancellation method proposed in [23] on an FPGA. However, no contributions regarding a software based implementation of digital SI cancellation targeted for a mobile-scale device, as the one reported here, can be found in the literature. Especially this work uses COTS elements and eliminates the need for custom hardware design and additional hardware components.

Some contributions with similar arithmetic computations and implementation techniques, used for digital predistortion design, can be found in the literature. These works, found in [12,25,26], also study parallel processing on mobile-scale multicore processors, in which evaluations of the achieved performance are also reported. However, experimental measurements of power, or energy consumption are not carried out.

3 Digital self-interference cancellation

In order to reduce the SI signal to a level not interfering with the desired received signal decoding, both radio freqncy (RF) and digital domain cancellation are required. The former prevents the analogue-to-digital converter and the receiver low-noise amplifier (LNA) from saturating. However, further suppression of the SI signal should be carried out in the digital domain to improve system performance. The overall structure of the full-duplex transceiver, including both the RF and

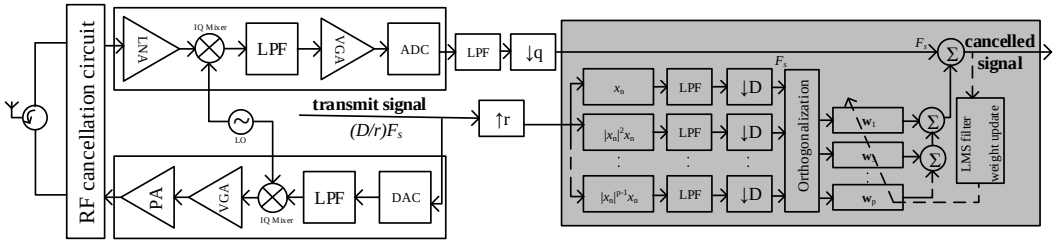


Fig. 1 Overall structure of a full-duplex transceiver, where the grey part is implemented in software.

digital cancellation is shown in Fig. 1. In this section, we address the latter by first introducing a model for the SI signal.

3.1 Self-interference Modeling

The transmitter and receiver paths contain numerous non-ideal components which distort the transmitted signal in linear and nonlinear ways. The transceiver impairments include nonlinear distortion by power amplifiers, phase-noise of the local oscillator, quantization noise from the analogue-to-digital converter, and in-phase/quadrature (I/Q) imbalance of the transmitter and receiver. Since the transmitter power amplifier (PA) is usually the most significant source of nonlinearity, we model the transmit signal by adopting the parallel Hammerstein (PH) model, commonly used for a highly nonlinear PA. Denoting the PA input by $x_{PA,in}$, the PA output, using the PH model, can be written as [23]:

$$x_{PA,out} = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{k=0}^{K-1} h_p^{PA}(k) u_p(x_{PA,in}(n-k)), \tag{1}$$

where P represents the highest nonlinearity order of the PA model, K is the memory length of the PA, h_p^{PA} is the pth-order model for the PA memory, and $u_p(x_{PA,in}(n))$ is computed as $|x_{PA,in}(n)|^{p-1} x_{PA,in}(n)$ and produces the pth-order basis function.

Now with the transmitter PA as the most prominent cause of nonlinear distortion, the whole SI channel can be effectively modelled using (1). Thus, the received signal at the digital canceller input, with respect to the original transmitted signal $x(n)$, can be expressed as:

$$r_x(n) = \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} h_p(l) u_p(x(n-l)) + z(n), \tag{2}$$

where L is the memory length of the effective SI channel, $h_p(l)$ contains the coefficients for the effective pth-order SI channel, and $z(n)$ represents the noise and possible modeling mismatch. After estimating the unknown SI channel coefficients, denoted here by $\hat{h}_p(l)$, the signal at the output of the digital canceler can be written as:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_p(l) u_p(x(n-l)). \tag{3}$$

Looking at equations (2) and (3), with accurate estimation of the SI channel coefficients, only noise should remain after digital cancellation, meaning that $e(n) \approx z(n)$. Furthermore, the estimated coefficients need to be updated, as the surrounding environment of a mobile device changes over time. The method used for the estimation should also have low computational complexity in order to be suitable for mobile-scale processing resources. Taking the aforementioned requirements into account, we have adopted the LMS based solution proposed in [23].

3.2 Orthogonalization

Since the different basis functions, mentioned in the previous section, are functions of the same transmit signal, they tend to be somewhat correlated. This will result in slow convergence of the LMS-based coefficient estimation. To alleviate this problem, the basis functions are orthogonalized using the method proposed in [23], which is briefly described here.

The basis functions are orthogonalized using a whitening transformation matrix. This matrix can be generated by eigendecomposition of the covariance matrix $\mathbf{\Sigma}$. Defining the instantaneous basis function vector as:

$$\mathbf{u}(n) = \begin{bmatrix} u_1(x(n)) & u_3(x(n)) & \dots & u_p(x(n)) \end{bmatrix}^T, \tag{4}$$

with $u_p(x(n)) = |x(n)|^{p-1} x(n)$, the covariance matrix of basis functions across different nonlinearity orders can be defined as:

$$\mathbf{\Sigma} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}(n)^H]. \tag{5}$$

Having $\mathbf{\Sigma} = \mathbf{VDV^H}$, where diagonal matrix $\mathbf{D}$ contains the eigenvalues of Σ, and matrix $\mathbf{V}$ consists of the eigenvectors, the transformation matrix $\mathbf{T}$ can be written as:

$$\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V}^H. \tag{6}$$

Using the transformation matrix $\mathbf{T}$, the orthogonalized basis functions can be calculated by:

$$\tilde{\mathbf{u}}(n) = \mathbf{Tu}(n). \tag{7}$$

Now (3) can be re-written using the orthogonalized basis functions as follows:

$$e(n) = r_x(n) - \sum_{\substack{p=1 \\ p\ odd}}^{P} \sum_{l=0}^{L-1} \hat{h}_{p,ort}(l)\tilde{u}_p(x(n-l)), \tag{8}$$

where $\tilde{u}_p(x(n))$ are the orthogonalized basis functions using matrix $\mathbf{T}$, and $\hat{h}_{p,ort}(l)$ represents the corresponding SI channel estimates. Adopting vector notations, (8) can be expressed as:

$$e(n) = r_x(n) - \mathbf{w}^H\mathbf{u}_{ort}(n), \tag{9}$$

where

$$\mathbf{w} = \Big[\hat{h}_{1,ort}(0), \hat{h}_{3,ort}(0), \dots, \hat{h}_{P,ort}(0), \dots \\ \hat{h}_{1,ort}(L-1), \hat{h}_{3,ort}(L-1), \dots, \hat{h}_{P,ort}(L-1)\Big]^T, \tag{10}$$

and

$$\mathbf{u}_{ort}(n) = \Big[\tilde{\mathbf{u}}(n)^T, \tilde{\mathbf{u}}(n-1)^T, \dots, \tilde{\mathbf{u}}(n-L+1)^T\Big]^T. \tag{11}$$

It is worth mentioning that the covariance matrix $\mathbf{\Sigma}$ depends only on the statistical properties of the original transmit signal, and consequently it is not time varying. Therefore, we can assume that the transformation matrix $\mathbf{T}$ is computed and known beforehand.

3.3 LMS parameter learning

In this step, the effective SI channel coefficients are estimated using the decorrelated basis functions. This is carried out using an LMS-based algorithm with specific step-sizes for the different nonlinear terms [31]. Both pre-cursor and post-cursor taps are considered for a precise memory model of the SI channel. The original learning algorithm proposed in [23] is modified so that the estimated weights are not updated with every sample but only after a block of N samples are processed. This computing-friendly LMS-based approach is described in Algorithm 1, where $\tilde{\mathbf{u}}$ is a vector containing the orthogonalized basis functions calculated in (7), $\mathbf{w}$ contains the corresponding SI channel coefficients, $r_x(n)$ is the received signal, $e(n)$ represents the cancelled signal, and L_{pre} and L_{post} are the amounts of pre-cursor and post-cursor taps, respectively. Furthermore, $\boldsymbol{\mu}$ contains the step sizes, and N controls how often $\mathbf{w}$ is updated.

Algorithm 1 LMS-based adaptive nonlinear digital cancellation.

Initialize:
  $\mathbf{w} \leftarrow [0\ \dots\ 0]$
  $n \leftarrow L_{post}$
while transmitting **do**
  $\mathbf{u_{ort}}(n) = \big[\tilde{\mathbf{u}}(n+L_{pre})^T \dots\ \tilde{\mathbf{u}}(n-L_{post})^T\big]^T$
  $e(n) = r_x(n) - \mathbf{w}(n)^H\mathbf{u_{ort}}(n)$
  if ($n \mod N == 0$) **then**
    $\mathbf{w}(n+1) \leftarrow \mathbf{w}(n) + \boldsymbol{\mu}e^*(n)\mathbf{u_{ort}}(n)$
  end if
  $n \leftarrow n+1$
end while

4 Implementation

4.1 Implemented blocks

In this section, the blocks, implemented in software, for the digital SI canceller, shown in Fig. 2 are described in short.

Basis functions calculation The first step is to calculate the nonlinear transformations of the original transmit signal. The pth-order basis function is computed for each sample as $u_p(n) = |x(n)|^{p-1}x(n)$. In this implementation, highest considered nonlinearity order is $P = 3$.

Polyphase filtering As shown in Fig. 1, the transmit signal is oversampled before generating the basis functions. Thus the calculated basis functions can be resampled to the final cancellation signal sample rate. Assuming a decimation factor equal to D, only every D-th sample is kept after appropriate lowpass filtering.

To eliminate the unnecessary computations, we have designed a polyphase filter to perform the resampling task. This results in a more efficient implementation as the filtering is not performed on all original signal samples. An illustration of the adopted polyphase filter with downsampling factor D can be seen in Fig. 3, where $F_0, \dots, F_{D-1}$ are sub-filters of length M. The total length for the polyphase filter is equal to $M \times D$.

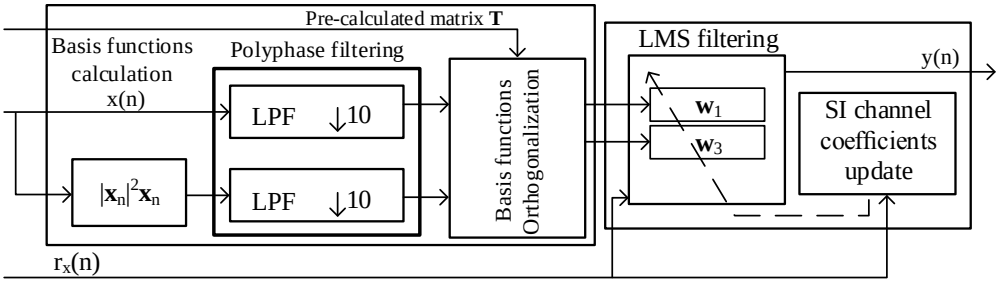


Fig. 2 Implemented blocks for a third-order digital SI canceller, shown also in the grey part in Fig. 1.

This work employs a polyphase filter with total length of 20, having downsampling factor $D = 10$, and sub-filter length $M = 2$.

The OpenCL implementation for the polyphase filter was carried out with both vector and scalar data types. With careful re-arrangement of the filter coefficients, the data loads can be carried out in a more efficient way. Fig. 4 illustrates an example implementation and work-load distribution for the polyphase filter. In this figure, the data and the coefficients are loaded as vectors of length four into vectors $\mathbf{x}$ and $\mathbf{p}$, respectively. After multiplication and summation, each work-item produces one output sample $\mathbf{y}[\mathbf{n}]$. In Fig. 4, k denotes the polyphase filter length, $k = M \times D$, number of work groups is represented by n, and a local size of 16 is assumed for a clearer presentation.

Computing orthogonalization matrix This step is done according to equations (4) to (6). However, as mentioned in the previous section the transformation matrix depends only on the statistical properties of the transmit signal, and does not change over time. Thus, we have assumed that the transformation matrix $\mathbf{T}$ is precomputed to reduce complexity and unnecessary computations. Having nonlinearity order $P = 3$, $\mathbf{T}$ is a 2×2 matrix.

Basis function orthogonalization After going through the polyphase filter, the basis functions are orthogonalized using the precomputed matrix $\mathbf{T}$, according to equation (7). This helps the LMS learning process to converge faster.

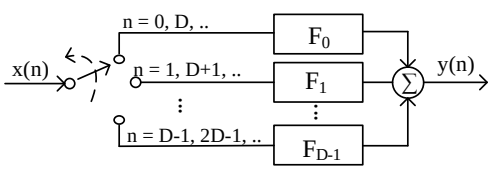


Fig. 3 Functional structure of a polyphase filter with decimation factor D, where $y_(n)$ represents the signal samples after downsampling and filtering $x(n)$.

LMS filtering The orthogonalized basis functions are filtered with the SI channel coefficient estimates. The filter length, i.e., the SI channel memory, is defined as $L = (L_{pre} + L_{post} + 1) \times (\frac{P+1}{2})$. Then the filtered results are subtracted from the received signal to produce the cancelled signal. This corresponds to the computations from line 6 in Algorithm 1. The SI channel coefficients are updated after a block of N samples are processed using the *SI channel coefficients update* kernel.

SI channel coefficients update Having the cancelled signal samples and step sizes $\boldsymbol{\mu}$, the SI channel estimates are updated as described in lines 7-10 in Algorithm 1. The selected step size is equal to 0.01 and 0.001 for the linear and third order terms, respectively. To reduce the computations, this step is also modified such that the coefficients are only updated after processing every N sample. This is done so that the LMS filter kernel would not have to wait for updated coefficients after processing every single sample. Less frequent updating of the coefficients reduces the dependency of the two kernels, the *LMS filter* and *SI channel coefficients update* kernels, and helps to increase parallelism, having larger blocks of input samples for the LMS filter kernel.

4.2 Platforms

In this work, three mobile scale multi-core processors and one desktop CPU are selected as the processing platforms. These are commercial off-the-shelf products that are currently employed in some of the available devices in the market. These platforms are briefly introduced in the following.

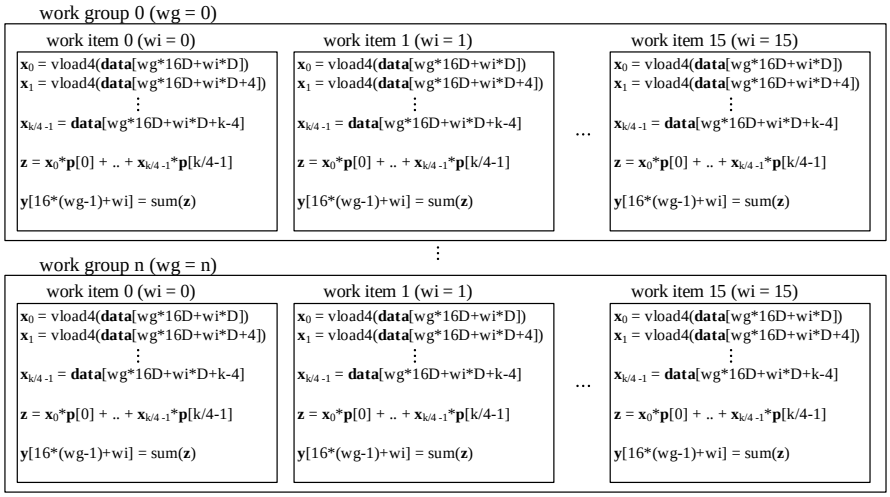


Fig. 4 OpenCL kernel structure and workload distribution for the polyphase filter.

Qualcomm® Adreno^{TM} 430 Adreno 430 is a mobile GPU by Qualcomm, and is available in the Snapdragon 810 System on Chip (SoC). This GPU is designed for mobile-scale devices and can run at 500 MHz, 600 MHz, or 650 MHz clock frequency [28]. Very little information about Adreno's architecture is publicly available, but it seems that it can approximately support 200 floating point operations in one clock cycle. To run the digital canceller blocks on Adreno 430, a commercial Android phone was used.

ARM® Mali^{TM}-T628 MP6 Similar to Adreno, Mali is a mobile-scale GPU and runs at a 600 MHz clock frequency [5]. Mali-T628 is a part of the Samsung Exynos 5 Octa (Exynos 5422) SoC. This GPU can scale from one to eight cores. Each core can handle up to eight floating point operations per cycle [15]. In this work, Odroid XU3 board [14] was used to access Mali.

ARM® Cortex®-A15 The Cortex-A15 MPCore is a low power multicore processor that can have one to four cores [4]. This multicore processor can be found, for example in the Exynos 5 Octa (Exynos 5422) SoC. It runs at 1.4 GHz clock frequency. Each of the four cores has one NEON (advanced Single Instruction Multiple Data instruction set) and vector floating point unit. The same Odroid XU3 board was used for implementing the digital canceller on A15 CPU.

Intel® Core^{TM} i7-4800MQ Unlike the other three processing units mentioned above, the Intel Core i7 is a desktop CPU. This processor has four cores and can run at up to 3.7 GHz [18].

5 Evaluation and analysis

In this section, the implementation results of the digital canceller blocks introduced in the previous section are presented. First, using the data from an actual full-duplex prototype system, described in [23] and [24], we demonstrate that the presented digital cancellation implementation can efficiently suppress the self-interference signal. Then, we evaluate this solution in terms of execution time, power, and energy consumption to study the feasibility of such software-based implementation using the four aforementioned COTS processors.

Software tailoring: To optimize the implementation, the kernels are tailored for each platform. Having a scalar or vector based implementation, the different possible vector lengths, and workload distribution between the OpenCL work-items are the factors that greatly affect the execution time of each processing task.

Different kernel designs on Mali showed that employing floating point vectors of length four yields the best results. Running the kernels on A15, different vector lengths and in some cases the scalar based implementation show similar results. However, execution of the kernels is fastest when the workload is distributed such that there are two work groups. The kernels designed for the Core i7 use vectors of length 16, and in most cases perform more efficiently when the processing is divided among eight work groups. Similar to Mali, Adreno achieves higher performance when using vectors of length four. Furthermore, workload should be spread among four work groups. The implementation results presented in the following section are obtained having

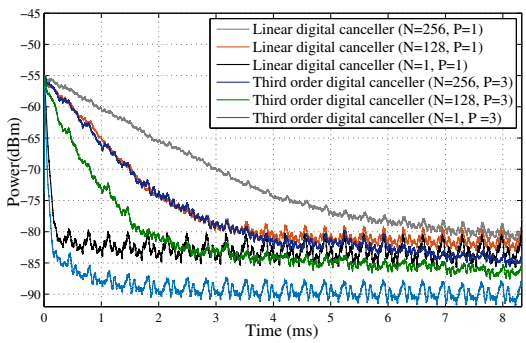


Fig. 5 The instantaneous power of the SI signal, averaged over 1000 samples, of linear ($P = 1$) and third order ($P = 3$) digital canceller output signal, implemented on the Adreno 430, with respect to time, for $N = 1$, $N = 128$, and $N = 256$.

designed the most efficient kernel implementation for each platform.

5.1 Digital SI canceller performance

After the sampled data, collected from the real full-duplex prototype system, was processed on the platforms, the cancelled signal was used to plot Fig. 5. Input buffers of 10, 1280 and 2560 samples are considered, which means that after downsampling by a factor of 10 and orthogonalization, one sample or blocks of 128 and 256 samples are processed before updating the canceller coefficients. When creating the plots in Fig. 5, L_{pre} and L_{post} are set to 8 and 7, respectively. Having total channel length $L = (L_{pre} + L_{post} + 1) \times (\frac{P+1}{2})$, L is equal to 16 for the linear canceller and 32 for the third order canceller.

This figure shows that the implemented canceller is capable of sufficient suppression of the SI signal, close to the receiver noise floor (-90 dBm). Allowing more time to converge will result in almost perfect SI cancellation. Being able to cancel the third order nonlinear SI, the third order canceller shows superior performance compared to the linear one. Comparing the curves in Fig. 5, it can be seen that less frequent updating of the SI channel coefficients has resulted in slower convergence of the LMS-learning algorithm. However,the difference is relatively small, especially after the initial learning phase, indicating that less frequent updating of the channel coefficients is a feasible option for controlling the computational complexity of the digital canceller.

5.2 Execution time analysis

In this section, the execution times related to the different building blocks of the SI digital canceller running on four different platforms, introduced in Section 4.2, are reported. A key factor in using OpenCL and multicore platforms with single instruction, multiple data (SIMD) or single program, multiple data (SPMD) optimized hardware is being able to take advantage of the available data parallelism. High performance can be achieved when parallel elements of the processor are utilized efficiently and the work load is distributed properly between these elements. We add to the inherent parallelism of the algorithms by increasing the amount of data processed in each kernel call. As a result, the processing time for each signal sample is decreased. Furthermore, vector lengths and workload distribution are adjusted for each implemented block on each platform so that kernel executions are carried out more efficiently.

The execution times for each digital canceller block implemented on the four platforms are presented in Tables 1 - 4. It should be noted that the reported times do not include data transfer, as SoC design can be easily made so that the processing unit sees the same memory as the radio hardware. The tables show the result using different buffer sizes for both linear and third order cancellers.

As the buffer lengths increase, the processing time related to one data sample decreases. In many cases, the processing time is approximately reduced by a factor of two, when the buffer size is doubled. This is the case for the “orthogonalization” and “weight update” kernels. However, the two filtering kernels, “polyphase” and “LMS”, achieve lower speed-up due to their inherent lack of parallelism, stemming from the summation step of convolution in the filters. The size of the buffers fed to the first block are chosen as powers-of-two multiplied by $D = 10$, which is the downsampling factor.

Moreover, the “basis functions” kernel’s execution speed does not scale linearly with the buffer size. This can be explained by the input buffer size of this kernel which is ten times bigger than that of the “orthogonalization” and “weight update” kernels, which are executed after downsampling. This larger amount of data could saturate the available parallel resources of the cores, resulting in a slower speed-up. The effect of increasing the buffer size on the overall achieved performance is illustrated in Fig. 6. With longer buffers, the production rate improves less as the processing resources reach saturation.

The presented results show that Mali and A15 are only capable of processing the signal at rates lower than

Table 1 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Adreno 430* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	1,89	-	1,50	-	1,37	-	1,21
Polyphase	23	44,10	16	30,5	13,75	26,50	12,21	22,75
Orthogonalization	11	18	5,50	11,50	2,75	5,75	2,25	4,75
LMS filter	23	32,76	17	23,28	14,25	20,05	12,75	18,32
Weight update	11	11	5,50	5,50	2,75	2,85	1,38	1,27
Total [ns]	68	107,75	44	72,28	33,50	56,52	28,59	48,30
Rate [MHz]	14,71	9,29	22,73	13,84	29,85	17,69	34,98	20,70

Table 2 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Cortex A15* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$	$P = 1$	$P = 3$
Basis functions	-	34,76	-	17,96	-	10,25	-	5,85
Polyphase	312,50	622,50	207,03	411,52	122,07	242,49	77,63	152,94
Orthogonalization	320,31	328,12	164,02	164,06	82,03	84,96	44,92	45,41
LMS filter	398,43	476,56	222,65	306,64	134,76	214,84	92,77	167,96
Weight update	265,62	242,18	142,57	125	83	81,05	42,96	40,52
Total [ns]	1296,8	1704,1	736,27	1025,2	421,86	633,59	258,28	412,68
Rate [MHz]	0,77	0,59	1,36	0,97	2,37	1,57	3,87	2,41

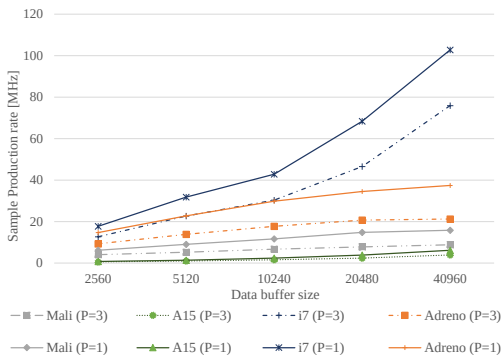


Fig. 6 Sample production rate increase with regards to buffer size on the four platforms for both linear and third order cancellers.

15 MHz, even with large input data buffers. However, linear digital cancellation can be carried out on the Adreno 430 GPU and the Core-i7 CPU at rates over 20 MHz, having buffer sizes of 5120 samples. Furthermore, the Core-i7 and the Adreno 430 can perform third order digital cancellation for a 20 MHz waveform with buffer size of 5120, and 20480 samples, respectively.

Comparing the linear and third order cancellers in Tables 1 - 4, it can be seen that the polyphase filtering in the third order canceller takes approximately twice as much time as in the linear one. This is due to the fact that in case of a third order canceller, two filtering kernels are employed for both the linear and third order basis functions. The calculation of third order basis functions, carried out by the "basis functions" kernel is redundant in the linear canceller. The rest of the implemented blocks require equal or slightly more time for the third order canceller, as they only differ in a few multiplications and/or additions.

5.3 Delay analysis

As discussed previously, to add to the available parallelism of the algorithm and utilize the parallel resources of the processors more efficiently, we increase the amount of data processed in each kernel, having longer input buffers. The disadvantage of this approach are longer delays for the system as larger blocks of data must be processed in each kernel call. The overall delays related to different buffer sizes for each platform

Table 3 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Mali-T628* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$
Basis functions	-	3,41	-	2,49	-	2,21	-	2,02
Polyphase	46,14	90,11	39,66	77,68	36,08	71,09	33,2	64,85
Orthogonalization	19,33	40,97	10,36	21,62	5,43	12,31	2,46	6,7
LMS filter	65,79	87,16	44,52	72,32	37,83	55,51	28,48	50,41
Weight update	30,51	25,19	16,08	16,84	6,15	8,44	3,43	4,54
Total [ns]	161,77	246,84	110,62	190,95	85,49	149,56	67,57	128,52
Rate [MHz]	6,18	4,05	9,04	5,23	11,70	6,68	14,80	7,78

Table 4 Execution times of one signal sample for different kernels with respect to buffer lengths when implemented on *Core i7* for both the linear and third order canceller.

Buffer size	2560		5120		10240		20480	
Nonlinearity order	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$	$P=1$	$P=3$
Basis functions	-	1,92	-	0,66	-	0,55	-	0,48
Polyphase	20,78	39,86	12,02	22,34	9,64	17,97	6,12	11,38
Orthogonalization	5,93	7,42	3,71	4,45	2,22	2,59	1,29	2,22
LMS filter	23,75	22,26	12	12,90	9,64	10,02	6,30	6,49
Weight update	5,93	7,42	3,71	3,71	1,85	1,85	0,92	0,92
Total [ns]	56,39	78,88	31,44	44,06	23,35	32,98	14,63	21,49
Rate [MHz]	17,73	12,67	31,80	22,69	42,82	30,32	68,35	46,53

are listed in Table 5. This delay is calculated as:

$$\begin{aligned} overall\ delay = & T_{basisfunctions} \times buffer\ size \\ & + T_{polyphase} \times \frac{buffer\ size}{D} \\ & + T_{orthogonalization} \times \frac{buffer\ size}{D} \\ & + T_{LMS\ filter} \times \frac{buffer\ size}{D} \\ & + T_{weight\ update} \times \frac{buffer\ size}{D}, \end{aligned} \tag{12}$$

where T_{kernel} is the processing time for one signal sample of "kernel", and D is the downsampling factor.

The calculated overall delay is equal to $25,6$ μs for a third order SI canceller implemented on Core i7 and $70,5$ μs on Adreno 430 with input buffer sizes of 5120 and 10240, respectively. These delays can be considered more than reasonable, when compared, e.g., to the inherent receiver processing latency of LTE user equipment (UE) which is, in minimum, 1 ms due to the downlink reference symbol structure as well as the adopted codeword mapping and interleaving processing. Furthermore, the specifications [11] allow an additional processing time of 3 ms for sending downlink hybrid ARQ (HARQ) acknowledgement within uplink L1/L2 control signaling. Thus, a balance can be achieved in the delay and sample production rate trade-off for a real application.

5.4 Power consumption analysis

Power measurement is not possible on Adreno 430 and Core i7, as no tools are provided for this purpose on the employed platforms. However, the Odroid XU3 board is equipped with sensors which allow measuring the power consumed by the Mali GPU, the DRAM, and both A7 and A15 CPUs. It is possible to probe the sensor data at discrete time instances. Thus, to provide a more reliable power consumption estimate, we take 200 samples of the sensor data in intervals of 100 ms. As the kernels are very small, they should be repeatedly run during this 20s interval. This keeps the processor cores occupied by the intended kernel. Then, the data from the sensors is averaged over the 20s period. However, any program running in the background, such as the operating system could partly account for the CPU/GPU power consumption. Thus, the processors idle power, i.e., power consumption while not running any kernels, are computed and subtracted from the measured results.

Fig. 7 shows the average power measured when running the kernels for processing 5120 signal samples, as the consumed power by the GPU and CPU does not change significantly with different buffer lengths. It can be seen that there is very little or no difference in power consumption between the linear and third or-

Table 5 Overall delay in microseconds for different buffer lengths on all four platforms.

Buffer length	2560		5120		10240		20480	
Nonlinearity order	P=1	P=3	P=1	P=3	P=1	P=3	P=1	P=3
Mali	41,41	71,04	56,63	109,24	87,54176	173,51	42,08	300,44
A15	331,99	516,34	376,97	607,65	431,9846	743,26	42,08	952,07
i7	14,43	24,61	16,09	25,6	23,9104	38,83	42,08	52,85
Adreno	17,40	31,93	22,52	43,91	34,304	70,50	59,38	121,22

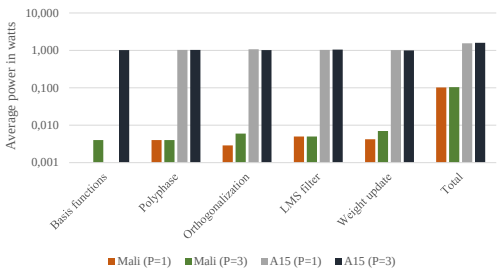


Fig. 7 Consumed power by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

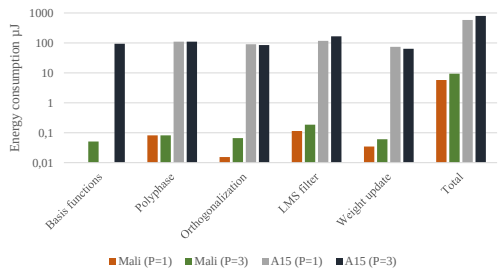


Fig. 8 Consumed energy by Mali and A15 running the linear and third order digital canceller kernels with input buffer length of 5120.

der canceller. As basis functions calculation is not required in the linear cases, only the measurements from the third order canceller are visible in the figure. The bars labelled as "total" correspond to the average power measured when running the complete digital canceller chain, which is slightly higher than the average power of all implemented blocks.

Comparing the results from the two processing platforms, it can be seen that A15 uses approximately 20 times more power compared to Mali when executing the same kernels. This can be explained by the higher clock frequency of the CPU (1.4 GHz compared to 600 MHZ), as well as the extra hardware on the CPU chip dedicated to the more general purpose computing. Increasing parallelism saves power by reducing the clock frequency for the same throughput. This reduces the switching activity, and more importantly the voltage which has quadratic effects to the power[1].

Mali consumes roughly an average of 104 mW running the third order digital canceller blocks with input buffer of 5120 samples. This can be considered negligible compared to the power consumption of e.g., an LTE receiver, which according to [19] is close to a couple of watts.

5.5 Energy consumption analysis

To better evaluate the feasibility of the proposed solution, it is also important to investigate the energy consumption of the implemented canceller. Since battery life depends on energy consumption, it is especially critical in hand-held devices. Furthermore, energy consumption comparison leads to fairer analysis compared to power, as we normalize the execution time.

We have used the measured average powers and the delays when processing 5120 samples for each kernel, and calculated the energy consumption. The results are shown in Fig 8, in which the missing bars correspond to linear cases, where basis function calculation is redundant. As delay increases with longer buffers and power consumption remains the same, it can be concluded that energy consumption increases with longer buffers.

Using higher power and slower execution of tasks has resulted in higher energy consumption by A15 compared to Mali. Total energy used by the third order canceller, implemented on Mali, and processing 5120 signal samples is approximately 9 μJ.

6 Conclusion

In this paper, we proposed a software-based implementation of a nonlinear digital SI canceller for full-duplex transceivers, using an adaptive cancellation algorithm, suitable for mobile-scale devices. To demonstrate the feasibility of a real-time SDR implementation, general-purpose low cost COTS processing platforms were selected, reducing the design time and costs compared

to custom hardware design. The implementation was carried out on multicore processors and software tailoring was done using OpenCL to achieve high performance. The ability of the designed canceller to sufficiently suppress the SI signal was shown using the data from a real full-duplex RF test-bench. Then the implementation was evaluated in terms of execution time, delay, power, and energy consumption to investigate the feasibility of a real-time digital canceller suitable for hand-held devices. The results showed that the Qualcomm Adreno 430, a mobile-scale GPU, and the Intel Core i7, a desktop CPU, can run the proposed digital canceller with the required sample rate for, e.g., a 20 MHz LTE band. However, there is a trade-off between the achievable SI cancellation rate and the system delay, as longer data buffers are required for high sample production rates. The results also showed that, although the delay is shorter with a real-time linear SI canceller, it converges much slower and may not reach sufficient SI cancellation levels. As a proof of suitability to mobile platforms, also power and energy consumption of the implemented digital canceller were measured on Exynos 5422 SoC, and the Mali-T628 GPU showed more promising results compared to the Cortex-A15 for a mobile-scale device. It can be concluded that a real-time programmable implementation of a nonlinear digital canceller can be realized using the Adreno GPU, on the user equipment side, and the Core i7 CPU on the base station side. In the continuation of this work, we aim at adopting a platform which would allow dividing the workload between the CPU and one or more GPUs, and as a result achieving higher sample production rates with shorter delays. Furthermore, another interesting topic for future work is to use OpenCL to program an FPGA for digital SI cancellation and compare performance results of GPU and multicore processors in terms of time, power, and energy consumption.

Acknowledgements This work was supported by Tampere University of Technology graduate school, and the Academy of Finland via projects "In-Band Full-Duplex Radio Technology: Realizing Next Generation Wireless Transmission" (304147) and "Making Programmable Logic Feasible in the Cloud." (297548).

References

1. CMOS power consumption and C_{pd} calculation (1997). URL `http://www.ti.com/lit/an/scaa035b/scaa035b.pdf`. Last accessed 08.04.2017
2. AghababaeeTafreshi, M., Koskela, M., Korpi, D., Jääskeläinen, P., Valkama, M., Takala, J.: Software defined radio implementation of adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio. In: IEEE Global Conference on Signal and Information Processing (2016)
3. Ahmed, E., Eltawil, A.M.: All-digital self-interference cancellation technique for full-duplex systems. IEEE Transactions on Wireless Communications **14**(7), 3519–3532 (2015). DOI 10.1109/TWC.2015.2407876
4. ARM Ltd.: ARM® Cortex® -A15 MPCore™ Processor (2011). URL `https://static.docs.arm.com/ddi0438/i/DDI0438.pdf`. Last accessed 08.04.2017
5. ARM Ltd.: The ARM® Mali™ Family of Graphics Processors (2013). URL `http://malideveloper.arm.com/downloads/events/2013/GDC/0319-11%20Mali%20Minibook_TB.pdf`. Last accessed 08.04.2017
6. Duarte, M., Dick, C., Sabharwal, A.: Experiment-driven characterization of full-duplex wireless systems. IEEE Transactions on Wireless Communications **11**(12), 4296–4307 (2012). DOI 10.1109/TWC.2012.102612.111278
7. Duarte, M., Sabharwal, A.: Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results. In: Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers, pp. 1558–1562 (2010). DOI 10.1109/ACSSC.2010.5757799
8. Duarte, M., Sabharwal, A., Aggarwal, V., Jana, R., Ramakrishnan, K.K., Rice, C.W., Shankaranarayanan, N.K.: Design and characterization of a full-duplex multiantenna system for WiFi networks. IEEE Transactions on Vehicular Technology **63**(3), 1160–1177 (2014). DOI 10.1109/TVT.2013.2284712
9. El-Rewini, H., Abd-El-Barr, M.: Advanced Computer Architecture and Parallel Processing. Wiley (2005)
10. Everett, E., Duarte, M., Dick, C., Sabharwal, A.: Empowering full-duplex wireless communication by exploiting directional diversity. In: Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers, pp. 2002–2006 (2011). DOI 10.1109/ACSSC.2011.6190376
11. 3rd Generation Partnership Project: Technical Specification Group Radio Access Network; Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced) (Release 14) (2017). URL `http://www.3gpp.org/ftp//Specs/archive/36_series/36.913/36913-e00.zip`. Last accessed 19.08.2017
12. Ghazi, A., Boutellier, J., Anttila, L., Juntti, M., Valkama, M.: Data-parallel implementation of reconfigurable digital predistortion on a mobile GPU. In: 2015 49th Asilomar Conference on Signals, Systems and Computers, pp. 186–191 (2015). DOI 10.1109/ACSSC.2015.7421110
13. Grayver, E.: Implementing Software Defined Radio, 1 edn. Springer (2013)
14. Hardkernel co., Ltd.: ODROID-XU3. (2013). URL `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127`. Last accessed 08.04.2017
15. Harris, P.: The mali GPU: An abstract machine (2014). URL `https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core`. Last accessed 08.04.2017
16. Heino, M., Korpi, D., Huusari, T., Antonio-Rodriguez, E., Venkatasubramanian, S., Riihonen, T., Anttila, L., Icheln, C., Haneda, K., Wichman, R., Valkama, M.: Recent advances in antenna design and interference cancellation algorithms for in-band full duplex relays. IEEE Communications Magazine **53**(5), 91–101 (2015)

17. Hong, S., Brand, J., Choi, J.I., Jain, M., Mehlman, J., Katti, S., Levis, P.: Applications of self-interference cancellation in 5G and beyond. IEEE Communications Magazine **52**(2), 114–121 (2014)
18. Intel Corporation: Intel® Core™ i7 Processor Family for LGA2011 Socket (2014). URL `http://www.intel.com/content/www/us/en/processors/core/4th-gen-core-i7-lga2011-datasheet-vol-1.html`. Last accessed 08.04.2017
19. Jensen, A.R., Lauridsen, M., Mogensen, P., Srensen, T.B., Jensen, P.: LTE UE power consumption model: For system level energy and performance optimization. In: IEEE Vehicular Technology Conference (VTC Fall), pp. 1–5 (2012). DOI 10.1109/VTCFall.2012.6399281
20. Khronos OpenCL Working Group: The OpenCL Specification, version 2.0 (2015). URL `https://www.khronos.org/registry/cl/specs/opencl-2.0.pdf`. Last accessed 08.04.2017
21. Kolodziej, K.E., McMichael, J.G., Perry, B.T.: Multitap rf canceller for in-band full-duplex wireless communications. IEEE Transactions on Wireless Communications **15**(6), 4321–4334 (2016). DOI 10.1109/TWC.2016.2539169
22. Korpi, D., AghababaeeTafreshi, M., Piilila, M., Anttila, L., Valkama, M.: Advanced architectures for self-interference cancellation in full-duplex radios: Algorithms and measurements. In: 2016 50th Asilomar Conference on Signals, Systems and Computers, pp. 1553–1557 (2016). DOI 10.1109/ACSSC.2016.7869639
23. Korpi, D., Choi, Y.S., Huusari, T., Anttila, L., Talwar, S., Valkama, M.: Adaptive nonlinear digital self-interference cancellation for mobile inband full-duplex radio: Algorithms and rf measurements. In: IEEE Global Communications Conference, pp. 1–7 (2015). DOI 10.1109/GLOCOM.2015.7417188
24. Korpi, D., Tamminen, J., Turunen, M., Huusari, T., Choi, Y.S., Anttila, L., Talwar, S., Valkama, M.: Full-duplex mobile device: pushing the limits. IEEE Communications Magazine **54**(9), 80–87 (2016). DOI 10.1109/MCOM.2016.7565192
25. Li, K., Ghazi, A., Boutellier, J., Abdelaziz, M., Anttila, L., Juntti, M., Valkama, M., Cavallaro, J.R.: Mobile GPU accelerated digital predistortion on a software-defined mobile transmitter. In: 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 756–760 (2015). DOI 10.1109/GlobalSIP.2015.7418298
26. Li, K., Ghazi, A., Tarver, C., Boutellier, J., Abdelaziz, M., Anttila, L., Juntti, M., Valkama, M., Cavallaro, J.R.: Parallel digital predistortion design on mobile GPU and embedded multicore CPU for mobile transmitters. Journal of Signal Processing Systems (2017). DOI 10.1007/s11265-017-1233-y. URL `https://doi.org/10.1007/s11265-017-1233-y`
27. Mikhael, M., van Liempd, B., Craninckx, J., Guindi, R., Debaillie, B.: An in-band full-duplex transceiver prototype with an in-system automated tuning for rf self-interference cancellation. In: 1st International Conference on 5G for Ubiquitous Connectivity, pp. 110–115 (2014). DOI 10.4108/icst.5gu.2014.258118
28. Qualcomm Technologies: Snapdragon 810 processor product brief (2015). URL `https://www.qualcomm.com/media/documents/files/snapdragon-810-processor-product-brief.pdf`. Last accessed 08.04.2017
29. Sabharwal, A., Schniter, P., Guo, D., Bliss, D.W., Rangarajan, S., Wichman, R.: In-band full-duplex wireless: Challenges and opportunities. IEEE Journal on Selected Areas in Communications **32**(9), 1637–1652 (2014). DOI 10.1109/JSAC.2014.2330193
30. Tuttlebee, W. (ed.): Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations, 1 edn. Wiley (2004)
31. Widrow, B., McCool, J.M., Larimore, M.G., Johnson, C.R.: Stationary and nonstationary learning characteristics of the lms adaptive filter. Proceedings of the IEEE **64**(8), 1151–1162 (1976). DOI 10.1109/PROC.1976.10286

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4254-1

ISSN 1459-2045