TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY
Julkaisu 604 • Publication 604

Lasse Harju

Programmable Receiver Architectures for Multimode Mobile Terminals

Tampere 2006

Tampereen teknillinen yliopisto. Julkaisu 604
Tampere University of Technology. Publication 604

Lasse Harju

Programmable Receiver Architectures for Multimode Mobile Terminals

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 11th of August 2006, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2006

ISBN 952-15-1618-6 (printed)
ISBN 952-15-1833-2 (PDF)
ISSN 1459-2045

ABSTRACT

This thesis considers the design of a programmable baseband receiver platform for WCDMA and OFDM mobile terminals. The design challenges introduced by the evolution of wireless systems are highlighted and design methodologies deployed in the platform development are introduced. The receiver algorithms of WCDMA and OFDM receivers are summarized and potential processor based architectures for implementing these algorithms are studied.

The Espresso platform is composed of a RISC processor core and three coprocessors. The coprocessor provide the functions needed to implement the WCDMA and OFDM receiver algorithms. The key of the coprocessor approach is the exploitation of the computational similarities of the WCDMA and OFDM receiver algorithms. This enables effective reuse of hardware resources between the WCDMA and OFDM modes of the receiver. The RISC processor is used to initiate the coprocessor functions and to implement symbol rate channel estimation and equalization tasks. The interconnection between the host processor and the coprocessors is realized with a dedicated coprocessor bus which reduces the communication overhead typically associated with memory mapped coprocessor.

The programming interface of the platform is implemented with a set of coprocessor functions. Typically application-speci c processors require low-level programming which affects negatively to the software development ef cienc y. The programming interface of the proposed platform is implemented with standard C-language which enables productive software development.

The platform architecture and the programming interface constitute a template baseband receiver architecture that can be employed in WCDMA and OFDM receivers. The hardware and the software can be ne tuned to the target application without affecting each other as long as the programming interface is kept unchanged. Thus, the platform enables effective reuse of existing hardware and software implementations.

PREFACE

The research work for this thesis was carried out in the course of the years 2000-2006 in the Institute of Digital and Computer Systems at Tampere University of Technology.

I would like to thank my supervisor Professor Jari Nurmi for his expertise and guidance throughout my research. I am truly grateful to my mentor Dr. Mika Kuulusa for his inspiration, support, and friendship. I would also like to thank Dr. Mikko Valkama for his valuable input. Special thanks to the colleagues I have had the pleasure of working with and who have provided me mental and technical support in the course of past few years: Teemu Taskinen, M.Sc., Simo Lehto, M.Sc., Heikki Hurskainen, M.Sc., Timo Rintakoski, M.Sc., and Tapio Ristimäki, Dr.Tech. I would also like to thank my thesis reviewers Kimmo Kuusilinna, Dr.Tech., and John Glossner, Ph.D.

The research work has been funded by Nokia Research Center, The European Union, and Elekrobit, Ltd. I gratefully acknowledge the support from the Nokia Foundation, the Finnish Cultural Foundation, Walter Ahlström Foundation, the HPY Research Foundation, the Foundation of Advancement of Technology, and Tuula and Yrjö Neuvo Foundation.

Finally, I would like to thank my parents Pekka and Aino Harju, my sister Elina, and my grandmother Kaija Harju for their encouragement and kind support.

Thank you Mia for your love, support, and understanding.

Tampere, June 2006

Lasse Harju

TABLE OF CONTENTS

LIST OF PUBLICATIONS

This thesis is composed of two parts. Part I contains argumentation of the research and Part II contains the conference and journal publications published during the course of the work. In the subsequent chapters these publications will be referred to as [P1], [P2], [P3],..., [P7].

[P1] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Rake Receiver Architecture for WCDMA Mobile Terminals,” in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications*, Tao Yuan, Taiwan, Mar. 2001, pp. 9-12.

[P2] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver,” in *Proc. IEEE Workshop on Signal Processing Systems*, San Diego, CA, USA, Oct. 2002, pp. 177–182.

[P3] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver,” in *The Journal of VLSI Signal Processing*, Springer Science, vol. 39, no 1–2, pp. 147–160, Apr. 2005.

[P4] L. Harju and J. Nurmi, “A Baseband Receiver Architecture for UMTS/WLAN Interworking Applications,” in *Proc. IEEE International Symposium on Computers and Communications*, Alexandria, Egypt, June 2004, vol. 2, pp. 678–685.

[P5] L. Harju and J. Nurmi, “A Programmable Baseband Receiver Platform for WCDMA/OFDM Mobile Terminals,” in *Proc. IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, Mar. 2005, vol. 1, pp. 33–38.

[P6] L. Harju and J. Nurmi, “A Synchronization Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations,” in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2005, pp. 141–145.

[P7] L. Harju and J. Nurmi, “A Demodulation Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations,” in *Proc. NORCHIP Conference*, Oulu, Finland, Nov. 2005, pp. 66–69.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Program
3GLTE	Third Generation Long Term Evolution
AAA	Authentication, Authorization, and Accounting
ADC	Analog-to-Digital Converter
API	Application Programming Interface
ASIC	Application-Speci c Integrated Circuit
ASIP	Application-Speci c Instruction Set Processor
AWGN	Additive White Gaussian Noise
BPF	Band-Pass Filter
CCK	Complementary Code Keying
CPICH	Common Pilot Channel
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DPCCH	Dedicated Physical Control Channel
DPCH	Dedicated Physical Channel
DSP	Digital Signal Processor

DSSS	Direct Sequence Spread Spectrum
EDA	Electronic Design Automation
EDGE	Enhanced Data Rates for GSM Evolution
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
GPRS	General Packet Radio System
GSM	Global System for Mobile Communications
HSDPA	High-Speed Downlink Packet Access
HSDSCH	High-Speed Downlink Shared Channel
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
IP	Intellectual Property
ISA	Instruction Set Architecture
ISM	Industrial, Scienti c, and Medical
LNA	Low Noise Ampli er
LS	Least Squares
LTE	Long Term Evolution
MAC	Multiply Accumulate
OFDM	Orthogonal Frequency Division Multiplexing
OVSF	Orthogonal Variable Spreading Factor
PDG	Packet Data Gateway

PSCH	Primary Synchronization Channel
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
RTOS	Real Time Operating System
SDF	Single-path Delay Feedback
SDR	Software De ned Radio
SIMD	Single-Instruction Multiple-Data
SoC	System-on-Chip
SSCH	Secondary Synchronization Channel
UMTS	Universal Mobile Telecommunication System
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLIW	Very Long Instruction Word
WAG	WLAN Access Gateway
WCDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Part I

INTRODUCTION

1. INTRODUCTION

Wireless communications are evolving towards multistandard systems. In the future, the same wireless applications can be accessed alternatively through multiple wireless networks using a single wireless terminal. The rst step towards multistandard systems will be taken by the forthcoming interworking between second and third generation cellular networks and wireless local area networks (WLAN). The bene t of this interworking is that the strengths of the two network types are combined, i.e., high mobility can be provided through the cellular networks, and high data throughput through the WLAN. WLAN networks are becoming very popular at public sites, such as airports, coffee shops, libraries, etc. Thus, interworking between WLANs and cellular networks provides a very ef cient way of offering increased throughput to wireless subscribers, and at the same time, a way of decreasing the load of the cellular networks.

The multistandard functionality imposes several challenges for network and mobile terminal implementations, as well as to the service providers. The biggest challenge for the network implementations is to handle the mobility of the users between the heterogeneous networks. The concern of service providers is the management of a common billing system for the usage of the multiple networks. In a multistandard environment, the users have to be equipped with mobile terminals that can operate in multiple wireless networks. Thus, the challenge for the mobile terminal implementations is to integrate several radio technologies into a single device. Utilizing dedicated transmitter and receiver architectures for each radio technology would result in an intolerable increase in physical size, weight, and power consumption. Thus, the key in multimode mobile terminal implementations is to share components and processing resources between the different radio technologies.

While the complexity of the mobile terminals is increasing due to the evolution of wireless communications, the highly competitive market forces mobile terminal manufacturers to release new products at shorter time intervals and with lower prices. The con ict between increasing complexity and shrinking time-to-market has lead manufacturers to adopt new design methodologies for hardware and software. The main goal of these new design methodologies is to bene t from the similarities between software and hardware solutions of similar product families. This goal can be achieved by designing software and hardware architectures that are reusable within a restricted eld of application. Another trend in mobile terminals, and embedded systems in general, is the increasing role of software. Some manufacturers prefer software implementations because of the shorter lead-time of software development and because necessary redesigns are much cheaper to carry out with software. As a result of the multimode functionality, software is utilized increasingly also in the baseband section of the transceivers.

Programmable receiver and transmitter solutions have already been utilized successfully in second and third generation mobile terminals, particularly in the symbol rate processing. The symbol rate processing involves typically interleaving and channel coding in the transmitter, and channel estimation, deinterleaving, and channel decoding in the receiver. The sample rate processing on the other hand, has typically required dedicated hardware solutions. This is mainly because of the high sampling rates and complex signal processing, that is especially computationally intensive at the receiver end. The sample rate processing at the receiver involves typically synchronization and demodulation algorithms. One of the biggest challenge for multimode mobile terminal implementations is to extend the programmability also to the sample rate processing.

One of the key technologies in mobile terminals are digital integrated circuits (IC). Modern mobile terminals utilize ICs that are very complex system-on-chip (SoC) solutions, composed of several processors, interconnection networks, application speci c processing units, recon gurable hardware, memories, peripherals, etc. Because software design has an increasingly important role in saving design time and costs, the programmability of the SoC solutions for mobile terminal implementations is of paramount importance. However, the programmability itself is not enough as the programmability should be realized in a way that enables effective software development and software reuse. In practice this means that usage of low-level programming languages should be avoided.

1.1 Objective and Scope of Research

The main objective of the research presented in this thesis is to design a programmable baseband receiver platform for multimode mobile terminals. The goal is to extend the programmability also to the sample rate processing without compromising the energy efficiency of the architecture. The approach for achieving this target is to use application-specific coprocessor accelerators to boost the performance of a reduced instruction set computer (RISC) processor. Reconfigurable hardware or low-level circuit design issues are not considered in the research. The radio technologies considered in the multimode operation are wideband code division multiple access (WCDMA) and orthogonal frequency division multiplexing (OFDM).

Another very important motivation for the research is to develop a programming interface for the platform that enables software development without low-level programming. Low-level programming affects negatively to the software development efficiency and also restricts the reuse of existing software. The approach for achieving the programmability is to abstract the implementation details of the platform architecture behind a programming interface in order to make the software less machine dependent. Embedded software development is not studied in detail.

1.2 Main Results

The main result of the research presented in this thesis is the Espresso platform. The platform comprises a template baseband receiver architecture, composed of a processor core and three coprocessors, and a programming interface for the coprocessors. The presented platform enables software implementation of baseband procedures, including the sample rate processing, and minimizes the need for low-level programming. Thus, the platform enables programmable multimode receiver implementations, with support for efficient software development.

1.3 Outline of Thesis

This thesis is composed of two parts: introduction and publications. The first part is organized as follows. In chapter 2, the main characteristics of future wireless systems are presented. In chapter 3, the various design challenges related to commercial wireless products are highlighted and the potential design methodologies for software and hardware are summarized. In chapter 4, the receiver algorithms of WCDMA and OFDM systems are studied in detail. Chapter 5 presents architectural alternatives that can be employed to build programmable baseband receiver architectures. The Espresso architecture is presented in chapter 6 and simulation and synthesis results are given in chapter 7. Conclusions are drawn in chapter 8. Summary of the publications included in the second part of the thesis is given in chapter 9.

2. FUTURE WIRELESS COMMUNICATIONS

Wireless communications has been one of the most successful sectors of consumer electronics for over a decade now. The increase in the number of wireless subscribers since the introduction of second generation systems has been phenomenal [1]. Because of the commercial success, the market for wireless products is also a very competitive one. Manufacturers are ghting for new customers by releasing products that are smaller, cheaper, and include more features. As a result, wireless communications has been a major driving force behind the evolution of cutting edge embedded systems and semiconductor technologies. The future of wireless communications is steered not only by the user's desire for faster and ubiquitous wireless connectivity but also by industry forces that try to maintain the economical success of the previous wireless generations. In this chapter the future directions of wireless communications are studied and the technical challenges resulting from this evolution are highlighted.

2.1 Next Generation Wireless Systems

The transition from third generation (3G) wireless systems to fourth generation (4G) systems is not likely to be as clear as the transition from second generation (2G) systems to 3G. The evolution of wireless communications will carry on in two directions: evolution of the current 3G networks and convergence of cellular and ad hoc networks [2]. The convergence of cellular networks and short range wireless systems operating in the unlicensed frequency bands is to be expected because it provides a convenient way to provide high data rate connections to wireless subscribers and an ef cient approach for increasing the capacity of cellular networks [3]. In the future, wireless personal area networks (WPAN), such as Bluetooth, can also be used as a complementary access medium to wireless services. A future addition to the WPAN category will be wireless technologies utilizing ultra wideband (UWB) technology. The features of different wireless technologies are listed in Table 1.

Table 1. *Comparison of cellular and short range wireless technologies.*

	3G	HSDPA	3G LTE	
Frequency Band	2 GHz	2 GHz	2 GHz	
Channel Bandwidth	5 MHz	5 MHz	5, 10, 15, or 20 MHz	
Physical Layer	WCDMA	WCDMA	WCDMA/OFDM	
Range	1-3 km	1 km	5 km	
Data Throughput	384 kbps	14 Mbps	100 Mbps	
	802.11b	802.11a	UWB	Bluetooth
Frequency Band	2.4 GHz	5 GHz	3.1-10.6 GHz	2.4 GHz
Channel Bandwidth	20 MHz	20 MHz	500 MHz	2 MHz
Physical Layer	DSSS	OFDM	Multiband OFDM	Frequency Hopping
Range	100 m	100 m	4 m	30 m
Data Throughput	11 Mbps	54 Mbps	480 Mbps	3 Mbps

New wireless applications and integration of different types of networks draw several requirements for wireless systems. The biggest challenge for the network implementations is to handle the mobility of the users between the heterogeneous networks. The concern of service providers is the management of a common billing system for the usage of the multiple networks. The most relevant requirement concerning the work presented in this thesis is that the users need to be equipped with terminals that can operate in multiple networks. Dual-band, tri-band, and quad-band handsets already exist for WCDMA and GSM800/900/1800/1900 networks, and many terminals include also infrared and Bluetooth technologies. In the future, the same handset has to support a richer set of physical layers, e.g., GSM, EDGE, WCDMA, 802.11b, 802.11a, Bluetooth, and UWB. In addition to supporting the physical layers, multimode terminals also have to be able to detect the available networks, and select the network that best suites the user's transfer requirements.

2.1.1 Evolution of 3G Systems

Third generation wireless systems, based on wideband code division multiple access (WCDMA) are currently being deployed around the world. In Europe, 3G systems are known as universal mobile telecommunication system (UMTS), which is standardized by the 3rd Generation Partnership Program (3GPP). WCDMA utilizes direct sequence spread spectrum (DSSS) physical layer with 3.84 MHz chip rate, and quadrature pulse shift keying (QPSK) or 16 quadrature amplitude (16-QAM) modulation [4]. The channel bandwidth of WCDMA is 5 MHz, and in the frequency division duplex (FDD) mode it is centered at 1.92-1.98 GHz frequency band in the uplink (UL) direction and 2.11-2.17 GHz frequency band in the downlink (DL) direction. WCDMA technology enables higher data rates compared to 2G systems and improves the spectral ef cienc y. Spectral ef cienc y is essential as the traf c loads of wireless systems are only expected to increase in the future. Another key advantage over 2G systems is the improved service e xibility [5]. The properties of the wireless connection can be varied according to the quality of service (QoS) requirements of the application being used [6]. The theoretical peak data rate of 3G systems is 2 megabits per second (Mbps), but the maximum data rate provided by the rst 3G networks is 384 kilobits per second (kbps).

HSDPA

The data rates of 3G systems will be improved by the forthcoming high speed downlink packet access (HSDPA) [7]. HSDPA employs high-speed downlink shared channel (HSDSCH) that is a common resource to all users in a single cell. The HSDSCH resources are divided into time slots and code channels, which are allocated dynamically to the users at 2 ms time intervals. By allocating the shared resources for a single user, data rates up to 14 Mbps can be achieved. The limitation of HSDPA is that the high data rates can only be achieved at proximity to the base station. This is because HSDSCH is transmitted at constant power, and hence, the interference levels at cell edges may be too high to fully utilize the shared resources.

3G LTE

3GPP has launched the standardization work of long-term evolution (LTE) 3G systems [8]. The target of this work, also referred to as Super 3G, is to develop a radio access technology optimized for packet based traf c that achieves signi cantly lower latency, higher data rates, broader coverage, and better spectral ef cienc y than current 3G systems. The target peak rate of Super 3G is 100 Mbps in the downlink direction and 50 Mbps in the uplink direction. The higher data rates will be achieved by employing orthogonal frequency division multiplexing (OFDM) and multiantenna techniques [9]. 3G LTE also will employ spectrum e xibility by adapting the employed channel bandwidth according to the traf c characteristics.

2.1.2 Convergence of 3G and WLAN Networks

Wireless local area networks were originally intended to be used for establishing temporary computer networks and for providing an alternative for wired computer networks in buildings where the necessary wiring could not be installed. Currently WLANs are becoming very popular at public sites, such as cafes, libraries, and airports, to provide internet access to customers with laptops. The convergence with cellular networks enables these WLAN hotspots to be employed as an alternative access medium to the core network of the cellular system.

A majority of the current WLANs are based on the IEEE 802.11b standard that enables data rates up to 11 Mbps and approximately 100 m operating range [10]. This technology is often referred to as Wi-Fi (wireless delity). 802.11b utilizes direct sequence spread spectrum (DSSS) technique with 11 MHz chip rate and complementary code keying modulation (CCK). 802.11b networks operate in the 2.4 GHz unlicensed industrial, scienti c, and medical (ISM) band. In the future, the available data rates will be increased to 54 Mbps by WLANs based on the IEEE 802.11a and IEEE 802.11g standards that utilize an OFDM physical layer [11, 12]. 802.11a and 802.11g networks are operated at the 5 GHz and 2.4 GHz frequency bands, respectively. The principle of OFDM is that the high data rate signal is divided into parallel lower data rate signals that are transmitted on dedicated subcarriers. Each of the subcarriers is modulated using phase shift keying (PSK) or quadrature amplitude modulation (QAM).

The interworking between 3G and WLANs enables the 3G core network to be accessed through a WLAN network. The standardization is currently ongoing in 3GPP [13, 14]. A simpli ed block diagram of the interworking architecture is depicted in Fig 1. The WLAN network is connected to the core network through a WLAN access gateway (WAG) and a packet data gateway (PDG). In addition to the depicted blocks, additional network elements are also needed for authentication, authorization, and accounting (AAA) functions.

Six different scenarios for the interworking between 3G and WLANs are outlined in [15]. The actual interfaces between the WAG, PDG, and the core network are quite different for these scenarios. In the simplest interworking scenario, the only integration between the networks is common billing and customer care. A 3G subscriber can access internet services through a WLAN network operated by the 3G service

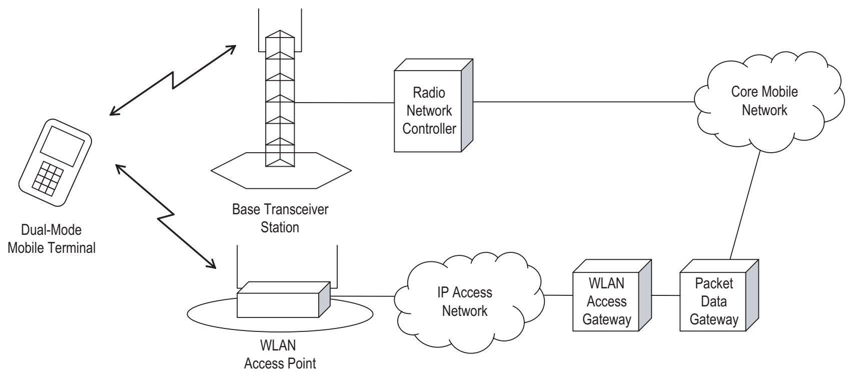


Fig. 1. *The interworking architecture for 3G networks and WLANs.*

provider, and the WLAN access charge is added to the 3G bill of the subscriber. At its full extent, the interworking enables access to both packet switched and circuit switched services of 3G through WLAN networks. A subscriber equipped with a dual-mode mobile terminal can receive multimedia messages, browse the internet, and make voice calls through either network and move about the networks without a notable difference in the service quality.

2.2 Software Radio

The evolution of wireless communications is likely to proceed towards the software radio concept, introduced in the early 1990s [16–18]. The original idea of the software radio was to free wireless communications from the limitations drawn by wireless standards. A software radio receiver, in its original form, consists of a multi-band antenna, a band-pass lter , a low-noise ampli er (LNA), a wideband analog-to-digital converter (ADC), and a powerful digital signal processor (DSP), as depicted in Fig. 2. The radio frequency signal picked up by the multiband antenna is band-pass ltered and immediately converted to digital domain. All the radio frequency, intermediate frequency, baseband, and protocol processing are then executed in the digital domain on a single processor. This fully programmable transceiver would enable adaptation to a wireless technology through software downloads [18]. Upon entering a previously unknown network, the mobile terminal could seamlessly adapt to the air interface and protocol stack of the network by downloading the necessary software without user involvement. In addition to ubiquitous coverage, this approach would

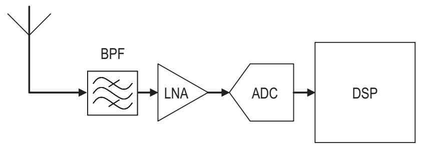


Fig. 2. *Architecture of a software radio receiver.*

enable more ef cient usage of spectrum resources. The terminal could monitor the available frequency resources through spectrum analysis, and select the appropriate frequency range, bandwidth, modulation, coding, and error corrections according to the channel conditions. Ideally, frequency resources would be shared between wireless systems, and the resources would be managed dynamically. This methodology is also known as cognitive radio [19].

Since its introduction, the software radio term has been used rather loosely in various contexts. The term software de ned radio (SDR) has emerged as a more practical view of the original concept. The Software De ned Radio Forum has speci ed ve categories, or tiers, that de ne variants of software radio [20]. Terminals that are implemented completely with hardware belong to tier 0, a category that is not considered as software radio at all. Terminals belonging to tier 1 include control functions implemented in software but do not have the ability to change modulation method or operation frequency without changing the hardware. Tier 2 is the de nition of software de ned radio. In this category, terminals are capable of changing operating frequencies by switching between parallel receiver front-ends and executing different modulation techniques under software control. In tier 2, it is also required that new modulation techniques can be added to terminals through software updates. In tier 3, the down-conversion is done completely in the digital domain, which requires that the analog-to-digital conversion is done directly after the antenna, as depicted in Fig. 2. Tier 4 is the de nition of an ultimate software radio. Terminals belonging to this category have no limitation in operating frequency, channel bandwidth, or modulation technique, and they are capable of switching between air interfaces instantly. In addition, the physical size, weight, and power consumption are at the level of today's mobile terminals.

3. DESIGN CHALLENGES AND METHODOLOGIES

Commercial wireless products form a segment of embedded systems that are characterized by a combination of features that make their design particularly challenging. Modern mobile terminals include a very rich set of features and applications, e.g., multiple radio technologies, games, and media players, that require a lot of processing power. Real time constraints are also essential as reactive applications and protocol processing require very accurate timing. Moreover, there exist hard limits for power consumption, price, weight, shape, and size of the products. The evolution of wireless standards also introduces design challenges as small additions and re nements are made continuously to the standards that affect the implementation of the mobile terminals. Ultimately, the goal of any embedded system company is to minimize development time and costs. Development time is largely dependent on the complexity of the design, head count of the personnel, availability of legacy designs, and ef cienc y of the design methodologies used. Correspondingly, development costs are made up of labor costs and infrastructure costs, such as design tools licenses, testing equipment etc. Various other costs, such as manufacturing, marketing, and administration affect the total production costs of a product. In this chapter, the design challenges affecting the development costs of future wireless products are covered and potential hardware and software design methodologies are introduced. Although the design challenges covered in this chapter apply for all type of embedded systems and all domains inside an embedded system, the focus of this text is on system-on-chip (SoC) solutions targeted for mobile terminals.

3.1 Design Challenges

A number of design challenges speci c for future wireless terminals are highlighted in the following sections.

3.1.1 Time-to-Market

Wireless products have particularly strict time-to-market constraints. Because of the evolution of wireless communications and semiconductor technologies, higher data rates, new wireless services, better standby times, and new attractive features emerge constantly. Consequently, the life span of wireless products has shrunk and manufacturers are forced to release new products at shorter time intervals to maintain their market positions. This trend results in very tight time-to-market constraints for new wireless products, and it has become increasingly dif cult for the manufacturers to design products within the required time frame.

3.1.2 Flexibility

Small-scale redesign or updates are often required for SoCs. Redesigns may be necessary to cope with a change in the product speci cation or a minor design a w. In the case of wireless products, redesigns may be needed as the wireless standards evolve and the physical layer parameters and procedures are subject to changes. Moreover, utilization of new radio frequency and mixed-signal front-ends may affect the baseband section in a way that requires design updates. In such situations, design time and cost are likely to be minimized if the original architecture is e xible enough that the changes can be implemented with software updates.

Another important motivation for maximizing the e xibility of wireless terminal implementations has been the dramatic rise in the manufacturing costs of modern semiconductor technologies [21]. The non-recurring engineering costs of SoCs, i.e., costs resulting from a single production run, are reported to be several million dollars for the latest technologies. The best way to mitigate these costs is to increase the volume of a single production run, and thereby minimize the effect of the manufacturing costs per chip. However, this is not feasible for products that do not go for mass production, and thereupon, the only way to guarantee high-volume production is to employ the same hardware in several products. This kind of e xibility is best achieved with programmable architectures where the product differentiation can be done by software.

3.1.3 Complexity

The amount of features found in wireless products has increased dramatically within the last few years. As the available processing power and storage space have increased, new attractive user interfaces and features, such as video games and multimedia, have emerged in wireless products. The complexity increase caused by the new features concerns mainly applications processing but as a result of the multimode functionality, the baseband section is also subject to a considerable complexity increase.

When the complexity of the systems increase, the size of the design teams is likely to increase in proportion. In order to avoid this and to keep labor costs under control, the productivity of the design methodologies needs to be improved. Programmability is an ef cient way to increase productivity simply because of the faster turnaround time of software development and the smaller cost of redesigns. Furthermore, utilization of existing hardware and software components can boost the design productivity further. Designing hardware and software components for reusability does require additional design effort but when pursued systematically, reuse can decrease development time signi cantly .

3.1.4 Embedded Software

In response to the e xibility requirements and the increased system complexity the amount of software utilized in mobile terminal implementations has increased signi cantly . Hence, software development is becoming the biggest design effort of wireless terminals and the importance of the software development productivity is emphasized. To date, the methods for increasing software development productivity in embedded systems have been rather primitive because the design methodologies and tools used in software development of other application elds do not suit embedded software development. One reason for this mismatch is that embedded systems involve specialized requirements regarding real-time operation, safety issues, and power consumption. One of the most critical shortcomings of embedded software development practices is the lack of ef cient reuse methodologies [22]. The software written for embedded systems is typically so machine dependent that reuse is limited to ad hoc methodologies, e.g., copying parts of existing source code. Another symptom of the machine dependency is that the underlying hardware determines the software architecture to a large extent, and hence, embedded software designers cannot make use of the bene ts of careful architectural planning.

3.1.5 Summary of Design Challenges

From the analysis of the previous sections, the following challenges are observed. First, the implementation of wireless products will become more and more software dominant. The lead-time of software development is shorter and therefore design time and costs can be minimized by utilizing programmable architectures. Moreover, programmability provides a way to cope with evolving design speci cations. New features can be added and existing features can be changed with minimum costs. Second, reuse will be of paramount importance to boost the productivity of the design projects. Reuse of existing components facilitates both design and veri cation. Third, embedded software is becoming a major factor in the total design costs and new design methodologies to boost the development productivity are needed.

3.2 Design Methodologies

Although programmability and reuse are among the most important factors leading to higher design productivity, current design methodologies provide little support for them. Hence, design methodologies of both hardware and software have been extensively researched.

3.2.1 Platform Based Hardware Design

Platform based design methodology can be regarded as a natural next step from current hardware design methodologies. Currently, the dominant design approach of ICs is based on utilization of intellectual property (IP) blocks that are pre-designed functional entities, such as memories, processors, buffers, busses, interfaces etc. [23]. The IP blocks can be imported to the design either at synthesis stage (soft IP) or at place & route stage (hard IP). The platform based design methodology can be seen as a extension to the IP based approach, where also the basic architecture, i.e., interconnect network and communication mechanisms, is considered as an IP block.

A clear de nition of what constitutes a platform has not been made but in a strict sense it can be seen as an off the shelf implementation engine targeted for a speci c application eld [24]. An example of such a platform is the OMAP platform by Texas Instruments [25]. The OMAPV1030 is a single chip architecture composed of two processors accompanied by a library of essential software routines and reference designs, needed to implement a 2.5G GSM/GPRS/EDGE handset. This is an example of a platform that has a very restricted applications space, but at the same time, it is a very good example of how production volume can be increased by using a single platform in several products of the same application eld. Signi cant savings on the SoC development costs are achieved as product differentiation is done with software by implementing different types of user interfaces and different sets of features.

More generally, a platform can be viewed as a template architecture targeted for a speci c application eld, accompanied by a library of IP blocks (both hardware and software) that can be plugged into the template [23]. One of the most important goals of platform based design methodology is to raise the level of abstraction at the design o w entry stage and to streamline the design o w from speci cation to implementation. At the highest abstraction level the system can be modeled at transaction-level without any consideration of timing, parallelism, or any physical constraints [26]. Once the designer has selected an appropriate platform for the target application, the transaction-level model of application is mapped onto the platform components, effectively resulting in a software-hardware partition. The architecture is then tuned to the target application by setting the parameters for the hardware blocks and by writing the nal software for the programmable components. Although the design o w back-end has to be completed just as with any ASIC designs, development time and costs are saved because the implementation is constructed from a template architecture and IP block that are readily available and veri ed.

To boost the software development ef cienc y, an application programming interface (API) is necessary that hides the architectural details of the platform architecture from the programmer. This enables software to be written independent of the underlying hardware which, in turn, enables software reuse. Furthermore, a real time operating system and a set of device drivers may be running under the API.

3.2.2 Product Line Based Software Design

The evolution of wireless communications and semiconductor technologies creates similar challenges for embedded software development as they do for SoC development. The complexity of software, measured by the size of utilized program memory, used in embedded systems has increased dramatically within recent years [27, 28]. Another trend affecting the embedded software design is the growing diversity of products. Often, several products for the same market need to be released with different sets of features for different price ranges, and the same products may need to be released with minor differentiation in different countries. Because of the shorter lead-time of software development these types of differentiations are best to implement in software [29]. Some consumer electronics companies have responded to this by designing software for a family of products that have a common basic software architecture. A good example of a product family is GSM handsets. Products belonging to different product families often include many commonalities. For example, a DVD player and a digital TV set might belong to different product families but they both include an MPEG decoder. This has lead to the adoption of the product line based software development.

Product line based software design methodology is in many ways analogous to the platform based hardware design. However, its tenets are much more clearly de ned than those of platform based hardware design, and it has reached a state of maturity as several product line success stories already exist [30]. Products developed inside the product line are created by using the template software architecture and selecting components from a library of the product line. When the needed components are not available in the library they are developed in a disciplined manner that guarantees their reusability in future projects. In addition to the component library, the product line assets include detailed instructions for building a product using the library components, producing the required documentation, testing, etc.

Product developed inside a product line typically employ similar hardware architectures. The software architecture of a product line is largely dictated by the underlying hardware. This implies that the exact characteristics of the hardware architecture have to be available when the product line is being established. This con icts with the goal of platform based hardware design, where the system is rst captured with transaction-level models, and the split between hardware and software is x ed as late as possible in the design o w to allow exploration between alternative solutions. Therefore, software product lines of the future should be built on an abstraction of the underlying hardware platform rather than an existing physical implementation.

3.2.3 Future Design Tools

Reuse of existing designs is the basic tenet of both platform based hardware design and product line based software design. In the endeavor to achieve more ef- cient reuse, the role of electronic design automation (EDA) tool vendors will be signi cant. The importance of better EDA tools is addressed in the embedded systems roadmap [27], technology roadmap for software intensive systems [31], and international technology roadmap for semiconductors [21]. Although it is possible to achieve better reuse of hardware and software separately with the current design methodologies, the true challenge in the future is to combine the hardware and software domains and develop the system level methodologies. Similar design, veri cation, and synthesis tools that are available at register transfer level (RTL) today are needed at system level in the future. Design tools should provide designers means to verify the functionality of the design, and measure the performance and the cost of the architecture at system level. This would require information about the physical characteristics of the design already at high abstraction levels. Tool suites that combine the hardware and software domains and automate the mapping of the functional system level models onto the available software and hardware platforms will be needed.

3.2.4 Applying New Design Methodologies in Practice

Despite the appeal of new design methodologies in a theoretical context, their bene ts can be quite dif cult to measure in practice. This is a major obstacle for companies considering a paradigm change to current design methodologies. Besides the costs of purchasing new design tools and training personnel to apply the new design methodologies, a paradigm change may have a negative effect on the income of the company because it takes a period of time before pro table projects using the new methodology can be launched. One of the most important factors affecting the design methodologies employed is legacy designs. A complete paradigm change requires that also the legacy designs are transformed to support the new methodology. Hence the costs and effort to carry out the adoption of the new methodology are signi cant, and it would be important to companies to measure the pro tability of their investment.

Increasing design productivity is not only a product of the methodologies applied and EDA tools used. It is also affected by non-technical factors, such as the organizational structure of a company, the management style practiced, and the personal commitment of the employees [32]. Because the adoption of a new design methodology greatly affects how information o ws in a company, the personnel and the projects need to be managed in a way that best suites the design methodology. The company management also has to establish the guidelines and instructions that de ne how the design methodology is applied in practice. It also has to monitor that these instructions are followed and that the development of the core assets and the products support the long term interest of the company, i.e., the applicability of the core assets also in future projects.

4. BASEBAND PROCESSING IN WCDMA/OFDM RECEIVERS

Baseband processing in a wireless receiver includes typically synchronization, demodulation, channel estimation, and channel equalization procedures. Fundamentally, its task is to recover the transmitted symbol stream from the received signal that is distorted by the wireless channel, by exploiting the redundancy added to the signal at the transmitting end. The receiver algorithms needed in the baseband sections of WCDMA and OFDM receivers are studied in this chapter. The purpose of this study is to identify the computational structures that can be shared in both modes of a dual-mode WCDMA/OFDM receiver. All the presented WCDMA and OFDM parameters are according to 3GPP [33] and IEEE 802.11a [11] standards, respectively.

4.1 Dual-Mode WCDMA/OFDM Receiver Front-End

A receiver front-end comprises the analog radio frequency (RF) section and the mixed-signal section. The goal in multimode front-end designs is to share as many components as possible between the different modes of the receiver, and at the same time, achieve comparable performance to single-mode receivers. The main challenges of a dual-mode WCDMA/OFDM receiver front-end implementation are brie y introduced in the next sections.

4.1.1 RF-Section

A dual-mode WCDMA/OFDM receiver requires a front-end that is able to operate at two different frequency bands and channel bandwidths, and that supports two different duplex methods and several different modulation techniques. The bandwidth of a radio frequency OFDM signal is 20 MHz centered at 5 GHz, whereas the bandwidth of a WCDMA signal is 5 MHz centered at 2 GHz. A good topology for multimode receivers is the direct conversion topology that converts the radio signal straight

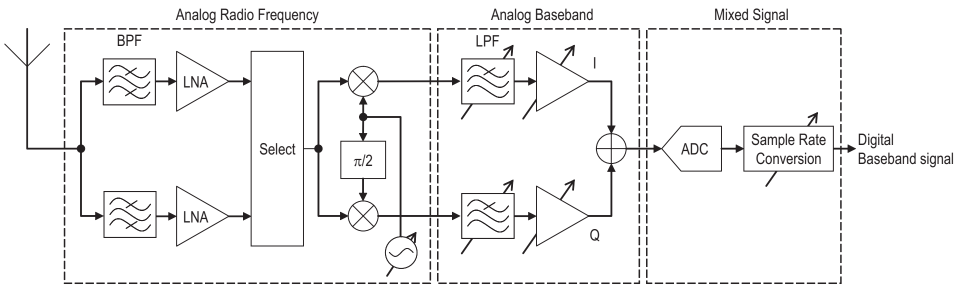


Fig. 3. *A block diagram of a dual-mode WCDMA/OFDM receiver front-end.*

to baseband, thus eliminating the components needed at the intermediate frequency stage in heterodyne architectures [34]. A typical solution in multimode receivers is to employ dedicated band-pass lters (BPF) and low-noise ampli ers (LNA) for each mode, and to use a shared mixer with a programmable oscillator frequency [35]. A block diagram of a theoretical dual-band WCDMA/OFDM receiver front-end is depicted in Fig. 3. Each frequency band of interest is separately band-pass ltered and ampli ed and one of the radio frequency signals is then down-converted in a single step by the mixer that features a tunable local oscillator. After the down-conversion, the signal is ltered with a tunable low-pass lter and then ampli ed with a variable gain ampli er . The depicted architecture is based on a dual-mode direct conversion RF front-end presented in [36] that comprises dedicated components for the antenna, channel selection lter , and LNA. A similar approach is used in a quad-mode front-end designed for GSM/GPRS/EDGE and WLAN terminals presented in [37].

4.1.2 Analog-to-Digital Conversion

The double-sided bandwidth of a complex baseband signal in WCDMA is 5 MHz, assuming ideal ltering of adjacent channels. Thus, according to Nyquist sampling theorem, the minimum sampling frequency for WCDMA receivers would be 5 MHz. However, it is convenient to choose the sampling frequency to be a multiple of the chip rate 3.84 Mcps, and hence the minimum sampling frequency for WCDMA is 7.68 MHz. In order to increase the resolution of the multipath estimation, oversampling is required. Furthermore, the effect of quantization noise is diminished when oversampling is utilized, and the complexity of the analogue anti-aliasing lter is reduced. The dynamic range of the analog-to-digital conversion (ADC) in WCDMA is typically 4-6 bits per sample [38].

In OFDM, the double-sided bandwidth of the complex baseband signal is 20 MHz, resulting in a minimum sampling frequency of 20 MHz. Although the requirements for multipath resolution are not that critical in OFDM, the other bene ts of oversampling apply similarly as in WCDMA. The needed dynamic range is also much larger in OFDM. In a WLAN chipset presented in [39], a sampling frequency of 80 MHz with a resolution of 10 bits per sample is utilized.

Considering the dual-mode implementation, the ADC requirements are clearly dictated by the OFDM characteristics. However, it is not energy ef cient to run the system at the OFDM sampling rate because it is likely that the OFDM connection is employed rarely compared to WCDMA. For this reason a sample rate conversion is included in the mixed signal section of the receiver front-end, as depicted in Fig. 3. The requirements of the ADC can be eased by making certain restrictions to the OFDM traf c. The high dynamic range requirement of OFDM receivers is partly caused by the modulation methods with dense constellations. By restricting the supported modulation methods to binary phase shit keying (BPSK) and quadrature phase shift keying (QPSK), the ADC requirements can be eased at the expense of the available data rate.

4.2 Digital Baseband Section

The digital baseband section comprises synchronization, demodulation, channel estimation and channel equalization blocks. High level block diagrams of the baseband sections of WCDMA and OFDM receivers are depicted in Fig. 4.

The Rake receiver is the key component in WCDMA receivers. As a result of multipath propagation, several copies of the transmitted signal with different delays, attenuations, and phases are picked up by the receiver antenna. A Rake receiver isolates the strongest multipath components from the received signal and combines them coherently [P3]. A Rake receiver comprises a multipath searcher, Rake ngers, a channel estimator, and a maximal ratio combiner. The actual number of Rake ngers is not speci ed by WCDMA speci cations but typically 4-8 ngers are employed [40–42]. Each nger includes resources for despreading a number of parallel data channels dedicated for the user, control channels of the active cell, or control channels of neighboring cells. The synchronization in WCDMA receivers is performed by the

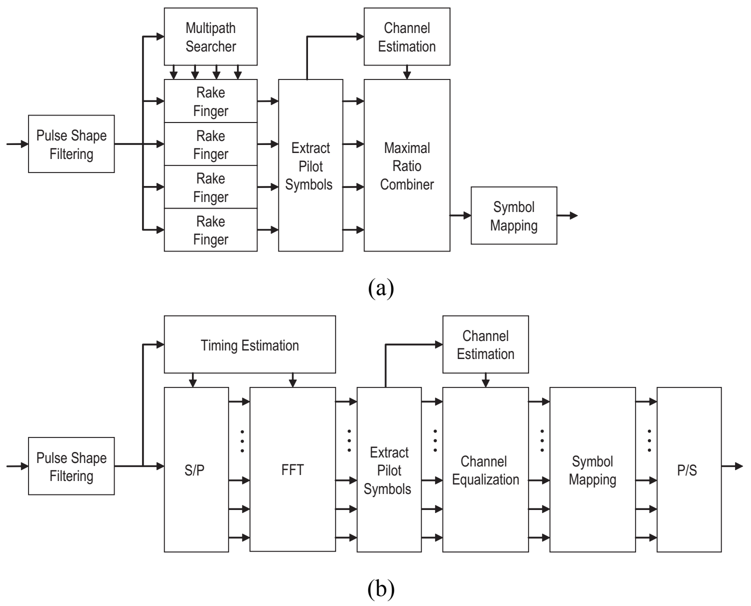


Fig. 4. *Block diagrams of the digital baseband sections of (a) WCDMA and (b) OFDM receivers.*

multipath searcher. It detects the strongest multipath components of the received signal and controls the Rake ngers accordingly. The demodulation is performed in the Rake ngers by correlating the received signal with a spreading code over a period corresponding to the spreading factor. In principle, the start indices of the Rake nger correlations are determined by the multipath searcher. After the demodulation, maximal ratio combining is applied to the symbol dumps from the ngers. In maximal ratio combining, the phases of the symbols are aligned and their amplitudes are weighted according to the complex tap coef cients acquired by the complex channel estimator. After the combining, decision of the transmitted symbol is made and the resulting bit stream is deinterleaved and decoded.

In OFDM, the demodulation is performed by applying 64-tap discrete Fourier transform (DFT) to the samples within a symbol window. Typically the transform is implemented in the form of fast Fourier transform (FFT). The synchronization task in OFDM involves detecting the symbol boundaries in the sample stream. A merit of OFDM systems is that they are less vulnerable to multipath fading than WCDMA systems. A guard period is inserted in front of each symbol to eliminate inter-symbol interference caused by the multipath fading [43]. Each symbol is cyclically extended

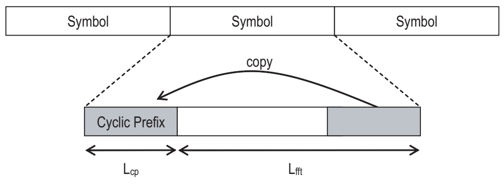


Fig. 5. *Illustration of the cyclic prefix in OFDM symbols.*

over this guard period as illustrated in Fig. 5. Once the symbol timing has been established, the samples within a symbol window are serial-to-parallel converted (S/P) and fed to the 64 FFT inputs. After the demodulation, pilot symbols are extracted from the FFT outputs and used to estimate the channel coef cients. After the channel equalization, the symbol mapping is performed. The bit stream is then parallel-to-serial converted (P/S) and sent to the outer receiver that performs deinterleaving, descrambling, and Viterbi decoding.

4.2.1 Baseband Signal Model

The general model of the received complex signal after down-conversion, sampling and pulse shape ltering can be expressed as

$$r(k) = \sum_{i=0}^{N_t-1} \alpha_i s(k-\tau_i) e^{j2\pi f_o k T_s} + n(k) \tag{1}$$

where $s(k)$ is the transmitted baseband signal, N_t is the number of multipath taps, α_i is the complex attenuation factor of the ith multipath, τ_i is the delay of the ith multipath (expressed as an integer multiple of samples), f_o is the offset between the oscillator frequencies of the transmitter and the receiver, and T_s is the sampling period. It is assumed that the channel conditions remain constant during the estimation period. The interference term $n(k)$ is the combination of various interference sources, such as signals transmitted from neighboring cells, signals of other users in the same cell, and thermal noise. It is assumed that the interference is Gaussian distributed.

In a typical WLAN environment multipath propagation can be omitted and an additive white Gaussian noise (AWGN) channel model can be used. Moreover, the frequency offset can be normalized to the frequency spacing of the OFDM system, and (1) can be reformulated as

$$r(k) = s(k-\tau) e^{j2\pi k\varepsilon/N_c} + n(k) \tag{2}$$

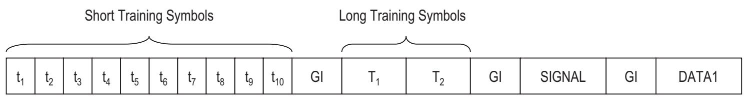


Fig. 6. *Structure of the OFDM packet preamble.*

where τ is the timing offset caused by the channel, ε is the normalized frequency offset, computed by dividing the original frequency offset by the frequency spacing of the OFDM subcarriers ($\varepsilon = f_o/\Delta f$), and N_c is the number of subcarriers.

4.3 Timing Synchronization

4.3.1 OFDM Timing Synchronization

Timing synchronization in OFDM systems is divided into packet detection and symbol timing estimation. The cyclic pre x makes OFDM systems less sensitive to timing errors because the timing estimate may vary within an interval bounded by the length of the cyclic pre x L_{cp} (see Fig. 5).

Packet Detection

The rst task of the synchronization procedure is to determine when a data packet has arrived. The detection can be made using the delay-and-correlate method, where the received signal is correlated against a delayed version of itself [44]. This method is effective when the received signal includes identical training symbols transmitted periodically. In 802.11a, the short training symbols in the preamble of a data packet can be used for packet detection. The data packet preamble is depicted in Fig. 6. The output of the delay-and-correlate algorithm $y(k)$ is given by

$$y(k) = \sum_{l=0}^{L-1} r(k+l)r^*(k+D+l) \tag{3}$$

where $r(k)$ is the received signal, $()^*$ is the complex conjugate operation, L is the length of the correlation, and the delay D is the distance of the two consecutive training symbols. The correlation $y(k)$ can be normalized by the received signal power during the correlation period computed as

$$p(k) = \frac{1}{2}\sum_{l=0}^{L-1} |r(k+l)|^2 + |r(k+D+l)|^2 \tag{4}$$

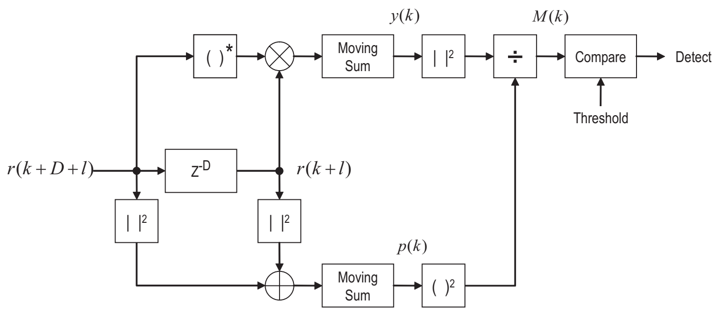


Fig. 7. *Signal flow representation of the OFDM packet detection algorithm.*

The decision metric is then computed as

$$M(k) = \frac{|y(k)|^2}{(p(k))^2} \tag{5}$$

The value of $M(k)$ is compared against a threshold value, and the detection is made when a correlation peak crosses this threshold. The packet detection algorithm is illustrated in Fig. 7. The packet detection algorithm presented in [45] uses this approach.

Symbol Timing Estimation

Symbol timing estimation is needed to nd the OFDM symbol boundaries. The estimation can be done with the delay-and-correlate approach by exploiting sequential training symbols or the cyclic pre x. An OFDM symbol timing estimator presented in [46] uses this approach. Alternatively, a matched lter approach can be employed [44]. In 802.11a, the long training symbols in the preamble of a data packet can be used for symbol timing estimation. The output of the matched lter $y(k)$ is then computed as

$$y(k) = \sum_{l=0}^{L-1} t^*(l) r(k+l) \tag{6}$$

where $t(l)$ is the training symbol. The index that gives the maximum value of $y(k)$ inside an observation window yields the symbol timing estimate.

$$\hat{\tau} = arg \max_k \{|y(k)|^2\} \tag{7}$$

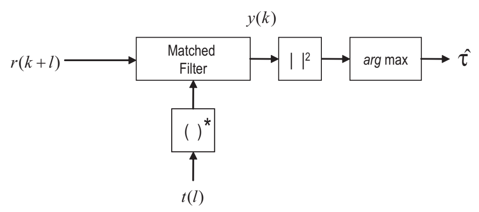


Fig. 8. *Signal flow representation of the OFDM symbol timing estimation algorithm.*

The beginning of the search window is determined by the packet detection and the length of the search window is determined by the longest expected propagation delay. A signal o w representation of the algorithm is depicted in Fig. 8.

4.3.2 WCDMA Timing Synchronization

In WCDMA receivers, synchronization is needed for frame and slot timing, as well as to determine the multipath pro le of the channel.

Cell Search

Cell search is the process in WCDMA systems where the mobile terminal synchronizes to the downlink scrambling code of the closest transmitting base station. The procedure is divided into three steps: slot timing synchronization, frame timing synchronization (scrambling code group identi cation), and scrambling code identi cation [40].

The rst step is similar to the symbol timing estimation in OFDM. The received signal is matched to the primary synchronization channel (PSCH) that is constructed of a pilot sequence transmitted at the beginning of every transmission slot, as illustrated in Fig. 9(a). The matched lter output is computed as in (6).

In the second step, the received channel is correlated with all 16 secondary synchronization channel (SSCH) sequences in parallel. These correlations are done at the beginning of each slot (rst 256 chips) over a period of 15 slots. This can be expressed as

$$y_i(n) = \sum_{l=0}^{L-1} c_i^*(l) r(l + \hat{\tau} + nL_s), \qquad i = 0, \ldots, 15, \quad n = 0, \ldots, 14 \tag{8}$$

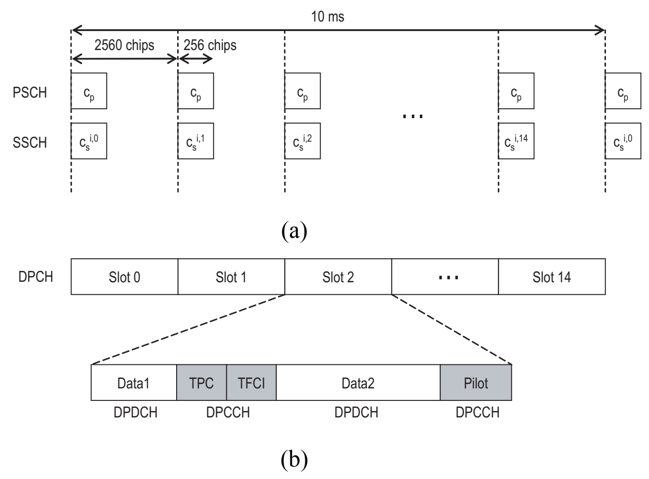


Fig. 9. *Slot structures of (a) downlink synchronization channel and (b) downlink dedicated physical channel.*

where $c_i(l)$ is the ith SSCH sequence, L is the length of the SSCH sequences, $\hat{\tau}$ is the index of the slot boundary identi ed in the rst stage, and L_s is the length of a slot measured in chips. After each correlation iteration, one of the 16 parallel correlations results in a signi cantly larger output than the others, and the indices of these correlators are stored as a sequence. The used scrambling code group is identi ed by matching this sequence to one of the 64 possible SSCH codewords.

In the third step, the code inside the code group is identi ed by correlating the received signal against each of the eight scrambling code candidates. The code resulting in the largest correlation output is selected as the scrambling code. The cell searcher implementation presented in [47, 48] employ the described basic principles.

Multipath Delay Estimation

The multipath estimation is often divided into two stages: acquisition and tracking. In the acquisition stage, the received signal and the locally generated codes are synchronized using the information from the cell search procedure as a starting point. In the tracking stage, the multipath searcher tracks the movement of the multipath taps inside a certain time window. In WCDMA, the pilot symbols transmitted on the downlink dedicated physical channel (DPCH) or common pilot channel (CPICH) can be used for this multipath delay estimation [49]. The structure of the downlink

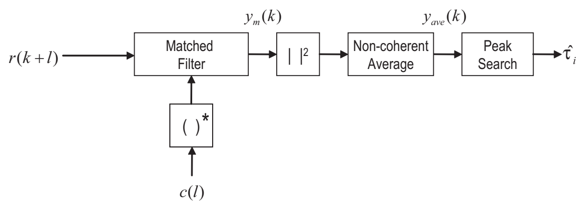


Fig. 10. *Signal flow representation of the WCDMA multipath delay estimation algorithm.*

DPCH is illustrated in Fig. 9(b).

The multipath estimation is done by matching the received signal to a known pilot sequence and detecting peaks in the matched lter output. In principle, the rst peak determines the acquisition point and following peaks that are within a certain window determine the other multipath components. The magnitude of a peak is proportional to the gain of the multipath, and the distance of a peak relative to the rst arrival gives a measurement of the path delay. In order to minimize the effect of noise and the interference caused by other users, the sequential estimation windows can be averaged non-coherently. This can be expressed as

$$y_{ave}(k) = \frac{1}{M}\sum_{m=0}^{M-1} |y_m(k)|^2 \tag{9}$$

where $y_m(k)$ is the kth element of the mth correlation window, computed as in (6). A signal o w representation of the multipath delay algorithm is depicted in Fig. 10. Many variations of the presented multipath searcher exist [50–52], but they all employ the same basic computations, namely matched ltering, non-coherent averaging, and peak searching. A multipath searcher implementation presented in [53] features an additional step after the peak selection that measures the power of the selected multipath candidates to improve the reliability of the estimates. The multipath delay estimation has to be performed at least at accuracy of one chip. Oversampling or interpolation is required if better multipath resolution is desired [51].

4.4 Frequency Offset Estimation

An offset between the oscillator frequencies of the transmitter and the receiver causes a rotation of demodulated symbols in the constellation. Consequently, the frequency offset can be estimated by observing the phase difference between two known symbols. Conveniently, the frequency offset can be estimated jointly with the timing estimation. The delay-and-correlate method of (3) can be applied for this purpose when identical training symbols are transmitted periodically. The correlation output is computed as in (3) and the frequency offset estimate $\hat{f}_o$ is computed from the arguments of the correlation output $y(k)$, at the index $\hat{\tau}$ that gives its maximum value of $y(k)$ within the observation window. This can be expressed as

$$\hat{f}_o = -\frac{1}{2\pi D T_s} \angle y(\hat{\tau}) \tag{10}$$

where T_s is the sampling period and D is the distance of the two training symbols measured in samples. A maximum likelihood frequency offset estimation algorithm for OFDM receivers presented in [45] uses this approach. In 802.11a, the short training symbols in the preamble of a data packet can be used for this purpose, and in WCDMA, common pilot channel (CPICH) can be employed [49].

4.5 Demodulation

4.5.1 WCDMA Demodulation

In WCDMA receivers, the demodulation is performed in the Rake ngers by correlating the received signal with a spreading code over a period corresponding to the spreading factor [54]. The starting index of the correlation is obtained from the multipath estimation. The output of the ith Rake nger can be expressed as

$$\hat{d}_i(n) = \sum_{l=0}^{L_{sf}-1} c_s^*(l + nL_{sf}) r(l + \hat{\tau}_i + nL_{sf}) \tag{11}$$

where L_{sf} is the spreading factor and $\hat{\tau}_i$ is the multipath estimate for the ith Rake nger . If parallel code channels are being employed, each of them is despread separately. The combined spreading and scrambling code $c_s(l)$ is given by

$$\begin{aligned} c_s(l) = (c_{sp}(l \bmod L_{sf}) \oplus c_{sc}^{(I)}(l \bmod L_f)) + \\ + j(c_{sp}(l \bmod L_{sf}) \oplus c_{sc}^{(Q)}(l \bmod L_f)) \end{aligned} \tag{12}$$

where $\oplus$ is the exclusive or operation, c_{sp} is the spreading code, $c_{sc}^{(I)}$ and $c_{sc}^{(Q)}$ are the real and imaginary parts of the scrambling code, respectively. The length of the frame L_f is 38 400 chips in 3GPP [49]. Because the scrambling and spreading codes are always sequences of ± 1, the multiplication and addition of each correlation stage are simpli ed. The multiplication in (12) is simpli ed to

$$\begin{aligned}(r^{(I)} + jr^{(Q)})(c_s^{(I)} - jc_s^{(Q)}) &= (r^{(I)}c_s^{(I)} + r^{(Q)}c_s^{(Q)}) + j(r^{(Q)}c_s^{(I)} - r^{(I)}c_s^{(Q)}) \\ &= \begin{cases} r^{(I)} + r^{(Q)} + j(r^{(Q)} - r^{(I)}), & \text{when } c_s^{(I)} = 0, \ \ c_s^{(Q)} = 0 \\ r^{(I)} - r^{(Q)} + j(r^{(Q)} + r^{(I)}), & \text{when } c_s^{(I)} = 0, \ \ c_s^{(Q)} = 1 \\ -(r^{(I)} - r^{(Q)}) - j(r^{(Q)} + r^{(I)}), & \text{when } c_s^{(I)} = 1, \ \ c_s^{(Q)} = 0 \\ -(r^{(I)} + r^{(Q)}) - j(r^{(Q)} - r^{(I)}), & \text{when } c_s^{(I)} = 1, \ \ c_s^{(Q)} = 1 \end{cases}\end{aligned} \qquad (13)$$

The chip value $+1$ is represented with a logic '0' and the chip value -1 is represented with a logic '1'.

4.5.2 OFDM Demodulation

In OFDM, the demodulation is performed by applying 64-point FFT. The reason why FFT is chosen over a straightforward DFT implementation by correlation is the reduced number of complex multiply-accumulate operations needed to complete a single instance of the algorithm. Using DFT, the *n*th symbol output of the *i*th subcarrier would be computed as

$$\widehat{D}_i(n) = \sum_{m=0}^{M-1} r(m + \hat{\tau} + n(L_{fft} + L_{cp}))W_M^{mi}, \qquad i = 0, \ldots, M-1 \qquad (14)$$

where M is the length of the transform, $\hat{\tau}$ is the symbol timing estimate, L_{fft} is the length of the FFT window (see Fig. 5), L_{cp} is the length of the cyclic pre x, and $W_M = e^{-j2\pi/M}$. It can be seen from (14) that one instance of the algorithm requires M^2 multiply-accumulate operations. The derivation of FFT from DFT can be found in [55].

4.6 Channel Estimation

The channel estimation algorithms are very similar in WCDMA and OFDM. In WLAN systems, it is commonly assumed that the channel is quasi-stationary, i.e., the channel conditions do not change during a data packet [43, 44, 56]. This means that the estimation is made using the pilot symbols in the preamble of a data packet, and used for the entire packet. In WCDMA, the assumption of a quasi-stationary channel does not apply because the signal bandwidth exceeds the coherence bandwidth of the channel, and the expected mobile speed is much higher than in typical WLAN scenarios.

4.6.1 WCDMA Channel Estimation

Assuming zero frequency offset and correct multipath delay estimates, the demodulated symbols from the each Rake nger can be expressed as

$$\hat{d_i}(n) = \alpha_i d(n) + w(n) \tag{15}$$

where $d(n)$ is the transmitted symbol, α_i is the complex attenuation of the ith multipath, $\hat{d_i}(n)$ is the output of the Rake nger assigned to that multipath, and $w(n)$ is additive noise. When the transmitted symbols are known, the channel estimate for the ith multipath can be computed as

$$\hat{\alpha}_i \approx \frac{\hat{d_i}(n)}{d(n)} \tag{16}$$

When quadrature phase shift keying (QPSK) modulation is used, the division in (16) can be substituted by multiplying the demodulated data with the complex conjugate of the reference symbol. The noisy channel estimates from each Rake nger are fed into a moving average lter . A signal o w representation of the algorithm is depicted in Fig. 11.

The pilots on the DPCCH or the CPICH are used for channel estimation in WCDMA. The difference between these is that the pilots transmitted on the CPICH are continuous, whereas the pilots on the DPCCH are transmitted only at speci c elds of a slot. If the DPCCH is used for channel estimation, interpolation is needed to obtain the estimates for the duration between the pilot symbols [57]. When the CPICH is used for the channel estimation, interpolation is needed to match the rate of the pilot symbols to the symbol rate of the data channel, as CPICH is transmitted with a constant spreading factor [58].

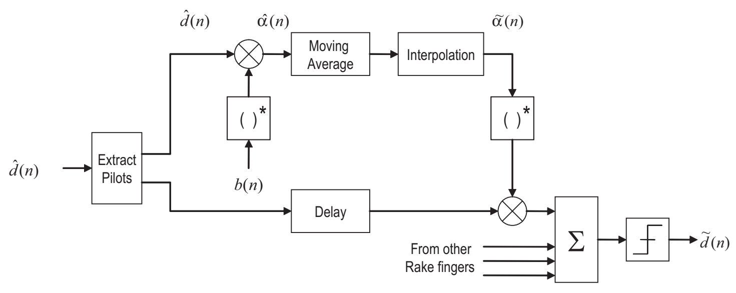


Fig. 11. *Signal flow representation of the WCDMA channel estimation algorithm.*

4.6.2 OFDM Channel Estimation

Assuming zero offset frequency and correct symbol timing estimation, the received data symbols after FFT can be expressed as

$$\widehat{D}_i(n) = \alpha_i D_i(n) + W_i(n) \tag{17}$$

where i is the subcarrier index, α_i is the complex attenuation of the ith subcarrier, $D_i(n)$ is the transmitted data symbol on the ith subcarrier, and $W_i(n)$ is additive noise. The channel estimates can be computed as in (16) for each subcarrier.

$$\hat{\alpha}_i \approx \frac{\widehat{D}_i(n)}{D_i(n)} \tag{18}$$

This results in a least square (LS) estimation of the channel coef cients [59,60]. The long training symbols in the preamble of the packet can be used as reference data in OFDM channel estimation. The packet preamble includes two identical training symbols and the estimates can be averaged between them to improve the quality of the estimation [44].

4.7 Analysis of the Receiver Algorithms

Simpli ed o w charts of the baseband processing in WCDMA and OFDM receivers are depicted in Fig. 12. The WCDMA o w chart only considers the multipath estimation, demodulation, and channel estimation of the DPCH. The channels that are employed at different stages are listed on the right side of the o w chart. In the o w chart, the multipath delay estimates are updated after each slot. However, the nal

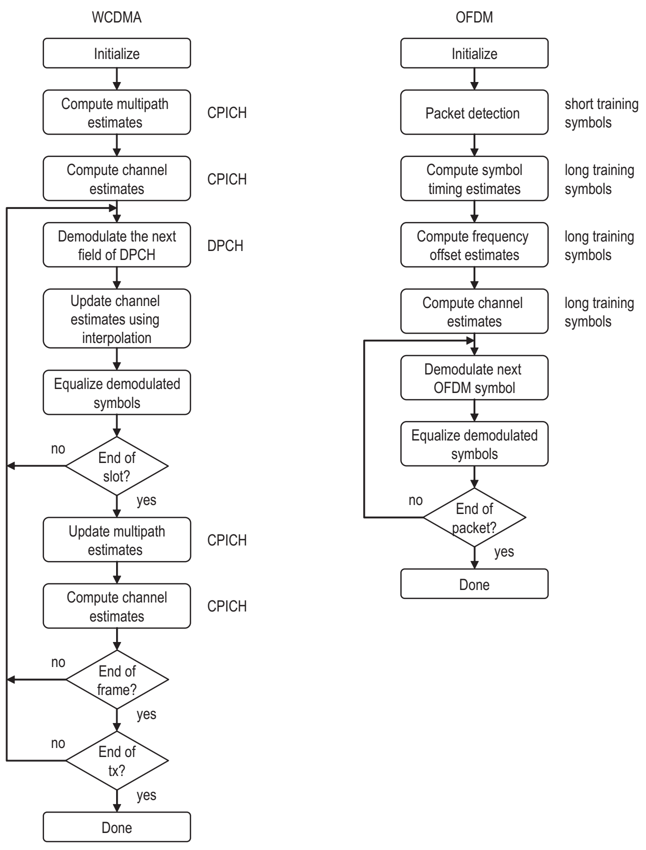

WCDMA
Initialize
Compute multipath estimates
CPICH
Compute channel estimates
CPICH
Demodulate the next field of DPCH
DPCH
Update channel estimates using interpolation
Equalize demodulated symbols
no
End of slot?
yes
Update multipath estimates
CPICH
Compute channel estimates
CPICH
no
End of frame?
yes
no
End of tx?
yes
Done
OFDM
Initialize
Packet detection
short training symbols
Compute symbol timing estimates
long training symbols
Compute frequency offset estimates
long training symbols
Compute channel estimates
long training symbols
Demodulate next OFDM symbol
Equalize demodulated symbols
no
End of packet?
yes
Done

Fig. 12. *Simplified flow charts of WCDMA (left) and OFDM (right) baseband procedures.*

update rate depends on the number of averaging iterations used to lter the sequential estimation windows. WCDMA channel estimation is performed once per slot and the channel estimates for the different elds inside a slot are computed using interpolation. The OFDM o w chart considers the reception of a single data packet. The symbol elds of the packet preamble employed at different stages are again listed on right side of the o w chart. In OFDM, the symbol timing and the channel estimates are computed once per packet.

The timing and frequency synchronization algorithms of WCDMA and OFDM clearly utilize similar correlation based computations. The two most important computation kernels of the synchronization algorithms are the delay-and-correlate algorithm and matched ltering, given by (3) and (6), respectively. Furthermore, non-coherent averaging of sequential correlation windows, as expressed in (9), is needed especially in the WCDMA multipath searcher. The peak detection block that compares the output

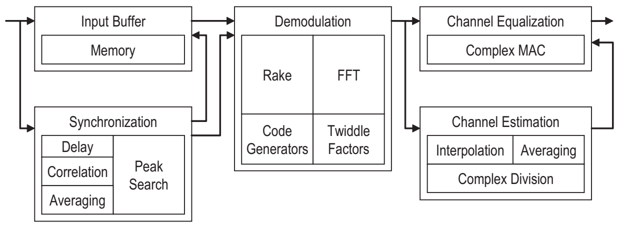


Fig. 13. *Functional block diagram of a WCDMA/OFDM baseband.*

of the correlation against a threshold value is also needed in both systems.

The biggest difference between WCDMA and OFDM baseband processing resides in the demodulation block. Because the o w of operation in Rake and FFT processing is so different, common computation kernels in the demodulation algorithms cannot be extracted similarly as with the synchronization algorithms. However, the FFT and Rake functionalities can be integrated effectively at the implementation level as will be shown in chapter 6. Moreover, the computation in the Rake ngers, expressed in (11), and the computation needed in the second step of the cell search procedure, expressed in (8), feature similar integrate-and-dump computations.

The channel estimation algorithms are very similar in WCDMA and OFDM. However, it has to be noted that their accuracy and throughput requirements are quite diverse. The main difference between the two is the feed-forward structure needed in WCDMA channels estimation.

The main conclusion that can be made based on the algorithm study is that the WCDMA and OFDM baseband receiver algorithms can be decomposed into computation kernels that can be employed as functional building blocks in the algorithm implementations. A functional baseband architecture including the most important computation kernels of WCDMA and OFDM receiver algorithms is depicted in Fig. 13. With proper parameterization, the functional building blocks can be shared between the WCDMA and OFDM modes of the receiver, which effectively minimizes the hardware overhead required by the multimode operation. Moreover, based on the analysis of this chapter it can be stated that by carefully identifying the functional similarities between the WCDMA and OFDM receiver algorithms, the receiver processing can be implemented very effectively with shared hardware resources without recon gurable hardware.

5. PROGRAMMABLE ARCHITECTURES FOR WIRELESS RECEIVERS

The importance of programmable SoC solutions in wireless products was addressed in chapter 3. In this chapter, alternatives for achieving the programmability are discussed. The basic types of programmable architectures are introduced and real life examples of each type are given. In general, programmability can be achieved either with software programmable processors or with eld programmable gate arrays (FPGA). In the chapter 4, it was pointed out that the baseband processing of a dual-mode WCDMA/OFDM receiver can be implemented without recon gurable hardware, and therefore, the text only considers software programmable architectures. Basic information of processors can be found in [61] and [62].

5.1 Non-Functional Requirements

New implementation approaches are needed in multimode baseband receivers that enable sample rate signal processing to be implemented with software. As battery life is always one of the main concerns in portable devices, low power consumption is stressed in addition to the processing power. Currently, all digital baseband processing in 2G mobile terminals can be implemented with software [63], but in 3G mobile terminals the processing power demands set by the chip rate processing, i.e., multipath searcher and Rake receive, can only be met with dedicated hardware. Typically, only the symbol rate procedures, i.e., channel estimation, deinterleaving, rate matching, and channel decoding, are implemented with software [64, 65]. In OFDM receivers, the receiver signal processing requirements are even higher because of the higher sampling rates and multicarrier operation. If mobile applications are targeted, the entire OFDM baseband receiver is likely to be implemented with dedicated hardware [66].

As stated in chapter 3, hardware and software reuse are very important in wireless terminal implementations. From a design productivity point of view, it is essential that the target application eld of the programmable architecture is as broad as possible to enable reuse of the platform in multiple applications. Furthermore, the architecture should be well suited for ef cient software development, i.e., it is required that the platform can be programmed with high-level programming languages.

5.2 Reduced Instruction Set Processors

Reduced instruction set computer (RISC) processors are architecturally simple processors targeted for general purpose computing [67]. As the name of the processor type implies, the instruction set architecture (ISA) of the processor is composed of a small amount of very simple instructions. Simple in the sense that it takes a longer sequence of instructions to execute a given function on a RISC processor than on a more elaborate processor.

Although some processors based on the RISC philosophy have in fact quite large ISAs, there are a number of design approaches common to all RISC type of processors [61,67]. First, all pipeline stages are executed in a single clock cycle. This property requires that the processing units, the instruction decoding, and the control logic of the pipeline are kept as simple as possible. Second, the instruction width is kept constant and the location of the opcode and operand registers elds within instruction are x ed. When the instruction length is x ed, the instructions in the memory do not cross word boundaries, which alleviates instruction fetching. The x ed instruction format alleviates the decoding as it enables parallel operation code decoding and register bank access. Third, memory accesses can be done only with dedicated load and store instructions. This means that the source operands of an instruction need to be read from memory to registers before the execution of the instruction. Similarly, the result of the instruction is rst written to a register, from where it can be store into memory. This method minimizes the effect of the memory access latency on the clock frequency. Fourth, RISC processors employ short pipelines, typically from 5 to 6 stages. Although increasing the number of pipeline stages would help to increase the clock frequency, the penalty of branching becomes more signi cant with deep pipelines.

The advantages of simplifying the processor architecture are evident. Because the decoding, control, and the arithmetic logic in the datapath are simple, very high clock frequencies can be achieved. Moreover, the limitations on the amount and complexity of the instructions, and the simplicity of the addressing modes make RISC processors very good targets for compilers. Because a given function needs to be built from simple operations, it is easier for the compiler to perform the optimizations. As a result of a successful optimization, the compiled code utilizes the resources of the processor very ef ciently . RISC processors can achieve very high performance by executing simple one-cycle instructions at a very high rate, and by leaving the optimization to the compiler. However, RISC processors alone are not suf cient to deliver the processing power needed in the receiver baseband processing. They are typically used in mobile terminals to perform protocol and user interface processing [63].

5.2.1 COFFEE RISC

The COFFEE RISC processor is an open source RISC core, developed at Tampere University of Technology [68]. It follows the RISC design philosophy and utilizes a Harvard architecture with a 32-bit nominal data word width. COFFEE features a 6-stage pipeline with complete hazard detection and forwarding logic. The architecture of the datapath is depicted in Fig. 14. A hardware description language implementation of the core, along with the software development tools, can be downloaded from the project web site [69]. Support for operating system is provided as the core features two operation modes (user and privileged) and necessary structures for allocating protected memory areas for the operating system. An internal interrupt controller is included in the core that enables connection of eight external interrupt sources.

The COFFEE core also features a dedicated coprocessor bus that can be used to connect four coprocessors to the core, as depicted in Fig. 15. Data between the coprocessors and the core is transferred through a common 32-bit *Data* signal and the direction of the transfer is selected with *Wr_cop* and *Rd_cop* signals. The target coprocessor is selected with the *C_indx* signal and the source or destination register is selected with the *R_indx* signal. In addition to the connection to the coprocessors register bank, each coprocessor has a dedicated interrupt signal *Cop_exc*. Because of the dedicated coprocessor bus, the read and write accesses to the coprocessor register bank are functionally identical to the accesses to the internal register bank of the core. Another important feature of the coprocessor interface is that it allows to connect

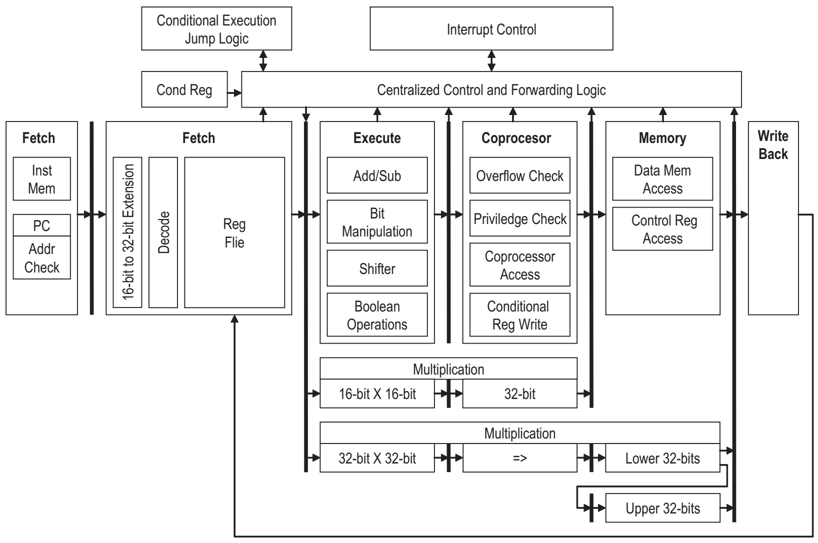


Fig. 14. *A block diagram of the COFFEE processor core.*

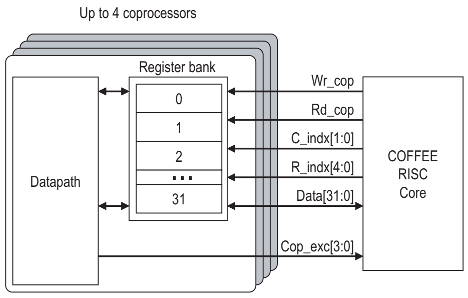


Fig. 15. *The COFFEE coprocessor bus.*

coprocessors operating in different clock domains.

5.3 Digital Signal Processors

Digital signal processors (DSP) are a step to application speci c direction from RISC processors. DSPs feature specialized instructions, addressing modes, and other hardware structures that are needed in typical signal processing algorithms. While the main performance criteria of RISC processors in general purpose computing is the average execution time, the main concern in signal processing computation is the worst case execution time.

Two DSPs designed for two different applications with different performance and cost requirements can comprise very diverse architectures. However, there exists a group of features that are common to majority of DSPs despite their target application eld [62]. The rst feature is single-cycle multiply-accumulate (MAC) operation, which is essential in many signal processing algorithms, e.g., nite impulse response (FIR) ltering and FFT. As DSP algorithms typically operate on matrices or vectors, the computation is composed of repetitive execution of short instruction sequences. Hence, a lot of execution time is spent on the looping control, i.e., updating a loop counter, testing the loop counter against an end value, and jumping back to the beginning of the loop. In order to minimize this looping overhead, many DSPs include hardware that does the mentioned operations with hardware. DSP algorithms also set very high requirement on the memory architecture of the processor. The access patterns of the applications are supported with parallel memories, special addressing modes, and dedicated address computation units. Many signal processing applications involve sensor or audio signals that are very sensitive in terms of spectral characteristics of the signal. Hence DSPs include special hardware, e.g., saturation arithmetics, for preserving the numeric delity with x ed-point data.

Because DSPs employ rather complex instructions, addressing methods, and data types, they are challenging targets for compilers. Software designers are often forced to use low-level languages to take full advantage of the specialized features. If the compiler fails to recognize the parts of the program that can bene t from the special hardware, the compiled code needs to be hand optimized in order to meet the performance and real time constraints of the application.

5.3.1 Texas Instruments TMS320C54x

A typical example of a DSP targeted for wireless communications application is the TMS320C54x from Texas Instruments [70]. The C54x is a 16-bit x ed point DSP that employs a modi ed Harvard architecture and a 6-stage pipeline. Low power consumption has been a priority for the designers of the processor, and it has reported to dissipate 17 mW with 1 V supply voltage and clock frequency of 65 MHz (0.21 mW/MHz). The processor features three different low-power modes that turn off the clock feed to the datapath, the on-chip peripherals, the interrupt lines, and the on-chip crystal oscillator in three different combinations. In addition to the typical

DSP features listed above, the C54x includes other specialized instructions, including the add-compare-select instruction needed in Viterbi decoding. The processor has been particularly successful in 2G wireless handsets to implement the digital baseband functionality. A common receiver hardware platform for 2G terminals has been a combination of the C54x DSP and a RISC core [63]. With this platform all communication algorithms are executed by the DSP, including demodulation, equalization, deciphering, channel codec, and deinterleaving, while the user interface and protocol processing are handled by the RISC core.

5.4 Multiple-Issue Digital Signal Processors

The performance of traditional DSPs can be increased simply by dividing the pipeline into smaller stages and increasing the clock frequency. However, an alternative method is to increase the parallelism of the processor by issuing multiple instructions per clock cycle. Two processor types use this approach: very long instruction word (VLIW) processors and superscalar processors. They exploit instruction level parallelism that exists in a sequence of independent instructions, i.e., instructions whose order can be altered without affecting the outcome. Both superscalar and VLIW processors exploit this property by issuing multiple instructions to parallel pipelines whenever instruction level parallelism exists. The difference between the two processor types is the method that is used for selecting the instructions for parallel execution. With VLIW processors the selection is done by the compiler (or the programmer), and therefore the parallelism is explicitly speci ed by the assembly code. Superscalar processors, in contrast, select the instruction for parallel execution on the y , with a specialized hardware that observes the data dependencies of the instructions at run time. Due to the dynamic behavior of superscalar processors, it is very hard to predict their performance. Hence, superscalar architectures are rarely used in signal processing applications that include strict real-time requirements.

VLIW processors typically employ from 4 to 8 parallel pipelines. The instructions executed in parallel are de ned by a single instruction word, and for this reason the instruction word is very long, typically from 64 to 128 bits. To fully exploit the potential of the parallelism it is required that all pipelines are utilized, i.e., useful instructions are issued to all pipelines at every clock cycle. However, the instruction issuing is on the responsibility of the compiler and occasionally the code cannot be split into parallel execution. Thereupon, idle operations have to be issued to some of the pipelines which decreases the performance of the processor. Another requirement for fully exploiting the parallel processing units is that they have to be fed with data at an adequate rate. For this reason VLIW processors employ a large amount of registers and wide memory busses. The parallel processing units, large register banks, and complex memory architectures leads to signi cantly larger chips than traditional DSP processors. Therefore, VLIWs are typically more power hungry than conventional DSPs, and thus, more suitable for applications that do not require battery operation. Another reason for VLIWs being bad candidates for mobile applications is that the size of the program memory is typically larger than with conventional DSPs. This is because the operations that compose the compound instruction are kept very simple, and hence, it can take a signi cantly larger amount of instructions to execute a given function.

5.4.1 Sandblaster

A multithreaded VLIW processor targeted for software de ned radio applications is presented in [71] and [72]. The design objective of the processor and its development tools has been to boost the software development productivity. One of the main limiting factors of the productivity of DSP software engineering is that programmers have to keep track of what is going on in the pipeline to squeeze the performance out of the processor. The target of the Sandblaster DSP has been to shift this responsibility from the programmer to the compiler.

The architecture of the processor includes a single-instruction multiple-data (SIMD) unit and a RISC based integer unit. The RISC unit is used for control processing and the SIMD unit is used for algorithm execution. Parallelism is exploited at three levels. Data level parallelism is provided by the SIMD unit that contains four parallel vector processing units, each containing a multiplier, an adder, and saturation logic. The memory architecture of the processor is designed so that each of the four vector processor units can access input data from a single-port memory within one clock cycle. Thread level parallelism is provided by including hardware support for concurrent execution of up to eight threads. Dedicated data and instruction memories are provided for each thread. Furthermore, interrupt logic is provided that enables a speci c thread to interrupt any other thread with a low latency. Instruction level parallelism is exploited by enabling instructions for the RISC and SIMD units to be issued in parallel. A baseband receiver platform for software de ned radio handsets using the Sandblaster DSP is presented in [73]. The platform can be used for implementing WCDMA, 802.11 WLAN, and GPS transceiver functionalities. A combination of four Sandblaster cores and an ARM micro controller are used to achieve a total of 9 billion MAC operation per second. The reported power consumption is less than 500 mW.

The compiler of the processor uses a technique called semantic analysis. The programmer writes the application using standard C-code and the compiler analyzes the code and identi es DSP structures that can utilize the application speci c structures of the processor cores. The compiler also extracts data level parallelism from the code and constructs the vector instructions for the SIMD-unit, and attaches these to the RISC instruction to produce compound instructions.

5.5 Application Speci c Instruction Set Processors

In many applications the above mentioned processor types are not suf cient, either because they do not meet the performance requirements of the application or because they are too costly in terms of power or area. Application-speci c instruction set processors (ASIP) are placed in the middle ground between ASICs and DSPs in terms of e xibility and energy ef cienc y. The performance gain compared to conventional DSPs is achieved effectively by narrowing the target application eld of the processor [74]. In practice this means that the ISA is tuned extensively for the application so that the number of instructions needed to execute a function is minimized. The memory architecture is designed so that it best supports the data access patterns of the application, minimizing the number of memory accesses and the penalty caused by a single access. The number of registers and the connections between registers and processing units is reduced to the absolute minimum in order to diminish the critical path delay and the physical size of the datapath. At the same time, the instruction format is simpli ed because source and destination registers of the instructions can be x ed and instruction elds for selecting them are no longer needed. Finally, the peripherals of the processor are designed to serve the special needs of the application. With these optimizations the feasibility of the processor is restricted to a very small number of applications but in return the performance of the processor can be boosted without increasing the clock frequency or the number of processing units signi cantly . As a result, the size and the power consumption of the processor are kept in control.

To nd the optimal architecture and ISA for the processor, the target application has to be analyzed rigorously. This is a signi cant design effort and one of the main drawbacks of the application-speci c approach. However, many tools exist that automate the process of architecture exploration and identifying the right instructions for the applications [75–77]. The biggest disadvantage related to ASIPs is that they cannot be programmed with high level programming languages. Because the ISAs of ASIPs are typically even more specialized than those of DSPs and VLIWs, designing a compiler that employs the architecture effectively is even more dif cult. The potential of a highly tuned datapath and memory architectures can only be utilized if a customized compiler is available that is able to make use of them. Typically, the most critical parts of the software need to be programmed using low-level programming and the productivity of the software development is greatly reduced. Although this problem is alleviated by ASIP design tools that provide straightforward programming models for the application speci c structures, the bigger problem is that the software written for ASIPs is highly machine dependent. This rules out the possibility of effectively reusing the existing software in projects utilizing even slightly different processor architectures.

5.5.1 Tensilica Xtensa

Xtensa is a commercial ASIP solution that is built upon a 32-bit RISC architecture [78, 79]. The processor is available as a soft IP block that can be licensed for use in SoC architectures. The Xtensa architecture can be used as a off-the-shelf RISC processor or it can be armed with optional processing units, I/O interfaces, and memory architectures, or it can customized by adding instructions to the ISA. Furthermore, the latest version of the core supports multiple-issue custom instructions. The custom instructions are described with a hardware description language, and the customized core and the software tools supporting the added instructions are produced by a tool. To alleviate the selection of the most pro table custom instructions a compiler is provided that analyzes the application software and generates the hardware descriptions automatically for potential custom instructions.

5.5.2 ASIPs for Wireless Receivers

An ASIP designed for WCDMA and OFDM receivers is presented in [80]. The processor features special instructions for butter y , add-compare-select, and sum of squared difference operations. Sub-word parallelism is utilized in the processor datapath which means that the processing units can be routed in various ways in order to implement complex or real valued instructions with single or double precision. The data word of the processor is split in two, for real and imaginary parts, and the processing units support complex arithmetics. For example, a complex MAC operation can be executed in a single clock cycle. At the maximum clock frequency 65 MHz the processor achieves a peak performance of 1.1 billion MACs per second. The maximum clock data rate is quite modest compared to many modern DSPs, which is due to the routing overhead needed for the sub-word parallelism. With 0.35 μm technology the processor employs 25.2 mm^2 of silicon area, and the reported power consumption is 14 mW at 1.5 V supply voltage and 22 MHz clock frequency.

An ASIP designed for OFDM receiver implementations is presented in [81]. The datapath of the processor comprises two multipliers, three adders, and a switching logic that routes the multipliers and adders in two different con gurations to execute a butter y operation in two clock cycles. The processor includes an address generation unit that is dedicated for computing the bit-reversed input data addresses of the FFT. In addition, special bit manipulation hardware is employed to support scrambling, convolutional encoding, and puncturing functions. An FFT operation can be implemented on the processor with only three assembly instructions and a 64-point FFT is executed in 390 clock cycles. The implementation of the processor on 0.18 μm technology consumes 80 000 gates and achieves a maximum clock frequency of 240 MHz. Demodulation of a single OFDM symbol in a 802.11a system requires 1.4 μs of computation time, which is 2.6 μs less than the symbol period.

5.6 Coprocessor Accelerators

An alternative approach for increasing the processor performance is to employ coprocessor accelerators. In the ASIP approach the datapath of the processor as a whole is tuned to the application, whereas in the coprocessor approach the application speci c processing units are attached in parallel to the processor datapath [82]. This results in a considerable performance boost especially in applications where a majority of the execution time is spent inside computation kernels that can be separated as clear functional entities.

A key factor in the coprocessor approach is to minimize the communication overhead between the core and the coprocessors. Ideally, the core simply passes parameters to the coprocessor, triggers the coprocessor, and then reads the results when the execution completes. The nal speedup enabled by the coprocessor is determined by the clock cycles saved in the computation and the cycles lost in communicating with the coprocessor. Thus, the communication mechanism between the core and the coprocessor is of paramount importance. Another very important factor in the coprocessor approach is the programming model of the architecture. One option is to use the coprocessor much like a function call, where the caller waits until the function completes. Another option is to use the execution time of the coprocessor to perform other useful computations. This naturally complicates the programming because concurrency introduces some well known pitfalls, including mutual exclusion and synchronization. The design effort of the coprocessors can be signi cant, and similarly as with ASIP design, automated design tools for design exploration are needed. Tools that analyze the application code and produce the hardware description for coprocessors automatically have been published [83].

5.6.1 Coprocessor Architectures for Wireless Receivers

A platform targeted for WCDMA and IEEE 802.11b baseband processing is presented in [84]. The architecture comprises four DSPs, a correlator coprocessor, a channel decoding coprocessor, and an ARM core for protocol processing. Each of the four DSPs employ a SIMD architecture composed of four processing elements. The channel decoding coprocessor is an ASIP type of engine that includes a general purpose control processor and an application-speci c processor specialized in Trellis recursion. The control processor is used for interfacing the coprocessor to the system bus and con guring the Trellis data path. The correlator coprocessor is a multithreaded FIR architecture that supports complex data and coef cients, and dedicated memory and I/O connections for all four concurrent ltering tasks. An application programming interface (API), a real-time operating systems, a compiler, and a debugger are also provided for the platform. The API is basically a library of functions that provides a programmer's view for the computational resources, peripherals, and operating system functions. The API includes instructions for setting up complete receiver implementations with a single function call. The compiler of the SIMD processors extracts the data-level parallelism from a sequential program but it is required that a data-parallel extension of C-language is used. The mapping of the receiver procedures onto the four DSPs is done manually. A system-level design methodology for the platform and a prototype multi-standard receiver design are presented in [85]. The design methodology is divided into ve systematic phases extending from functional system-level modeling to circuit implementation.

A programmable baseband receiver architecture for GSM, EDGE, and WCDMA base stations is presented in [86]. The architecture is split between chip rate processing running on a coprocessor and symbol rate processing running on a VLIW DSP. The interface between the host DSP and the coprocessor is implemented with a DMA and an additional host interface. The coprocessor can execute Rake nger , multipath searcher, and preamble detection procedures. The DSP triggers the coprocessor processing through task messages that include a number of input parameters and the starting time of the task. The coprocessor executes the task starting at the speci ed time instant, and dumps data to the host interface. Another implementation of the same platform presented in [87] features also a RISC core that is used for generating the micro code for the coprocessor. This enables changes to the coprocessor functions after the chip has been fabricated but it also introduces an area overhead that cannot be considered for mobile terminal implementations. The platform is operated with a 1.2 V supply voltage and 246 MHz clock frequency. The implementation of the platform comprises 75 million transistors, and the reported power dissipation is 2 W for the core and 200 mW for the I/O.

5.7 Comparison of Architectural Alternatives

Each category of programmable architectures presented in this chapter have their own advantages. The ultimate goal for a programmable baseband receiver architecture is to achieve small size, small power consumption, high e xibility, easy programmability, and support for ef cient reuse of hardware and software. If easy programmability is the primary concern, the optimal architecture would consist of a single general purpose processor operating at a suf cient clock frequency. The reason why general purpose processors are preferred is because they are good target for compilers, which enables effective usage of high level programming languages. Compromising the easy programmability, yet emphasizing the e xibility, would result in an architecture consisting of multiple general purpose processors or DSPs. Programming this type of architecture is more dif cult, because the peculiarities of concurrent programming need to be considered. If small power consumption is the biggest concern, the ASIP approach would be a good option but this would compromise both easy programmability and e xibility. The employment of application speci c processing units in the processors datapath restricts the target application eld and leads to usage of low-level programming or programming language extensions, often both. As a result, software reuse with ASIPs is very dif cult because of the machine dependence of the code. The coprocessor approach is a way to achieve improved programmability and e xibility with similar power ef cienc y as the ASIP approach. Compared to the DSP approach, the e xibility is compromised because the programmer has a limited control over the execution of the coprocessor functions. On the other hand, typical wireless receiver algorithms incorporate clear computational entities that require very little, if any controlling from the programmer. These type of computational kernels are very well suited for a coprocessor implementation. Because the coprocessor functions are running on dedicated hardware, the overhead caused by looping, branching, and other controlling tasks is also minimized.

Based on the above summary, the coprocessor approach comes off as the most attractive alternative for a programmable baseband receiver architecture. However, the coprocessor based baseband receiver architectures listed in section 5.6.1 featured some less attractive features. In the platform presented in [86] the most evident disadvantage is the 2 W power consumption. It has to noted that the platform was designed for base station applications, and presumably, power consumption has not been a primary concern in the design. Another disadvantage of the platform is that the communication between the DSP and the coprocessor was operated through a DMA and an additional host interface. The amount of traf c between the DSP and the coprocessors can be minimized by increasing the granularity of the coprocessor functions, but the communication overhead can still result in a failure to meet the real time constraints of the baseband processing. The coprocessor bus featured in the COFFEE core provides an effective solution for attaching coprocessors with minimized communication overhead. In the platform presented in [84] the programming of the platform requires use of a specialized C-language in order to bene t from the processing resources of the SIMD DSPs. As mentioned previously, low-level languages and language extensions limit the reuse of software. The optimum solution for achieving good programmability would be similar to the semantic analysis approach presented in [73]. However, the requirements of the compiler can be relieved by using a host processor that is compiler friendly, preferably a RISC core. In the coprocessor approach, the actual coprocessor implementations determine the performance of the architecture to a large extent, but when easy programmability and software reuse are more important, the connection between the host and the coprocessors, the compiler friendliness, and the programmer's view of the coprocessor are crucial.

6. THE ESPRESSO PLATFORM

The baseband receiver algorithm study in chapter 4 indicated that WCDMA and OFDM baseband receiver processing feature many similar computation kernels. The baseband processing of a dual-mode WCDMA/OFDM baseband receiver can be implemented effectively by exploiting the similarities of these computation kernels and by sharing the computation resources between the two modes of the receiver. The analysis of different implementation alternatives for programmable baseband receivers in chapter 5 showed that an architecture composed of a RISC core and an attached coprocessor is a good approach for achieving high performance, while maintaining programmability and e xibility of the architecture. The coprocessor approach was deployed in the Espresso platform presented in this chapter [P4, P5, P6, P7].

6.1 Espresso Overview

The receiver algorithms presented in chapter 4 are mapped onto a platform composed of a RISC core and three coprocessors. The three coprocessors are used for synchronization, demodulation, and I/O tasks, and the RISC core is used for channel estimation and equalization. The core itself does not process the incoming sample stream, but only the demodulated symbols. The RISC core used in the platform is the COFFEE processor introduced in chapter 5. The Espresso platform architecture is depicted in Fig. 16. In addition to the core and the coprocessors, the platform comprises direct memory access (DMA), two memory busses, and on-chip memory for instructions, data, and sample input buffering. However, implementations of these components are not studied in this thesis.

The general architecture common to all coprocessors is depicted Fig. 17. The architecture is composed of an instruction FIFO buffer, an instruction decoder, parameter registers, coef cient registers, code generators, datapath, control, and an output

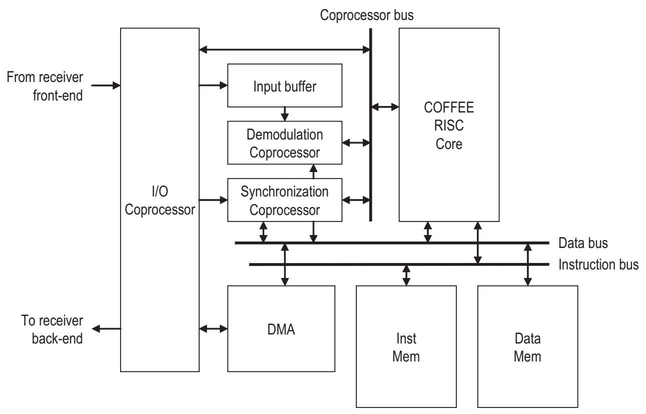


Fig. 16. *The Espresso platform.*

FIFO buffer. The coprocessor is connected to the coprocessor bus of the COFFEE core. The programmer can access the instruction register, the parameter registers, the coef cient registers, and the output FIFO through the coprocessor bus. The ISA of the COFFEE core includes special instructions for reading from and writing to the coprocessor register bank through the coprocessor bus. The synchronization coprocessor is also connected to the I/O coprocessor that provides the sample input, and the demodulation coprocessor is connected to the input buffer that is used to store the samples within the longest expected multipath delay spread. The execution of the coprocessor functions is divided into three stages: instruction fetch from the FIFO, decode, and execution. However, these stages do not overlap in a pipelined manner.

The programming interface of the coprocessors is implemented with a library of coprocessor function calls. The programmer writes code for the COFFEE core and uses these function calls to initiate the computation kernels on the coprocessors and to access the register bank inside the coprocessors. The instructions initiated by the programmer are executed sequentially in the order they are stored into the instruction FIFO. Thus, the programmer can call new coprocessor functions before the previous instruction has been completed.

6.2 Synchronization Coprocessor Architecture

The synchronization coprocessor is designed for the correlation based synchronization algorithms needed in OFDM and WCDMA receivers [P6]. The datapath of

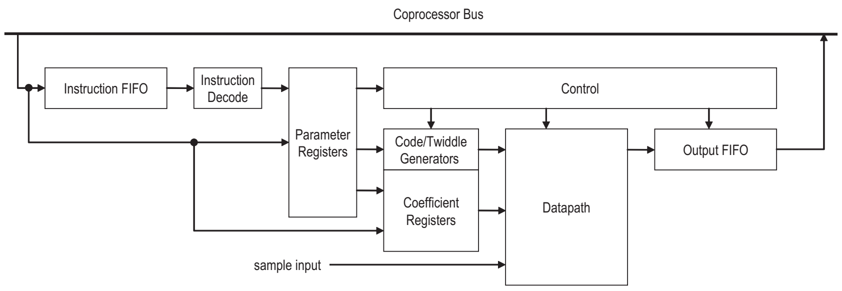


Fig. 17. *General architecture of the coprocessors.*

the coprocessor comprises a FIR lter architecture that can execute either matched ltering or delay-and-correlate algorithm. In addition to the FIR architecture, the synchronization coprocessor includes hardware for averaging sequential correlation windows and detecting peaks from the output of the FIR block.

6.2.1 The Datapath

The implementation of the FIR block is fundamentally a tapped delay line structure with complex data and coef cients. The dual mode operation of the FIR is implemented with a routing network that can select the coef cients for each stage from two different sources. In the matched lter mode the coef cients are routed from the coef cient registers and in the delay-and-correlate mode they are routed from the FIR delay line as illustrated in Fig. 18(b). In addition to the mode of the correlation, the length of the correlation can also be changed. Each stage of the FIR is connected to the output of the previous stage and to the sample input, which allows to select the tap acting as the rst lter stage. The maximum length of the lter in the matched lter mode is $L = 256$, and the maximum delay in the delay-and-correlate mode is $D = 128$. The matched lter mode of the FIR block, with convolution length $L = 4$, is illustrated in Fig. 18(a) and the delay-and-correlate mode, with delay $D = 4$ and correlation length $L = 4$, is illustrated in Fig. 18(b). The delay-and-correlate mode could be implemented alternatively by shifting the multiplication to the beginning of the delay line and by computing a moving sum of the multiplication results. However, this alternative implementation would require double length registers in the delay line which would result in a signi cantly larger silicon area and it would also complicate the dual-mode operation of the FIR block.

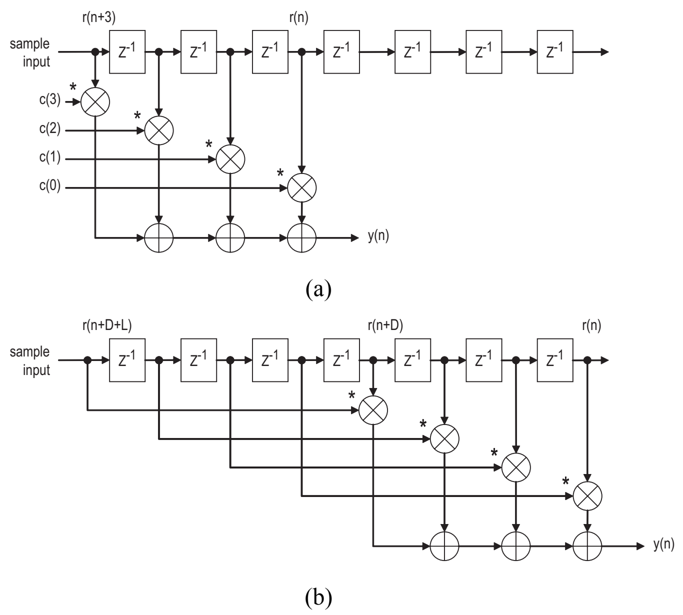


Fig. 18. *The datapath of the synchronization coprocessor: (a) matched filter mode ($L = 4$), (b) delay-and-correlate mode ($L = 4$, $D = 4$).*

The routing of the coef cients and the implementation of the variable length correlation is done with multiplexers. For a FIR structure of 256 taps and 8-bit complex coef cients this would result in a signi cant increase in silicon area and critical path delay. However, the complexity of the FIR block is greatly reduced by limiting the width of the coef cients to 1 bits. In WCDMA cell search and multipath searcher procedures the pilot sequences used in the correlation are always composed of values ± 1 only. Likewise in OFDM, the synchronization algorithms can be executed by using only sign bits of the coef cients [88]. As a result, the routing network of the coef cients is simpli ed greatly and also the complex MAC operation of the taps are simpli ed to a two stage add/subtract computation [P3].

6.3 Demodulation Coprocessor Architecture

The demodulation coprocessor is fundamentally a set of processing units (PU) that can be routed to implement either Rake nger correlations or FFT computations [P7]. The processing units perform the butter y operations in the FFT mode, and the multiply-accumulate operation in the Rake mode.

6.3.1 The Datapath

The datapath of the demodulation coprocessor in the Rake and FFT modes is illustrated in Fig. 19(a) and Fig. 19(b), respectively. The implementation of the coprocessor Rake mode is based on the FlexRake architecture [P1, P2, P3]. As mentioned in chapter 4.2, a traditional Rake receiver is composed of parallel ngers that are each used to despread one multipath component of the received WCDMA signal. The essence of the FlexRake architecture is that the multipath components are processed sequentially with a single correlator. The incoming sample stream is stored into a circular buffer that stores the samples within the longest expected multipath window. The samples are read from addresses determined by the estimated multipath delays. This approach facilitates the nger allocation task under rapidly changing channel conditions. Since its introduction, the FlexRake architecture has been used in numerous Rake receiver implementations [89–92]. The difference between a traditional Rake receiver and the FlexRake is illustrated in Fig 20. The main drawback of the FlexRake approach is that the memory used for the input buffering needs to be accessed at oversampling rate. The buffering in the traditional Rake architecture is used to equalize the time difference between completed symbol integrations, and therefore, the memory used for this buffering needs to be accessed only at symbol rate.

The implementation of the coprocessor FFT mode is based on the single-path delay feedback (SDF) architecture [93]. In this architecture, the FFT execution is divided into pipeline stages so that each stage of the FFT has a dedicated processing unit, as depicted in Fig. 19(b). The input samples are read sequentially from the input buffer and the correct input pairs for the butter ies are acquired by the delay lines at each stage. The memory accesses of the SDF architecture are functionally very similar to the the FlexRake. In the FlexRake, the read addresses are spread over the longest expected multipath window using the offset addressing, and in the SDF architecture the read addresses are spread over the OFDM symbol period using bit-reversed addressing.

In addition to the Rake and FFT functionalities, the coprocessor can be used to execute up to 24 parallel complex valued correlations. Six parallel processing units are available, each of which can be shared with four concurrent correlations. The operation of each processing unit is effectively divided into four time slots. When operated

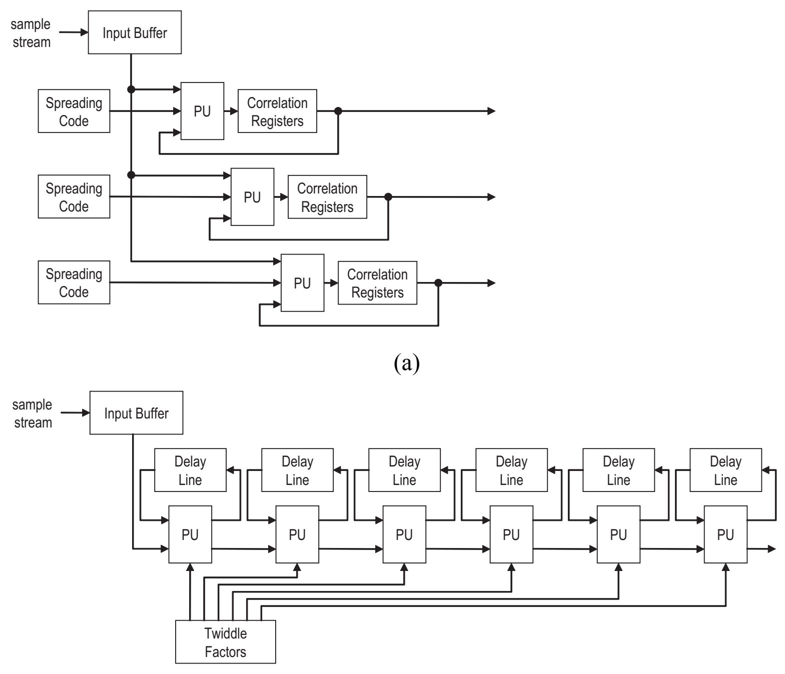


Fig. 19. *The datapath of the demodulation coprocessor: (a) FlexRake mode, (b) single-path delay feedback FFT mode.*

in this mode, the parallel processing units are always fed with the same input but the input for the time multiplexed correlation slots can be read from different input buffer addresses. Up to 32 complex valued coef cients vectors, with 1-bit real and imaginary components, can be loaded to the coef cient registers. This feature is needed, e.g., in the second phase of the cell search procedure in WCDMA, as explained in section 4.3.2.

6.3.2 The Processing Unit

The processing units perform the butter y operations in the FFT mode, and the multiply-accumulate operation in the Rake mode. A butter y operation in decimation-in-frequency (DIF) FFT is composed of one complex addition, one complex subtraction and one complex multiplication [55]. This requires a total of four adders, two subtracters, and four multipliers, as illustrated in Fig. 21(a). In the Rake mode, the operation of the PUs is much simpler because the code input is binary valued, as

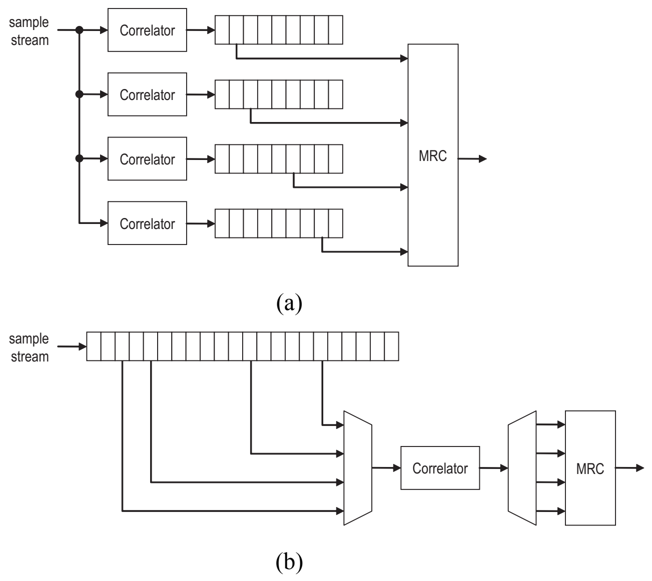
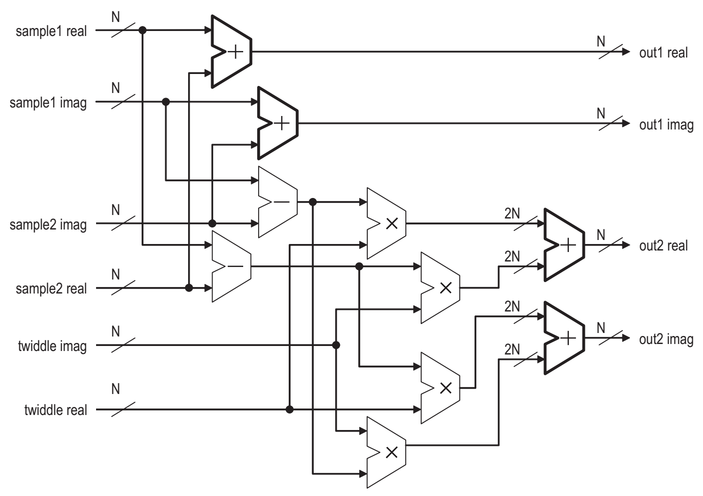


Fig. 20. *Comparison of Rake architectures: (a) a traditional Rake receiver, (b) the FlexRake.*

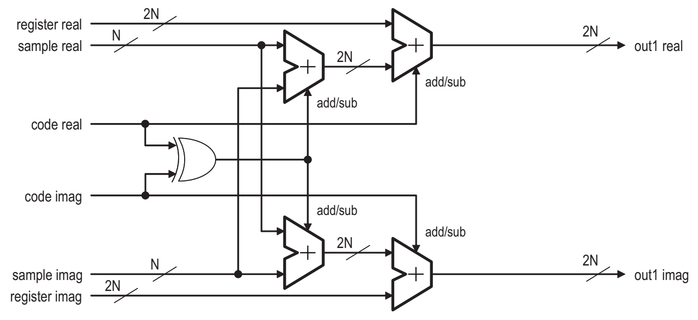
shown in chapter 4.5.1. The complex multiplication in the correlations is simpli- ed to a stage add/subtract structure, as depicted in Fig. 21(b). The computation resources are shared between the two modes of the PU, and the selection between the two modes is made automatically by the hardware. The particular elements that are shared are depicted with bold line in Fig. 21.

6.4 I/O Coprocessor Architecture

The architecture of the I/O coprocessor is very simple. It controls the sample feed to the input buffer and the synchronization coprocessor. It also provides the circular write addresses to the memory block used for the input buffer. The programmer can switch on/off the sample feed to the input buffer and to the synchronization coprocessor. Furthermore, the write address for incoming samples in the buffer can be adjusted which effectively determines the beginning of the time window in the synchronization algorithms.



(a)



(b)

Fig. 21. *The processing unit in (a) FFT and (b) Rake modes.*

6.5 Espresso Programming Interface

As addressed in chapter 5, the main drawback of application-speci c programmable architectures is that they are not well suited for effective software development because they often require low-level programming. In the Espresso platform this problem is overcome by employing a RISC core as a host processor and by implementing the most computationally demanding kernels in the coprocessors. As a result, the compiler only affects the less critical computation executed in the RISC core. All implementation details of the coprocessors are hidden from the programmer behind a programming interface.

Table 2. *The coprocessor functions.*

Common Functions
*cop_wr(**int** cop_idx, **int** reg_idx, **int** cop_data)*
*cop_rd(**int** cop_idx, **int** reg_idx)*
*flush_inst_fifo(**int** cop_idx)*
Synchronization Coprocessor
*init_sync_scode(**int** n, **bool** ack, **bool** intpt)*
*set_thr(**int** new_thr)*
*init_sync_corr_coef(**int** coef_handle, **int** length, **bool** ack, **bool** intpt)*
*set_track_param(**int** winl, **int** gapl, **int** nave)*
*corr_x_thr(**int** coef_handle, **bool** dly_and_corr, **int** dly, **int** length, **bool** use_gold, **bool** track, **bool** ack, **bool** intpt)*
Demodulation Coprocessor
*init_ovsf(**int** sf, **int** n, **int** coef_handle, **int** ack, **bool** intpt)*
*init_demod_scode(**int** n, **int** ack, **bool** intpt)*
*init_demod_corr_coef(**int** coef_handle, **int** length, **bool** ack, **bool** intpt)*
*init_multipaths(**int** dly1, **int** dly2, **int** dly3)*
*despread(**int** coef_handle, **int** start_addr, **int** nsym, **int** ncorr, **bool** use_gold, **bool** ack, **bool** intpt)*
*fft(**int** start_addr, **int** length, **int** nsym, **bool** ack, **bool** intpt)*
I/O Coprocessor
*rec_onoff(**bool** onoff)*
*set_win(**int** win_addr)*

The programming interface of the coprocessors is implemented with a library of coprocessor function calls [P5, P6, P7]. The list of the functions is given in Table 2. The programmer writes code for the COFFEE core and uses these function calls to initiate the computation kernels on the coprocessors. The available coprocessor functions and their parameters provide the e xibility required for employing the platform in a multiple applications inside a broader application eld. In the case of the Espresso platform the application eld is de ned by the WCDMA and OFDM radio technologies. The synchronization between the concurrent processes running on the coprocessors and the core is maintained with coprocessor interrupts. Functionally the coprocessors can be regarded as a hardware implemented version of a software library containing receiver signal processing routines. As mentioned in chapter 5, in a conventional DSP software development, the most computationally demanding kernels are hand optimized with low-level languages. In the presented coprocessor approach these critical kernels are implemented as the coprocessor functions.

The bodies of the coprocessor functions build the coprocessor instruction from an operation code and input parameters, and write the built coprocessor instructions to the instruction FIFO of the desired coprocessor. The only processor speci c functions are the ones implementing the data transfer between the core and the coprocessors, i.e., *cop_wr* and *cop_rd*. The *cop_wr* function writes the data from the input parameter *cop_data* to the speci ed coprocessor register. The *cop_rd* function returns the data from the speci ed coprocessor, from the register address given as a parameter. These functions are the only ones requiring low-level programming, and they only need to be rewritten if the communication mechanism between the host processor and the coprocessors is changed. The *cop_wr* and *cop_rd* functions employ special instructions for accessing the coprocessor bus. The other functions are written with standard C-language. As a result, the programming interface and the software developed on top of it are reusable.

The following sections demonstrate the use of the programming interface through examples.

6.5.1 WCDMA Multipath Estimation

A sequence diagram of WCDMA multipath searcher procedure is depicted in Fig. 22. When the programmer wants to implement WCDMA multipath searcher procedure with the synchronization coprocessor, the rst thing to do is to initialize the scrambling code generator of the coprocessor. This is done by calling the *init_sync_scode* function with the scrambling code number as an input parameter. The parameters *ack* and *intpt* determine whether the coprocessor issues an interrupt after it has decoded the instruction and after the initialization is completed.

Next, the coef cients used in the matched ltering need to be loaded to the coprocessor coef cient registers. The programmer initiates the coef cient load by calling the *init_sync_corr_coef* function with a coef cient handle and the length of the vector to be loaded as input parameters. By setting the *ack* ag, the programmer can monitor when the coprocessor has decoded the *init_sync_corr_coef* function and is ready to receive the coef cients. The coef cients are then written into a speci c coprocessor register sequentially starting from the element with smallest index. The real and imaginary parts of the coef cients are written in a single word, imaginary component in the most signi cant half and real component in the least signi cant half of the word. The coprocessor stores the coef cients to registers starting from a location speci ed by the coef cient handle.

The next thing to do is to set the peak detection threshold by calling the *set_thr* function with the new threshold as an input parameter. If the sample feed to the synchronization coprocessor is not tuned on, it has to be done by calling the *sync_onoff* function of the I/O coprocessor.

The matched ltering can now be triggered by calling the *corr_x_thr* function with the coef cient handle, mode of the correlation (matched ltering/delay-and-correlate), and the length of the correlation as parameters. Furthermore, the *use_gold* ag needs also to be set to turn on the feed of the scrambling code to the FIR taps. Once triggered, the correlator runs until the rst peak occurs that crosses the set threshold. The coprocessor issues an interrupt and writes the correlation index and the value of the detected peak to the output FIFO.

The coprocessor can also be programmed to compute averaging correlations starting from the rst detected correlation peak. This is done by specifying tracking parameters with the *set_track_param* function and setting the *track* ag when triggering the correlation. The input parameters for the *set_track_param* function include the length of the search window, the length of the gap between correlation iterations, and number of averaging iterations. After detecting the rst peak, the coprocessors start the tracking mode where it automatically averages the speci ed number of sequential correlation windows. The gap between correlation windows is useful when the correlation is performed against a speci c eld of a slot. During the gap, the delay line of the FIR block is shifted without any computations and the scrambling code indices of each tap are automatically incremented.

When the tracking is completed, the coprocessor detects all peaks from the averaged correlation window, issues an interrupt, and writes the indices and values of the correlation peaks to the output FIFO. The programmer can then read the results from the output FIFO with the *cop_rd* function and select the strongest peaks by comparing the peak values. Once the multipath delays are estimated, they have to be initialized to the demodulation coprocessor with the *init_multipaths* function. Three multipath delays are given as parameters, and their values are given as offsets relative to the rst multipath.

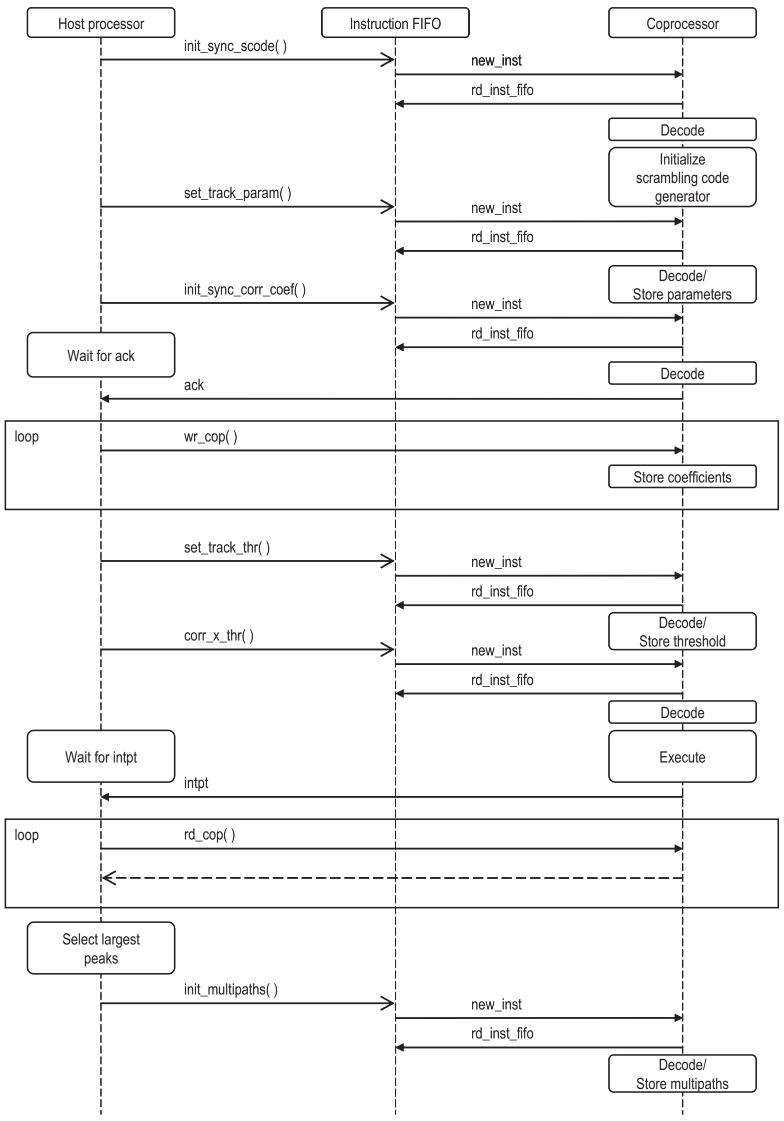


Fig. 22. *WCDMA multipath searcher sequence diagram.*

6.5.2 WCDMA Demodulation

A sequence diagram of the WCDMA demodulation procedure is illustrated in Fig. 23. After the multipath searching, the demodulation can be performed. First, the spreading code and scrambling code generators of the demodulation coprocessor have to be initialized. Up to three spreading codes can be initialized for demodulation of three

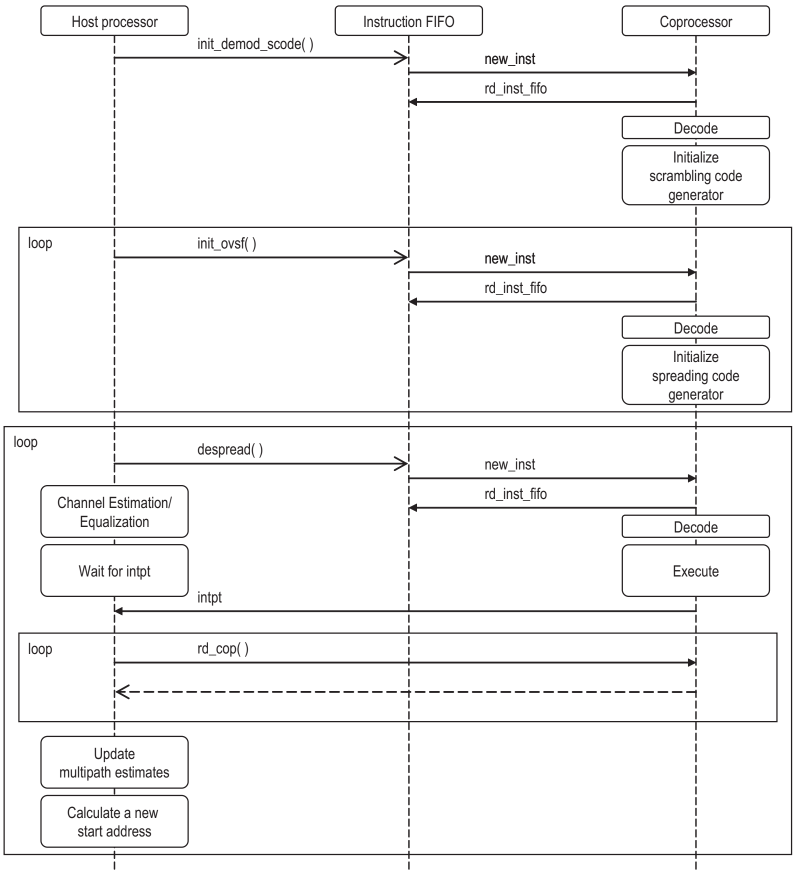


Fig. 23. *WCDMA demodulation sequence diagram.*

parallel code channels. Each of these are initialized by calling the *init_ovsf* function with the spreading factor, spreading code number, and a coef cient handle as input parameters. The coef cient handles for parallel code channel have to be sequential numbers. The scrambling code generator is initialized similarly as with the synchronization coprocessor.

Once the code generators have been initialized the demodulation can be triggered with the *despread* function. The coef cient handle of the desired spreading code is given as a parameter. The second input parameter determines the rst read address to the input buffer. This is given by the index of the rst detected peak in the multipath searcher procedure. Other parameters include the number of symbols to be despread and the number of parallel code channels. If parallel code channels are being employed, the spreading codes with sequential coef cient handles, starting from the handle given as an input parameter, are automatically employed in the despreading. The demodulation starts from the speci ed input buffer address and the multipath components are accessed from the input buffer using the multipath delays as offset addresses. While the demodulation is running on the coprocessor, the channel estimation and equalization of previous demodulated symbols can be executed on the the host processor.

If the *intpt* ag was set, the coprocessor issues an interrupt when the rst symbol is demodulated and written to the output FIFO. The programmer can then access the demodulated symbols from the output FIFO, update the multipath estimates, compute the start address of the next demodulation cycle, and call the *despread* function again.

6.5.3 OFDM Synchronization

The OFDM packet detection and symbol timing estimation procedures are illustrated in Fig. 24. In OFDM synchronization, the programmer rst stores the long training sequence to the coef cient register of the synchronization coprocessor using the *init_sync_corr_coef* function. A coef cient handle and the length of the coef cient vector to be loaded are given as input parameters. The procedure was described in section 6.5.1. The long training symbols are not needed until the symbol timing estimation because the packet detection is performed with delay-and-correlate algorithm. However, it is bene cial to initialize them beforehand. The detection threshold for the packet detection is initialized next with the *set_thr* function.

The delay-and-correlate computation is triggered by calling the *corr_x_thr* function with the type of the correlation, the length of the delay, and the length of the correlation as parameters. If the short training symbols are employed in the delay-and-correlate algorithm, the delay and the length of the correlation are set to 16. The coef cient handle does not have to be speci ed because the *dly_and_corr* ag is set. No tracking is used in the packet detection and the scrambling code feed is turned off. The programmer can also set the *ack* ag of the *corr_x_thr* function and pend for an interrupt from the synchronization coprocessor to monitor when the correlation starts.

While the coprocessor is executing the correlation, the programmer can set the threshold and the tracking parameters used in the symbol timing estimation. The length

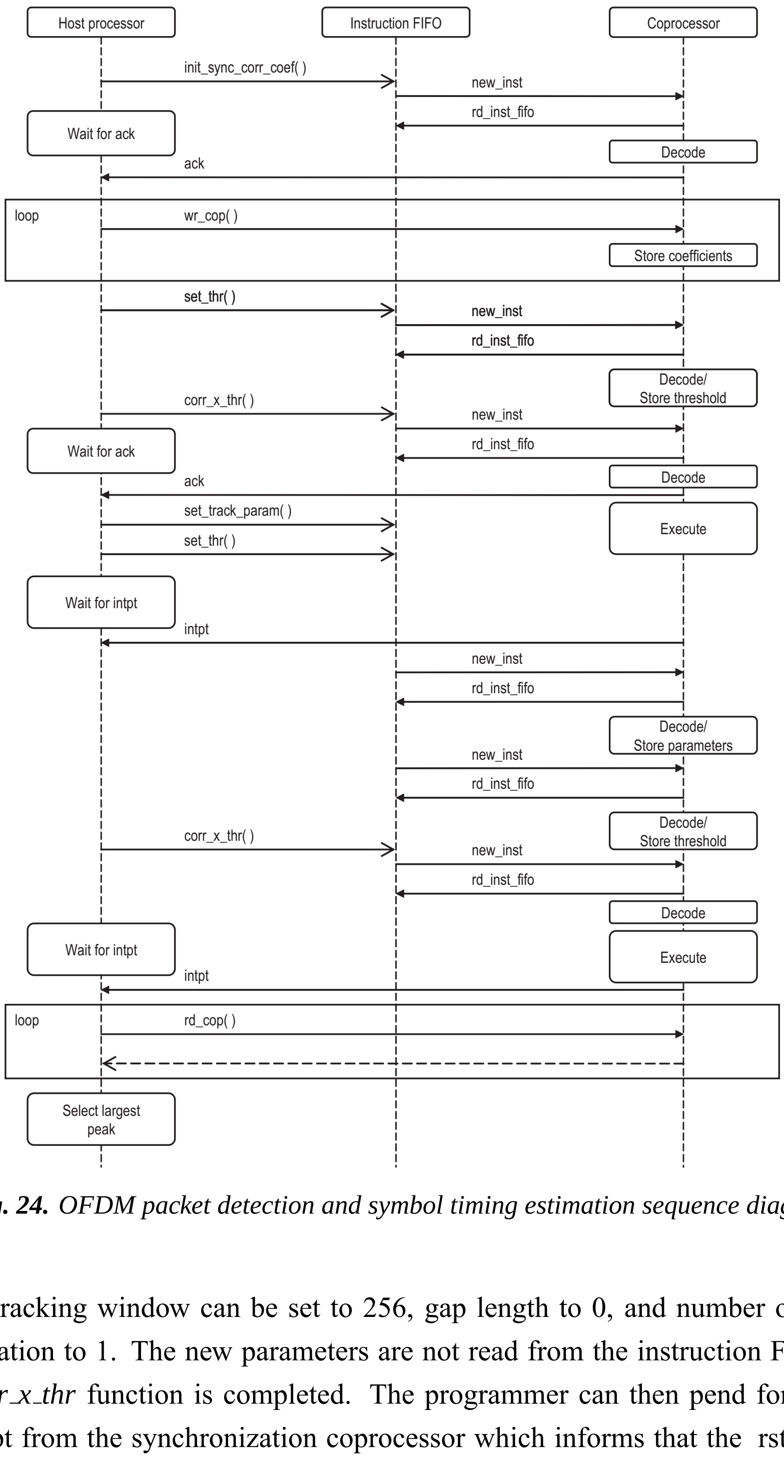


Fig. 24. *OFDM packet detection and symbol timing estimation sequence diagram.*

of the tracking window can be set to 256, gap length to 0, and number of averaging iteration to 1. The new parameters are not read from the instruction FIFO until the *corr_x_thr* function is completed. The programmer can then pend for the next interrupt from the synchronization coprocessor which informs that the rst correlation is completed and the packet has been detected. The programmer then calls the *corr_x_thr* function again to start the symbol timing estimation with the tracking parameters set earlier. Now a coef cient handle of the long training sequence and the length of the correlation are given as parameters. The *track* ag is also set to activate the tracking mode, and the *intpt* ag is set again to inform the coprocessor that an interrupt needs to be issued upon the completion of the function.

The coprocessor rst correlates until the rst peak occurs, then starts the tracking mode. This time no actual averaging is performed because the number of averaging iterations was set to 1. When the speci ed number of averaging iterations is completed, the coprocessor detects all peaks from the averaged correlation window, issues an interrupt, and writes the indices and values of the correlation peaks to the output FIFO. The programmer can then select the strongest peaks by comparing the peak values.

6.5.4 OFDM Demodulation

When the symbol timing estimate is established as described above, the OFDM demodulation can be initiated with the *fft* function. The rst read address to the input buffer is determined by the detected symbol boundary. Other input parameters include length of the FFT and number of symbols to be demodulated. While the FFT is running on the coprocessor, the channel estimation and equalization of previous demodulated symbols can be executed on the the host processor. If the *intpt* ag is set, the demodulation coprocessor issues an interrupt when the rst symbol has been demodulated and written to the output FIFO. A sequence diagram of the OFDM demodulation procedure is illustrated in Fig. 25.

6.6 Espresso Limitations

The architecture and the programming interface of the Espresso does draw certain limitations to the supported traf c. First, simultaneous traf c through WCDMA and OFDM air interfaces can only be achieved if the baseband processing is time-multiplexed between WCDMA and OFDM modes. In order to support simultaneous reception of both systems, a separate sample stream would be needed for WCDMA and OFDM from the receiver front-end, which rules out the possibility of integrating

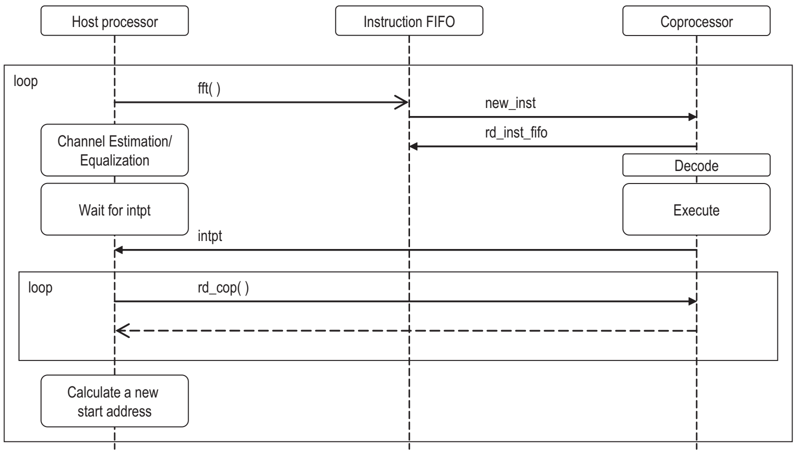


Fig. 25. *OFDM demodulation sequence diagram.*

the RF section and ADCs as depicted in Fig. 3. Furthermore, time-multiplexing the hardware resources of the baseband would lead to extensive clock frequencies.

Second, the width of the coprocessor instruction word is x ed to the internal word width of the COFFEE core which is 32 bits. This draws limitations to the range of the input parameters of the coprocessor functions. For example, the width of the *start_addr* parameter of the *despread* and *fft* functions limits the length of the input buffer to 1024 samples. Similarly, the width of the parameters of the *set_track_param* function limits the length of the correlation window and number of averaging iterations used in the tracking mode of the synchronization coprocessor.

Third, changes to wireless standards or different variants of OFDM and WCDMA physical layers may require special type of hardware that is not currently supported by the synchronization and demodulation coprocessors. For example, the demodulation coprocessor only includes generators for scrambling codes and orthogonal variable spreading factor (OVSF) codes required in 3GPP physical layer. If the number or type of code generators is not suf cient, the coprocessor architecture need to be changed.

Fourth, the programmability of the Espresso does not cover the execution of the coprocessor functions. The functions provided are designed to cover the required WCDMA and OFDM baseband procedures, but adding new air interfaces to the baseband does require changes in the coprocessor architectures. Considering the Software

De ned Radio Forum tiers mentioned in section 2.2, the Espresso platform falls into tier 2, but does not support the ability to add new air interfaces through software updates.

7. ESPRESSO SIMULATION AND SYNTHESIS

The first step in the design process of the Espresso platform was to model the whole transmission chains of WCDMA and OFDM systems with Matlab. The models included random symbol generation, slot or packet construction, modulation, baseband channel modeling, synchronization, demodulation, and channel estimation. The purpose of these models was to provide a reference implementation of the receiver algorithms and to serve as a test data generator for future simulations. Furthermore, the most critical computation kernels of the WCDMA and OFDM receiver algorithms were identified during these simulations. The Matlab models were then converted to fixed point versions employing SystemC data types. All the functions were converted by hand to C-language syntax. During this step the data word widths for the coprocessors were selected. 8-bit samples for real and imaginary components of the baseband signal were used in both WCDMA and OFDM. As mentioned in chapter 4, this is adequate for WCDMA but draws certain limitations to the OFDM traffic.

Next, the SystemC model was mapped onto the platform components, and the functionality of the individual components were modeled with clock cycle and bit accurate SystemC models. The test bench is illustrated in Fig. 26. Microsoft Visual C++ compiler was used for compiling the SystemC simulation models, and the simulations were carried out using only the operating system shell as a simulation environment. The correctness of the test bench output was verified with Matlab, by comparing the received symbols to the transmitted ones. The main purpose of these simulations was to test the programming interface of the coprocessors and the interrupt based synchronization scheme between the host processor and the coprocessors. The simulations covered WCDMA cell search, multipath searcher, demodulation, channel estimation, and equalization procedures, as well as OFDM packet detection, symbol timing estimation, demodulation, channel estimation, and equalization procedures. The main finding of these simulations was that the range of coprocessor functions and parameters constituting the programming interface of the coprocessors can be successfully

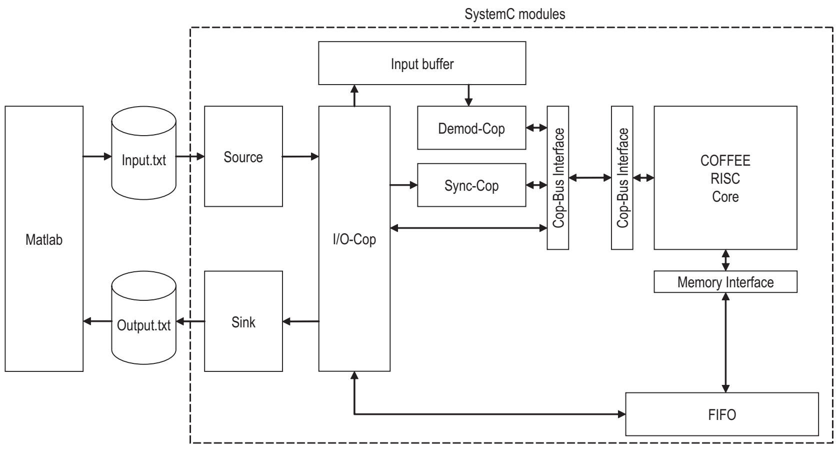


Fig. 26. *Espresso test bench.*

used to implement the mentioned WCDMA and OFDM baseband procedures. Furthermore, the minimum clock rates for the various baseband receiver procedures were estimated. The estimates were obtained by counting the clock cycles spent in a particular synchronization, demodulation, or channel estimation step and dividing the time available for that particular procedure with the clock cycle count. Simultaneous synchronization, demodulation, and channel estimation tasks of WCDMA requires a minimum clock frequency of 90 MHz. The minimum clock frequency for simultaneous OFDM demodulation and channel estimation was 120 MHz. Based on the simulation, the bottleneck of the system is the transfer of the demodulated symbols from the demodulation coprocessor to the host processor. For example, reading the 64 FFT outputs from the output FIFO of the demodulation coprocessor to the data memory has a signi cant effect on the required clock frequency.

Although the memory subsystem was only modeled as a FIFO, as shown in Fig. 26, the size of the data memory and the required memory bandwidth were estimated. Both WCDMA and OFDM receiver procedures can be implemented with approximately 2 kB of random access memory. This estimate is based on the number and type of variables used in the SystemC simulation models, and the effect of the processors stack was not considered. The peak memory bandwidth was estimated based on the symbol rates of WCDMA and OFDM, which determines how often the demodulation coprocessor produces its output, and thus, the rate at which the host processor needs to perform the channel equalization and write the demodulated symbols to memory. With the smallest spreading factor and three parallel code channels,

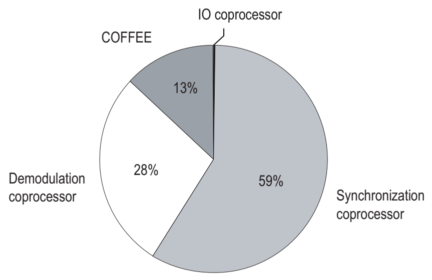


Fig. 27. *Espresso area distribution.*

the WCDMA symbol rate is $2.88 \cdot 10^6$ symbols per second. Assuming that the data memory is accessed at this rate, and 8-bits are used for both real and imaginary components of the symbols, the resulting memory bandwidth is 5.8 MB per second. Similarly, with $20 \cdot 10^6$ symbols per second and 16-bit symbols, the estimated memory bandwidth for OFDM is 40 MB per second.

The coprocessors were implemented with RTL VHDL description, and logic synthesis. A standard cell 0.13 μm technology was used in the synthesis. The main ndings of the synthesis was the total silicon area of the platform. The synthesis runs yielded 3.65 mm^2 area for the whole platform which is equivalent to approximately 452 300 gates. The total area was spread over the platform components as follows: 2.14 mm^2 for the synchronization coprocessor, 1.03 mm^2 for the demodulation coprocessor, 0.01 mm^2 for the I/O coprocessor, and 0.47 mm^2 for the COFFEE core. The area distribution is illustrated in Fig. 27. The memory system of the platform was not considered in the synthesis. The presented silicon areas are based on the area reports from the synthesis tool. Hence, the numbers do not include the contribution of the wiring to the area. The area is clearly dictated by the synchronization coprocessor. This is explained by the amount of registers required in the FIR block.

Direct comparison of the platform implementation to existing solutions was not feasible, because the implementation details of the related work covered in chapter 5 have not been published. However, existing single-mode WCDMA and OFDM were used as reference. An OFDM receiver implementation presented in [66] comprises automatic gain control, frequency correction, synchronization, demodulation, and channel estimation units. The chip is implemented with 0.18 μm technology and occupies 7.6 mm^2 area. It has to be noted that the chip is designed for digital video broadcasting applications and it utilizes a 1024-point FFT. The reported power consumption

Table 3. *Espresso synthesis and simulation results.*

		Synchronization	Demodulation	IO	COFFEE
	Area	2.14 mm^2	1.03 mm^2	0.01 mm^2	0.47 mm^2
Power	WCDMA	61.6 mW	33.8 mW	–	–
	OFDM	95.6 mW	47.2 mW	–	–

with 52 MHz clock frequency and 1.8 V supply voltage is 32 mW. A WCDMA Rake receiver implementation that comprises pulse shape ltering, frequency and timing recovery, multipath searcher, and MRC is presented in [94]. The chip is implemented with 0.18 μm technology and occupies 8.29 mm^2 area. The reported power consumption with 20 MHz clock frequency and 1.8 V supply voltage is 4 mW.

The power consumption of the coprocessors were simulated separately using the synthesized gate-level models. The switching activity from the simulations was stored and then back annotated to the synthesis tools to yield the power consumption estimates. The power consumption of the platform as a whole was not estimated because of the practical workload limitations of the simulation machines. The clock frequencies used in the estimations were the minimum clock frequencies needed in the WCDMA and OFDM modes of the platform. WCDMA synchronization with 90 MHz clock frequency yielded a power consumption of 61.6 mW for the synchronization coprocessor, and demodulation with 90 MHz clock frequency yielded a power consumption of 33.8 mW for the demodulation coprocessor. The WCDMA simulations were performed with spreading factor of 32 and three parallel code channels. OFDM synchronization with 120 MHz clock frequency yielded a power consumption of 95.6 mW for the synchronization coprocessor and OFDM demodulation with 120 MHz clock frequency yielded a power consumption of 47.2 mW for the demodulation coprocessor. Espresso simulation results are summarized in Table 3. As shown in the table, the power consumption of the IO coprocessor and the COFFEE core were not estimated. Because of the limited extent of the the power estimates, the results can be treated merely as sanity checks for the platform implementation. However, based on the presented numbers it can be concluded that the architectures of the synchronization and demodulation coprocessors enable high performance with moderate clock frequencies, and consequently are suited for practical implementations.

8. CONCLUSIONS

Interworking between cellular and ad hoc networks is an effective way to meet the diverse data throughput, latency, and coverage requirements of future wireless applications. Multistandard wireless systems require that users are equipped with mobile terminals that can operate in several wireless networks. As the evolution of wireless systems leads closer to the software radio concept, the required e xibility of the transceiver implementations increases even more, and effective solutions for implementing a multitude of radio technologies in a single device will be of paramount importance.

As a result of the design challenges introduced by time-to-market, complexity, and e xibility requirements, the role of software development will continue to increase and reusability of the software and hardware solutions will be essential. Combining the principles of platform based hardware design and product line based software design is the key to building a framework for ef cient reuse of hardware and software. In addition to the theoretical basis of the new design methodologies, their utilization in practice requires nancial investments, right management structure, EDA tool support, and prioritization of long term design productivity over short term nancial setbacks.

The programmability of the receiver hardware becomes more and more important as the number of supported radio technologies increases. The programmability has to cover also the sample rate processing of the receiver. The receiver algorithms of different radio technologies employ similar computation kernels which can be used effectively as functional building blocks in a multimode receiver. The algorithms needed in different radio technologies can be implemented by instantiating the kernels with different parameters.

A well suited approach for achieving the programmability of the sample rate receiver algorithms is to employ a RISC processor and attach coprocessors to accelerate the execution of the computation kernels. The key architectural choices of this approach are the interconnection between the host processor and the coprocessors, the synchronization of the concurrent processes of the host processor and the coprocessor, and the programming interface of the coprocessors.

8.1 Main Results

The main result of the work covered in this thesis is the Espresso platform that enables software implementation of the baseband processing of WCDMA and OFDM receivers. The required processing power for the sample rate receiver algorithms is achieved by utilizing coprocessors that implement the most critical computation kernels of WCDMA and OFDM receivers. The communication between the host processor and the coprocessors is realized through a dedicated coprocessor bus which minimizes the communication overhead inherent in typical accelerator based implementations. The programming interface for the coprocessors is implemented with a set of coprocessor functions that cover all the kernels needed in WCDMA and OFDM receivers. The parameters of the coprocessor functions provide enough e xibility to enable the employment of the platform in any receiver requiring WCDMA or OFDM connectivity. This enables effective reuse as the platform architecture can be utilized in several products inside the application eld determined by the WCDMA and OFDM radio technologies.

The programmability issues of typical application-speci c processors are solved by utilizing a RISC core as the host processor. High-level programming languages can be utilized for programming the platform because the host processor is an easy target for the compiler and because the implementation details of the coprocessors are hidden under a programming interface. The programming interface employs only two simple functions that require low-level programming. Thus, the software written for the platform is reusable when the programming interface of the coprocessors is kept unmodi ed.

The system level simulations of the platform have shown that the platform architecture and the programming interface provide the necessary resources to implement WCDMA and OFDM baseband procedures. The area and power consumption of the platform were only estimated at gate level, but the results are comparable to reference single-mode WCDMA and OFDM implementations which indicates that the platform can be used in practical implementations.

The Espresso platform does have limitations on the type of data traf c and range of physical layer procedures supported. Simultaneous traf c through the WCDMA and OFDM air interfaces is not possible unless the baseband processing is time-multiplexed between the modes of the receiver. Furthermore, the programmability of the platform does not extend to the coprocessor functions, and thus, adding new functions needed for an air interface is not possible without changing the design of the coprocessors.

8.2 Future Trends

The evolution of wireless communications will carry on in two directions: evolution of the 3G networks towards Super 3G and convergence of cellular and ad hoc networks. The biggest bene t from this evolution, considering the end user, will be higher data rates, ubiquitous coverage, and the emergence of new wireless services. The lack of killer applications has hindered the success of 2.5G and 3G systems but as the evolution of wireless systems opens new business models, new revolutionary wireless applications will emerge.

The multitude of radio technologies and the convergence of different type of networks will continue to be a challenge for the mobile terminal implementations. In addition to the existing radio technologies, future mobile terminals will have to support new technologies based on UWB, multicarrier, and multiantenna techniques. In future, different types of networks will be employed also simultaneously which increases the transceiver complexity even further. For example, users might want to browse the internet through WLAN, while making a voice call through a cellular network. Moreover, the mobile terminals should be able to detect the available networks and to select the appropriate connection based on the desired application.

The evolution of SoC design methodologies is likely to follow the tenets of platform based hardware development and product line based software development. The transition will be a slow one as legacy designs require design teams to commit to the old design methodologies, and the nancial setbacks of a complete paradigm change are too large. EDA tools will be the enabling factor to boost design productivity in the future. It is likely that complete tool suites will emerge that combine the hardware and software domains and automate the mapping of the functional system level models to the available software and hardware platforms.

The architectures employed in future mobile terminal transceiver will comprise more and more programmable components. Before the current semiconductor technologies reach the limit of their performance, the available processing power will keep on increasing, enabling more and more functionality to be implemented with software. Processor architectures deployed in receiver implementations are likely to follow the RISC principle because of their compiler friendliness. The biggest challenges related to programmable architectures will be the development of intelligent compiler methodologies. This is the key in increasing the design productivity as the programmer needs to be freed from the awareness of the architectural details of the hardware. If pursued successfully, this will also alleviate software reuse.

It can be argued whether the software radio concept will be employed in commercial systems in its original form. However, the concept certainly projects many attractive features and steers the evolution of wireless communications to a similar direction. The ongoing standardization work of future systems aims at global standards which would diminish the appeal of software radio if pursued successfully. Nevertheless, it is likely that the vision of software radio terminals will be realized. The current evolution of SoC design challenges and methodologies will eventually lead to a situation where the transceiver implementations are in fact implemented completely with software.

9. SUMMARY OF PUBLICATIONS

The FlexRake receiver concept is introduced in [P1]. The shortcomings of traditional nger based Rake architectures are highlighted and the basic operation of the FlexRake is described. Hardware implementation requirements are given and the system level simulations of the architecture are presented.

The hardware implementation of the FlexRake is presented in [P2]. Two versions of the architecture are presented and the implementation details are explained. Estimates of the silicon area and power consumption are given, and the difference between the two implementation alternatives are analyzed.

A more extensive analysis of traditional Rake architectures and the FlexRake is presented in [P3]. The functionality of the individual blocks of the FlexRake are explained in more detail. The implementation details of the address computation and code generators are also presented.

The Espresso platform is introduced in [P4]. The implementation issues of a dual-mode WCDMA/OFDM receiver, including the RF front-end and analog-to-digital conversion parts, are addressed. The baseband receiver algorithms of WCDMA and OFDM receivers are studied, and the feasibility of a dual-mode baseband implementation is investigated. The high-level architecture of the Espresso platform is presented and the coprocessor functionalities are described.

The programming interface of the Espresso is studied in more detail in [P5]. The programming interface of the coprocessors are described and a list of the coprocessor functions is given. System-level simulation results of the platform are presented.

The implementation of the synchronization coprocessor is presented in [P6]. The functionality and architecture of the coprocessor are described in more depth, and the coprocessor functions and their input parameters are presented. Synthesis results and power consumption estimations are given.

The implementation of the demodulation coprocessor is presented in [P7]. The datapath of the coprocessor and the implementation of the processing units are studied in detail. The coprocessor functions and their input parameters are presented. Synthesis results and power consumption estimations are given.

9.1 Author's Contribution to the Publications

The Author is the primary author in all seven publications. The co-authors of the publications have agreed with the following descriptions of their contributions.

The FlexRake concept introduced in [P1] was developed by Dr. Kuulusa. The Author contributed to the development of the system-level FlexRake architecture, and was responsible of building the simulation model of the receiver. The publication was co-authored by Dr. Kuulusa and by Professor Nurmi, who also supervised the work.

The implementation details of the FlexRake architecture presented in [P2] were designed by the Author, with support from Dr. Kuulusa and Professor Nurmi. The VHDL description, synthesis, and power consumption estimations were carried out by the Author. Dr. Kuulusa and Professor Nurmi contributed as co-authors. Professor Nurmi also supervised the work.

The work presented in [P3] comprised a more detailed study of traditional Rake architectures and the FlexRake architecture carried out by the Author. The implementation of the code generators also presented in the publications were done by M.Sc. Timo Rintakoski and Dr. Kuulusa. The publication was co-authored by Dr. Kuulusa and by Professor Nurmi.

The Espresso platform introduced in [P4] was based on the original idea of the Author. Professor Nurmi contributed to the development of the platform architecture. The details of the programming interface and the detailed architectures of the coprocessors presented [P5], [P6], [P7] were designed by the Author. Professor Nurmi provided valuable input and co-authored the publications.

BIBLIOGRAPHY

[1] Y. Neuvo, “Cellular Phones as Embedded Systems,” in *Digest of Technical Papers IEEE Solid-State Circuits Conference*, Feb. 2004, vol. 1, pp. 32–37.

[2] R. Tafazolli, Ed., *Technologies for the Wireless Future*. West Sussex, UK: John Wiley & Sons, Ltd., 2005.

[3] A. Doufexi, E. Tameh, A. Nix, S. Armour, and A. Molina, “Hotspot Wireless LANs to Enhance the Performance of 3G and Beyond Cellular Networks,” *IEEE Communications Magazine*, vol. 41, no. 7, pp. 58–65, July 2003.

[4] *Spreading and Modulation (FDD) (Release 5)*, 3GPP Technical Speci cation 25.213, Rev. 5.4.0, 2003.

[5] E. Dahlman, B. Gudmundson, M. Nilsson, and A. Skold, “UMTS/IMT-2000 Based on Wideband CDMA,” *IEEE Communications Magazine*, vol. 36, no. 9, pp. 70–80, Sept. 1998.

[6] H. Holma and A. Toskala, *WCDMA for UMTS*. West Sussex, England: John Wiley & Sons, Ltd., 2001.

[7] S. Parkvall, E. Englund, M. Lundevall, and J. Torsner, “Evolving 3G Mobile Systems: Broadband and Broadcast Services in WCDMA,” *IEEE Communications Magazine*, vol. 44, no. 2, pp. 30–36, Feb. 2006.

[8] *Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN) (Release 7)*, 3GPP Technical Report 25.913, Rev. 7.3.0, 2006.

[9] H. Ekstrom *et al.*, “Technical Solutions for the 3G Long-Term Evelution,” *IEEE Communications Magazine*, vol. 44, no. 3, pp. 38–45, Mar. 2006.

[10] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE Standard 802.11b, 1999.

[11] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; High-speed Physical Layer in the 5 GHz Band*, IEEE Standard 802.11a, 1999.

[12] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Further High Data Rate Extension in the 2.4 GHz Band*, IEEE Standard 802.11g, 2003.

[13] *3GPP System to Wireless Local Area Network (WLAN) Interworking; System Description (Release 6)*, 3GPP Technical Speci cation 23.234, Rev. 6.1.0, 2004.

[14] *Generic Access to the A/Gb Interface; Stage 2 (Release 6)*, 3GPP Technical Speci cation 43.318, Rev. 6.5.0, 2006.

[15] *Feasibility Study on 3GPP System to WLAN Interworking (Release 6)*, 3GPP Technical Speci cation 23.934, Rev. 6.2.0, 2003.

[16] J. Mitola, "Software Radios: Survey, Critical Evaluation and Future Directions," in *Proc. National Telesystems Conference*, Washington, DC, USA, May 1992, pp. 15–23.

[17] ——, "The Software Radio Architecture," *IEEE Communications Magazine*, vol. 35, no. 5, pp. 26–38, May 1995.

[18] W. Tuttlebee, Ed., *Software Defined Radio: Enabling Technologies*. West Sussex, UK: John Wiley & Sons, Ltd., 2002.

[19] J. Walko, "Cognitive Radio," *IEE Review*, vol. 51, no. 5, pp. 34–37, May 2005.

[20] "Software De ned Radio Forum," website, http://www.sdrforum.org.

[21] "International Technology Roadmap for Semiconductors," 2005.

[22] B. Graaf, M. Lormans, and H. Toetenel, "Embedded Software Engineering: The State of the Practice," *IEEE Software*, vol. 20, no. 6, pp. 61–69, Nov. 2003.

[23] F. R. Wagner, W. O. Cesario, L. Carro, and A. Jerraya, "Strategies for the Integration of Hardware and Software IP Components in Embedded Systems-on-Chip," in *Integration, the VLSI Journal*. Elsevier B. V., Sept. 2004, vol. 37, no. 4, pp. 223–252.

[24] T. A. C. M. Claasen, "Platform Design: The Next Paradigm Shift to Deal with Complexity," in *Proc. Symposium on VLSI Technology, Systems, and Applications*, Hsinchu, Taiwan, Oct. 2003, pp. 8–12.

[25] "OMAPV1030," Product Bulletin, Texas Instruments, Inc., 2005.

[26] A. Bernstein, M. Burton, and F. Ghenassia, "How to Bridge the Abstraction Gap in System Level Modeling and Design," in *Proc. IEEE/ACM Conference on Computer Aided Design*, San Jose, CA, USA, Nov. 2004, pp. 910–914.

[27] "Embedded Systems Roadmap 2002," STW Technology Foundation, Mar. 2002.

[28] R. van Ommering, "Software Reuse in Product Populations," *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 537–550, July 2005.

[29] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley, 2002.

[30] L. M. Northrop, "SEI's Software Product Line Tenets," *IEEE Software*, vol. 19, no. 4, pp. 32–40, July 2002.

[31] "Technology Roadmap for Software-Intensive Systems," Information Technology for European Advancement, May 2004.

[32] J. Bosch, "Organizing for Software Product Lines," in *Proc. International Workshop on Software Architectures for Product Families*, Las Palmas, Gran Canaria, Spain, Mar. 2000, pp. 117–134.

[33] *Physical Layer - General Description (Release 1999)*, 3GPP Technical Speci - cation 25.201, Rev. 3.0.2, 2003.

[34] A. Loke and F. Ali, "Direct Conversion Radio for Digital Mobile Phones–Design Issues, Status, and Trends," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 11, pp. 2422–2435, Nov. 2002.

[35] A. Loke and M. Abdelgany, "Multi Mode Wireless Terminals–Key Technical Challenges," in *Digest of Technical Papers IEEE Radio Frequency Integrated Circuits Symposium*, Philadelphia, PA, USA, June 2003, pp. 11–14.

[36] M. Hotti *et al.*, "A Direct Conversion RF Front-End for 2-GHz WCDMA and 5.8-GHz WLAN Applications," in *Proc. IEEE Radio Frequency Integrated Circuits Symposium*, Philadelphia, PA, USA, June 2003, pp. 45–48.

[37] T. Manku *et al.*, "A Single Chip Direct Conversion CMOS Transceiver for Quad-band GSM/GPRS/EDGE and WLAN with Integrated VCO's and Fractional-N Synthesizer," in *Digest of Technical Papers IEEE Radio Frequency Integrated Circuits Symposium*, Forth Worth, TX, USA, June 2004, pp. 423–426.

[38] L. Sumanen, "Pipeline Analog-to-Digital Converters for Wide-Band Wireless Communications," Ph.D. dissertation, Helsinki University of Technology, Helsinki, Finland, Dec. 2002.

[39] T. H. Meng, B. McFarland, D. Su, and J. Thomson, "Design and Implementation of an All-CMOS 802.11a Wireless LAN Chipset," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 160–168, Aug. 2003.

[40] A. Richardson, *WCDMA design handbook*. Cambridge, UK: Cambridge University Press, 2004.

[41] J. B. Groe and L. E. Larson, *CDMA Mobile Radio Design*. Norwood, MA, USA: Artech House, 2000.

[42] J. S. Lee and L. E. Miller, *CDMA Systems Engineering Handbook*. Norwood, MA, USA: Artech House, 1998.

[43] R. van Nee and R. Prasad, *OFDM Wireless Multimedia Communications*. Norwood, MA, USA: Artech House, 2000.

[44] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*. Indianapolis, IN, USA: Sams Publishing, 2002.

[45] J.-J. van de Beek, M. Sandell, and P. O. Borjersson, "ML Estimation of Time and Frequency Offset in OFDM Systems," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800–1805, July 1997.

[46] J.-J. van de Beek *et al.*, "A Time and Frequency Synchronization Scheme for Multiuser OFDM," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 11, pp. 1900–1914, Nov. 1999.

[47] Y.-P. Wang and T. Ottoson, "Cell Search in W-CDMA," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 8, pp. 1470–1482, Aug. 2000.

[48] N. Darbel, Y. Rasse, B. Jubelin, and M. Carrie, "A UMTS-FDD Cell Search Engine," in *The Journal of VLSI Signal Processing*. Springer Science, Aug. 2004, vol. 38, no. 1, pp. 73–84.

[49] *Physical Channels and Mapping of Transport Channels onto Physical Channels (FDD) (Release 5)*, 3GPP Technical Speci cation 25.211, Rev. 5.5.0, 2003.

[50] A. Huang, M. Hall, and I. Hartimo, "Multipath Channel Estimation for WCDMA Uplink," in *Proc. IEEE Vehicular Technology Conference*, vol. 1, Amsterdam, Netherlands, Sept. 1999, pp. 141–145.

[51] E.-S. Lohan, "Multipath Delay Estimators for Fading Channels in CDMA Receivers and Mobile Positioning," Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, Oct. 2003.

[52] K.-C. Gan, "Path Searcher for a WCDMA Rake Receiver," Application Note AN2252, Freescale Semiconductor, Mar. 2005.

[53] E. Grayver *et al.*, "Design and VLSI Implementation for a WCDMA Multipath Searcher," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 3, pp. 889–902, May 2005.

[54] R. Tanner and J. Woodard, *WCDMA–Requirements and Practical Design*. West Sussex, UK: John Wiley & Sons, Ltd., 2004.

[55] R. G. Lyons, *Understanding Digital Signal Processing*. Boston, MA, USA: Addison-Wesley, 1999.

[56] C. Hsu, Y.-H. Huang, and T.-D. Chiueh, "Design of and OFDM Receiver for High-Speed Wireless LAN," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 4, Sydney, Australia, May 2001, pp. 558–561.

[57] A. Zhuang, E.-S. Lohan, and M. Renfors, "Comparison of Desicion-Directed and Pilot-Aided Algorithms for Complex Channel Tap Estimation in a Downlink WCDMA System," in *Proc. IEEE Symposium on Personal, Indoor, and Mobile Radio Communications*, London, UK, Sept. 2000, pp. 1121–1125.

[58] A. Aziz, "Channel Estimation for a WCDMA Rake Receiver," Application Note AN2253, Freescale Semiconductor, Nov. 2004.

[59] J.-J. van de Beek, O. Edfors, M. Sandell, S. K. Wilson, and P. O. Borjesson, "On Channel Estimation in OFDM Systems," in *Proc. IEEE Vehicular Technology Conference*, vol. 2, Chicago, IL, USA, July 1995, pp. 815–819.

[60] Y. Shen and E. Martinez, "Channel Estimation in OFDM Systems," Application Note AN3095, Freescale Semiconductor, Jan. 2006.

[61] D. A. Patterson and J. L. Hennesy, *Computer Organization & Design*, 2nd ed. San Fransico, CA, USA: Morgan Kaufmann Publishers, Inc., 1998.

[62] P. Lapsley, J. Brier, and A. Shoham, *DSP Processor Fundamentals*. New York, NY, USA: IEEE Press, 1997.

[63] A. Gatherer, T. Stetzler, M. McMahan, and E. Auslander, "DSP-Based Architectures for Mobile Communications: Past, Present and Future," *IEEE Communications Magazine*, vol. 43, no. 15, pp. 1244–1245, 2000.

[64] R. Kokozinski, D. Greindorf, J. Stammen, and P. Jung, "Evolution of Hardware Platforms for Mobile Software De ned Radio Terminals," in *Proc. IEEE Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 5, Lisbon, Portugal, Sept. 2002, pp. 2389–2393.

[65] P. Jung, P. Schmidt, and J. Plechinger, "Implementation Aspects of Mobile UMTS FDD Receiver," in *Proc. IEE Colloquium on UMTS Terminals and Software Radio*, Glasgow, UK, Apr. 1999, pp. 1–6.

[66] H. Zou and B. Daneshrad, "VLSI Implementation for a Low Power Mobile OFDM Receiver ASIC," in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 4, Atlanta, GA, USA, Mar. 2004, pp. 2120–2124.

[67] D. A. Patterson, "Reduced Instruction Set Computers," *Communications of the ACM*, vol. 28, no. 1, pp. 8–12, Jan. 1985.

[68] J. Kylliäinen, M. Kuulusa, and J. Nurmi, "COFFEE—A Core for Free," in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2003, pp. 17–22.

[69] "COFFEE RISC Core," website, http://coffee.tut. .

[70] W. Lee *et al.*, "A 1-V Programmable DSP for Wireless Communications," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1766–1776, Nov. 1997.

[71] J. Glossner, T. Raja, E. Hokenek, and M. Moudgill, "Multithreaded Processor for SDR," *Proc. Korea Institute of Communication Sciences*, vol. 19, no. 1, pp. 70–84, Nov. 2002.

[72] J. Glossner, E. Hokenek, and M. Moudgill, "Multithreaded Processor for Software De ned Radio," in *Proc. Software Defined Radio Technical Conference*, vol. 1, San Diego, CA, USA, Nov. 2002, pp. 195–199.

[73] J. Glossner, D. Iancu, J. Lu, E. Hokenek, and M. Moudgill, "A Software-De ned Communications Baseband Design," *IEEE Communications Magazine*, vol. 41, no. 1, pp. 120–128, Jan. 2003.

[74] M. F. Jacome and G. D. Veciana, "Design Challenges for New Application-Speci c Processors," *IEEE Design and Test of Computers*, vol. 17, no. 2, pp. 40–50, Apr. 2000.

[75] R. Leupers *et al.*, "Retargetable Compilers and Architecture Exploration for Embedded Processors," *IEE Proc. Computers and Digital Techniques*, vol. 152, no. 2, pp. 209–223, Mar. 2005.

[76] N. Clark, H. Zhong, and S. Mahike, "Automated Custom Instruction Generation for Domain-Speci c Processor Acceleration," *IEEE Transactions on Computers*, vol. 54, no. 10, pp. 1258–1270, Oct. 2005.

[77] A. Hoffmann, F. Fiedler, A. Nohl, and S. Parupali, "A Methodology and Tooling Enabling Application Speci c Processor Design," in *Proc. International Conference on VLSI Design*, Kolkata, India, Jan. 2005, pp. 399–404.

[78] "Xtensa LX," Product Brief, Tensilica, Inc., 2004.

[79] "An Independent Overview of Tensilica Xtensa LX Processor with Vectra LX," Berkeley Design Technology, Inc., 2005.

[80] Y.-H. Huang, H.-P. Ma, M.-L. Liou, and T.-D. Chiueh, “A 1.1 G MAC/s Sub-Word-Parallel Digital Signal Processor for Wireless Communication Applications,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 169–183, Jan. 2004.

[81] J. H. Lee, K. Heo, and M. H. Sunwoo, “Implementation of Application-Speci c Signal Processor for High-Speed Communication Systems,” in *Proc. International Symposium on Intelligent Signal Processing and Communication Systems*, Seoul, South Korea, Nov. 2004, pp. 250–255.

[82] F. Honore, W. Gass, A. Gatherer, and S. Sriram, “Implementation Options for WCDMA,” in *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing*, Istanbul, Turkey, June 2000, pp. 3702–3705.

[83] B. Hounsell and R. Taylor, “Co-processor Synthesis: A New Methodology for Embedded Software Acceleration,” in *Proc. Design, Automation and Test in Europe Conference*, Paris, France, Feb. 2004, pp. 682–683.

[84] H.-M. Bluethgen, C. Grassmann, W. Raab, U. Ramacher, and J. Hausner, “A Programmable Baseband Platform for Software De ned Radio,” in *Proc. Software Defined Radio Technical Conference*, Pheonix, AZ, USA, Nov. 2004, paper 3.4-02.

[85] H.-M. Bluethgen *et al.*, “Finding the Optimum Partitioning for Multi-Standard Radio Systems,” in *Proc. Software Defined Radio Technical Conference*, Garden Grove, CA, USA, Nov. 2005, paper 1.5-01.

[86] A. Gatherer, “Texas Instrucments TCI Platform: A Cost Effective, Programmable Besestation Modem,” in *Proc. Software Defined Radio Technical Conference*, Orlando, FL, USA, Nov. 2003, paper HW1-03.

[87] S. Sriram *et al.*, “A 64 Channel Programmable Receiver Chip for 3G Infrastructure,” in *Proc. IEEE Custom Integrated Circuits Conference*, San Jose, CA, USA, Sept. 2005, pp. 59–62.

[88] J.-J. van de Beek, M. Sandell, M. Isaksson, and P. O. Borjersson, “Low-Complex Frame Synchronization in OFDM Systems,” in *Proc. IEEE Conference on Universal Personal Communications*, Tokyo, Japan, Nov. 1995, pp. 982–986.

[89] Z. Ye, Y. Kim, and A. Schooler, “A Flexible Chip Rate Processor for CDMA Rake Receivers,” in *Proc. Software Defined Radio Technical Conference*, Orlando, FL, USA, Nov. 2003, paper HW1-02.

[90] B. D. Andreev, E. L. Titlebaum, and E. G. Friedman, “Low Power Flexible Rake Receivers for WCDMA,” in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 4, Vancouver, Canada, May 2004, pp. 97–100.

[91] M. Chugh, D. Bhatia, and P. T. Balsara, “Design and Implementation of Con- gurable W-CDMA Rake Receiver Architectures on FPGA,” in *Proc. IEEE International Parallel and Distributed Processing Symposium*, Denver, CO, USA, Apr. 2005, p. 145b.

[92] S.-C. Han and R. Negi, “Early-Stopping for Rake Receivers,” in *Proc. International Symposium on Wireless Communication Systems*, Siena, Italy, Sept. 2005, pp. 245–249.

[93] T. Taskinen, “Hardware Implementation Architectures for Time-Frequency Transforms,” Master’s thesis, Tampere University of Technology, Tampere, Finland, Apr. 2002.

[94] A. M. Eltawil and B. Daneshrad, “A Low-Power DS-CDMA Rake Receiver Utilizing Resource Allocation,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1321–1330, Aug. 2004.

Part II

PUBLICATIONS

PUBLICATION 1

L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Rake Receiver Architecture for WCDMA Mobile Terminals,” in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications*, Tao Yuan, Taiwan, Mar. 2001, pp. 9–12.

Third IEEE Signal Processing Workshop on Signal Processing Adavances in Wireless Communications,Taoyuan, Taiwan, March 20-23, 2001

A Flexible Rake Receiver Architecture for WCDMA Mobile Terminals

Lasse Harju, Mika Kuulusa and Jari Nurmi
Digital and Computer Systems Laboratory
Tampere University of Technology
P.O. Box 553 (Hermiankatu 12),
Tampere, Finland
Tel. +358 3 365 4365, Fax. +358 3 365 3095,
Email: lasse.harju@tut.fi

Abstract — **This paper presents a novel Rake receiver architecture for WCDMA FDD downlink reception in mobile terminals. In contrast to conventional Rake finger approach, the proposed FlexRake architecture performs signal reception with a single correlator engine and a buffer which stores the entire delay spread of the baseband I/Q samples. This allows each of the tracked multipaths to be despread sequentially with codes that are exactly in the same phase. The main benefits of the proposed receiver architecture are flexible multipath allocation, symbol-synchronous operation, and straightforward receiver control.**

I. Introduction

This paper considers the frequency-division duplex (FDD) mode of the wideband code-division multiple access (WCDMA) standard [1, 2]. The main physical parameters for the FDD downlink are the following: QPSK data modulation, time-multiplexed control and user data channels, 3.84 Mcps chip rate, spreading factors between 4 and 512, orthogonal variable spreading factor (OVSF) codes for spreading (channelization), and Gold codes for complex scrambling. Variable user data rates are realized either by discontinuous transmission with a fixed spreading factor or by allocating multiple spreading code channels. Maximum downlink user data rate of 2.3 Mbit/s can be achieved using three parallel code channels that have a spreading factor of four [1].

The first WCDMA receivers are based on Rake receiver. The main principle of Rake receivers is that they exploit multipath propagation by receiving the multipath components of the transmitted signal separately and combining their energies. In conventional Rake architectures multipath components with significant signal energies are tracked and despread with dedicated Rake fingers.

In this paper we present a novel Rake receiver architecture for WCDMA mobile terminals. The proposed FlexRake architecture combines hardware efficiency, flexibility, and easy controllability. The paper is organized as follows. First, receivers based on Rake finger banks are discussed. Then the FlexRake receiver concept and its architecture is studied in detail. The main implementation requirements are investigated and the high-level receiver model is briefly described. Finally, the conclusions are drawn.

II. Conventional Rake Receivers

A conventional Rake receiver is depicted in Fig. 1. The input to the Rake receiver is a direct I/Q baseband sample stream.

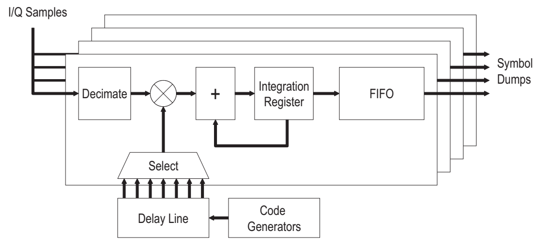


Fig. 1: Block diagram of a conventional Rake receiver based on multiple Rake finger devices.

In order to obtain higher multipath resolution the baseband signal is oversampled at 4-8 times the chip rate. After pulse shaping filtering and multipath estimation the sample stream is selectively decimated back to the chip rate by tracking the samples closest to the chip interval midpoints. In order to receive several multipath components of the transmitted signal, a dedicated Rake finger is allocated to each of the tracked components. Thus the Rake finger count corresponds to the maximum number of multipaths which is typically between two and six [3, 4, 5]. In a Rake finger, the received I/Q samples are correlated with a time-aligned spreading code and integrated over a period corresponding to the spreading factor. The time-alignment is typically carried out with a multiplexer that selects a specific phase of the code from a delay line [4]. Because the delay spread can be several times longer than the symbol integration period, the symbol dumps for a specific data symbol are completed at different times. Clearly this issue is particularly observable for high data rates with low spreading factors. Therefore, each Rake finger stores symbol dumps in a deskew buffer from which they can be accessed for channel correction and combining after all multipath symbol dumps are available [3, 4]. The maximum delay spread[1] and the lowest spreading factor supported by a Rake finger specifies the size of the deskew buffer, the size of the code delay line and the width of the code multiplexer.

Even though the operations required in a Rake finger are not computationally demanding, the implementation of a Rake

[1]Maximum delay spread is defined as the longest expected time difference between the first and the last received multipath component.

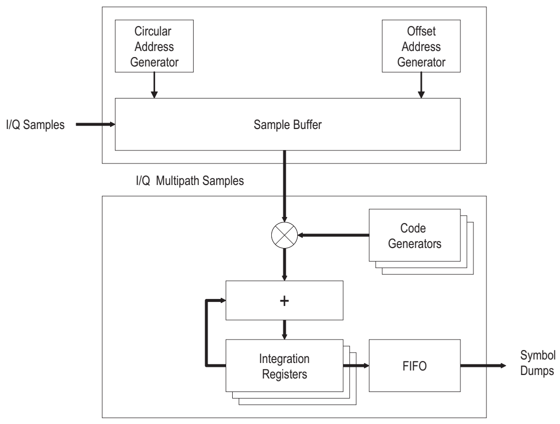


Fig. 2: Block diagram of FlexRake receiver architecture consisting of a stream buffer (upper half) and a correlator engine (lower half).

receiver is challenging because the receiver should support different spreading factors and parallel code channels, and also allow easy finger control according to the rapidly changing radio channel. One major disadvantage of the Rake finger in Fig. 1 is that it constantly has to adjust the phases of the code generators because of the rapidly changing multipath delays. Even more important, it can not perform correlation with passed I/Q samples. In a case when a Rake finger is assigned to the first arriving multipath component, and the delay of that multipath rapidly shortens, it fails to continue despreading because the sample stream can not be rewinded. Thus, a multipath component is lost and the multipath diversity degrades. Furthermore, in conventional Rake receiver architectures the overall hardware complexity increases for multicode reception because each finger requires as many despreaders as there are parallel code channels.

III. FlexRake Receiver Concept

Instead of using a number of dedicated fingers, FlexRake receiver performs correlation operations sequentially by accessing a buffer that serves as a time-sliding window to the received I/Q samples. A detailed block diagram of the FlexRake receiver is shown in Fig. 2. The FlexRake receiver contains two main units: Stream Buffer and Correlator Engine. The Stream Buffer stores the input stream and tracks multipath samples with a special addressing method controlled by multipath delay estimates. The Correlator Engine reads the multipath samples from the Stream Buffer and performs the despreading of the multipath components sequentially.

A. Stream Buffer

The Stream Buffer contains a sample buffer and two address generators. The sample buffer stores the I/Q sample pairs coming from the pulse shaping filtering. The sample buffer, depicted in Fig. 3, can be comprehended as a time-sliding window that is divided into three parts: write window, pre-window, and post-window. The write window allows writing to the buffer without overlapping the pre-window and the post-window needed to carry out multipath read accesses. These read and write accesses are interleaved in time in order to avoid the need of concurrent memory accesses. Whereas the post-window contains

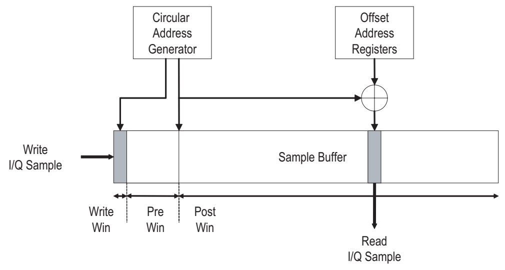


Fig. 3: Principle of the Stream Buffer read and write accesses.

the I/Q samples within the longest supported delay spread, the purpose of the pre-window is to add headroom for the movement of the first arriving multipath components. Even if the delays shorten considerably, the pre-window ensures that multipath samples are not lost because they can be despread from the pre-window.

Circular Address Generator provides a stream of sequential sample buffer cursor and write addresses. The cursor address points to the beginning of the post-window and it is incremented periodically after each processing cycle that is equal to the chip duration. Similarly, the write address points to the write window and is also incremented after each processing cycle. Offset Address Generator is employed for fetching I/Q multipath samples for correlation from the sample buffer. It contains a number of offset address registers which are controlled by multipath delay estimates. The number of offset address registers L corresponds to the maximum number of tracked multipath components and the offset values correspond to the delays of the tracked multipath components. As shown in Fig. 3, the effective read addresses are calculated by summing two values: the cursor address and an offset value.

Each multipath component is read to the Correlator Engine one at a time for despreading. After each processing cycle the sample buffer cursor and write addresses are incremented, the offset values can be updated, and new I/Q samples are written to the sample buffer. A processing cycle in the Stream Buffer contains a number of read and write accesses that correspond to the number of the tracked multipath components and the oversampling ratio, respectively.

B. Correlator Engine

Correlator Engine contains a complex correlator, code generators for channelization and scrambling codes, a number of integration registers, and a FIFO buffer for symbol dumps. The number of integration registers N_{ireg} defines the maximum number of concurrent symbol integrations. Therefore, for a single code $N_{ireg} = L$ and for three parallel code channels $N_{ireg} = 3L$.

The correlator performs a complex-valued correlation of the I/Q multipath samples with a combined OVSF/Gold code produced by the two code generators. Partial symbol integration results of each multipath component are stored in an integration register. Since the I/Q multipath samples are read from the sample buffer sequentially, all correlations can be performed using the same code phase. When multicode transmission is employed, L integration registers and a dedicated channelization code is assigned for each additional code channel. After correlating over one symbol period the final symbol dumps are stored into the FIFO buffer. It is important to note that since the multipath components are despread sequentially, L symbol dumps for a transmitted data symbol appear in a certain sequential order.

One processing cycle in the Correlation Engine is divided into a number of correlation cycles. On each correlation cycle, one (single code) or multiple (multicode) correlations are performed with each I/Q multipath sample. Thus for four multipath components ($L = 4$) and three parallel code channels ($N_{code} = 3$), one processing cycle in the Correlation Engine may include up to 12 correlation cycles.

C. Control

The functionality of the FlexRake is pipelined into two stages for Stream Buffer write/read cycles and Correlator Engine correlation cycles. The FlexRake contains a control unit which is needed to schedule various operations. The offset value updates are received from a programmable DSP processor which uses a dedicated multipath estimator unit to resolve the multipath delay profile of the radio channel. Typically, new offsets are generated every 10 ms interval, i.e. on a time frame basis. It is assumed that the offset update rate is sufficiently fast to avoid the need for delay-lock loops (DLLs) for code tracking. Furthermore, the control unit has three operational modes: receiver initialization, steady-state reception, and sleep modes.

There are a number of advantages gained from the FlexRake receiver architecture. First is the high flexibility of the multipath allocation because multipath components are tracked simply with the offset values and by allocating a dedicated integration register. Furthermore, the sample buffer pre-window allows the tracked multipath to move to earlier positions in the delay spread without being lost, i.e. negative offset values can be used. This ensures that no I/Q multipath samples are lost even if the delay profile is rapidly changing. Another advantage is that the OVSF/Gold code generators need not be time-aligned separately according to the multipath delays. Furthermore, multicode reception is more straightforward because the same I/Q multipath sample can be correlated with multiple spreading codes and thus it is not necessary to perform several reads from the same buffer address. Since the operation of the FlexRake receiver is symbol-synchronous, the symbol dumps of each multipath component are completed sequentially in time. This facilitates the implementation of the further processing operations, such as channel estimation and channel correction/combining.

IV. FlexRake Hardware Implementation Requirements

A. Operating Frequencies

In the Stream Buffer, the sample buffer is realized with a SRAM block which is accessed for writes at the baseband sample rate and for reads at an integer multiple of the chip rate. In the FlexRake architecture, a 1-port SRAM block is employed because of its lower silicon area with respect to a 2-port SRAM. Therefore, the required sample buffer read/write access frequency can be calculated with:

$$f_{sb} = f_s + Lf_c = (R + L)f_c \tag{1}$$

	T_{win}				
N_{sample}	4.17μs	8.33μs	16.67μs	33.33μs	66.67μs
6 bits	96	192	384	768	1024
8 bits	128	256	512	1024	2048
10 bits	160	320	640	1280	2560
12 bits	192	384	768	1536	3072

Tab. 1: Examples of sample buffer sizes in bytes.
($T_c = 1/(3.84 \cdot 10^6)\mu$s, $T_{win} = T_c + T_{pre} + T_{post}$, and $R = 4$)

where f_s is the sample rate, L is the number of tracked multipath components, f_c is the chip rate, and R is the oversampling ratio. Sample rate is computed with $f_s = Rf_c$. Assuming $f_c = 3.84$ MHz, $R = 4$, and $L = 4$, the resulting read and write accesses require a memory bandwidth of 30.72 MHz which is quite reasonable. The multicode reception does not add to these requirements because a I/Q sample is only read once and correlated several times with different codes.

The Correlator Engine operates at an integer multiple of the chip rate. Thus the operating frequency can be calculated with

$$f_{ce} = LN_{code}f_c \tag{2}$$

where L is the number of the tracked multipaths, N_{code} is the maximum number of parallel code channels, and f_c is the chip rate. Thus assuming values $f_c = 3.84$ MHz, $L = 4$, and $N_{code} = 3$ results in a 46.08 MHz operating frequency.

B. Sample Buffer Size

The size of the sample buffer can be calculated with the following formula:

$$N_{buf} = \left\lceil \frac{T_c + T_{pre} + T_{post}}{T_c} \right\rceil 2RN_{bits} \tag{3}$$

where N_{buf} is the size of the sample buffer in bits, T_c is the chip duration (write window), T_{pre} is the pre-window duration, T_{post} is the post-window duration, R is the oversampling ratio, and N_{bits} is the word length for both I and Q samples. T_{post} is determined by the maximum delay spread supported by the FlexRake. The typical delay spreads are 1-2 μs in urban areas whereas delay spreads of over 20 μs can be expected in mountainous areas [1]. Examples of sample buffer sizes for different total window sizes and sample word lengths are listed in Table 1. With 8-bit I/Q samples and a 33.33 μs window, a 1 kB sample buffer would be required which is feasible considering hardware implementations.

V. FlexRake Hardware Model

The simulation of the FlexRake was done using SystemC modeling environment [6]. SystemC is fundamentally a C/C++ class library and simulation kernel which can be used to create cycle-accurate models of hardware architectures, software algorithms and system interfaces. The SystemC libraries provide necessary constructs to model fixed-point data arithmetic, hardware timing, concurrency, and reactive behavior with the C/C++ language.

The FlexRake hardware model is depicted in Fig. 4. The functional model was divided into five SystemC modules: Stream Buffer, Correlator Engine, control unit, sample source, and symbol dump sink. A Matlab script was used to create the

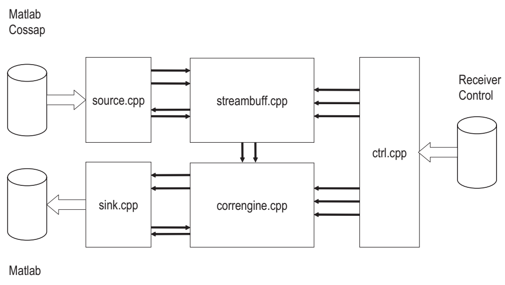


Fig. 4: Structure of the FlexRake hardware model.

baseband I/Q sample streams. The script produces a random binary sequence for transmission and then carries out spreading, scrambling, frame generation, and baseband signal generation. The multipath channel profile used by the Matlab channel model is static, i.e. the multipath delays do not change with time. Alternatively, a Cossap channel model can be used for creating the baseband input stream.

The interaction between the FlexRake receiver and a programmable DSP processor was simulated with a processor control file. This control file is composed of simulation instructions that specify a control register write, register read, or idle cycles. The control unit model decodes this syntax and performs cycle-accurate control updates accordingly. The symbol dumps are stored into an output file that can be further processed with a Matlab script. This script carries out algorithms for channel coefficient estimation, channel correction, maximal ratio combining, and symbol detection. In addition, raw channel bit error rate (BER) can be computed to evaluate different simulation scenarios.

The initial FlexRake receiver model was designed for a 3.84 Mcps chip rate and 15.36 MHz sampling rate (4x oversampling). The sample buffer size was 512 I/Q samples which was divided into a pre-window and post-window of lengths 2.07 μs and 31 μs, respectively. Tentatively, 10 bits for both I and Q samples was employed thus resulting in a buffer size of 1280 bytes. The number of Offset Address Registers was four and the number of integration registers was 12. This configuration enables concurrent reception of three parallel code channels.

A number of simulations with different downlink transmission scenarios were carried out and the correct operation of the FlexRake receiver was successfully verified. Numerical results are not in the scope of this paper.

VI. conclusions

A novel Rake receiver architecture was presented in this paper. The proposed FlexRake receiver stores the baseband I/Q sample stream in a circular buffer and sequentially performs correlations with the tracked multipath components. The multipath components are read from the buffer using offset addressing where the offset values correspond to the delays of the individual multipath components. Multipath I/Q samples are despread sequentially with a single complex correlator with codes that are exactly in the same phase. With respect to receivers based on Rake finger banks, the main benefits of the FlexRake receiver architecture are flexible multipath allocation, symbol synchronous operation, and straightforward receiver control.

References

[1] H. Holma and A. Toskala, *WCDMA for UMTS*, John Wiley & Sons, Ltd., New York, U.S.A., 2000.

[2] T. Ojanperä and R. Prasad, *Wideband CDMA for Third Generation Mobile Communications*, Artech House, Boston, MA, U.S.A., 1998.

[3] M. Kuulusa and J. Nurmi, “Baseband implementation aspects for W-CDMA mobile terminals,” in *Proc. Baiona Workshop on Emerging Technologies in Telecommunications*, Baiona, Spain, Sep. 1999, pp. 292–296.

[4] S.D. Lingwood, H. Kaufmann, and B. Haller, “ASIC implementation of direct-sequence spread-spectrum RAKE-receiver,” in *Proc. IEEE Vehicular Technology Conference*, Stockholm, Sweden, Jun. 1994, pp. 1326–1330.

[5] K. Easton J.K. Hinderling, T. Rueth and D. Eagleson, “Cdma mobile station modem ASIC,” *IEEE Journal of Solid-State Circuits*, vol. 28, no. 3, pp. 253–260, Mar. 1993.

[6] J. Gerlach and W. Rosenstiel, “System level design using the SystemC modelling platform,” in *Proc. Workshop on System Design Automation*, Rathen, Germany, Mar. 2000.

PUBLICATION 2

L. Harju, M. Kuulusa, and J. Nurmi, “A Flexible Implementation of A WCDMA Rake Receiver,” in *Proc. IEEE Workshop on Signal Processing Systems*, San Diego, CA, USA, Oct. 2002, pp. 147–160.

FLEXIBLE IMPLEMENTATION OF A WCDMA RAKE RECEIVER

Lasse Harju, Mika Kuulusa and Jari Nurmi

Tampere University of Technology
Institute of Digital and Computer Systems
P.O. BOX 553, FIN-33101 Tampere, FINLAND
E-mail: lasse.harju@tut.

ABSTRACT

Abstract - This paper presents an ASIC implementation of a WCDMA Rake receiver. The implementation is based on a FlexRake architecture that shares resources between multipath components and uses parallelism for multiple code channels. This approach facilitates the multipath allocation and improves the receiver modularity. The architecture was implemented using register-transfer-level VHDL description and logic synthesis with standard cells. Synthesis for 0.35 μm technology resulted in 0.894 mm^2 area and 3.63 mW power consumption at 2.7 V.

1. INTRODUCTION

Third generation communications systems are based on Wideband CDMA air-interface that employs direct-sequence spread spectrum with 3.84 Mcps chip rate, and QPSK modulation for obtaining a peak data rate of 2.3 Mbps. Variable user data rates are obtained by changing the user spreading factor or by employing multicode transmission [1]. These methods provide the bandwidth-on-demand needed for service e xibility, but at the same time introduce modularity issues that are a major design challenge concerning the receiver architecture.

Rake receivers are used in CDMA systems for obtaining multipath diversity which is one of the most important capacity improving features of CDMA systems. Typically this involves a number of Rake ngers, each receiving a multipath signal, and a Maximal Ratio Combiner for combining the outputs of each nger . Even though the functionality of a Rake receiver is fairly simple, its implementation is of paramount importance when considering the overall receiver e xibility.

In this paper we present a WCDMA Rake receiver implementation based on the FlexRake architecture presented in [2]. The proposed architecture provides improvements to the shortcomings of the traditional Rake architecture concerning modularity issues like multicode reception. This paper is organized as follows. First the Rake functionality in general is introduced shortly, followed by a comparison between a conventional and the FlexRake architecture. In the next section, two implementation versions of the proposed architecture are studied and their advantages are highlighted. Next, simulation and synthesis results are presented, and nally , conclusions are drawn.

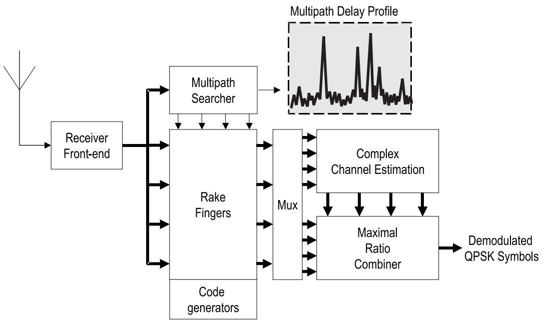


Fig. 1. Functional block diagram of a Rake receiver.

2. RAKE RECEIVERS

Because of multipath propagation, several copies of the transmitted signal with different delays, attenuation, and phases are picked up by the receiver antenna. A Rake receiver isolates the strongest multipath components from the received signal and combines them coherently. A functional block diagram of a Rake receiver is depicted in Fig. 1.

Each multipath component is despread by correlating the received signal with the spreading code, and integrating over a period corresponding to the spreading factor. After the despreading, maximal ratio combining (MRC) is applied to the symbol dumps from the ngers. Each symbol is weighted in proportion to its amplitude and the phases of the symbols are aligned after which they are added [3]. The phase and amplitude estimations of the multipath com-

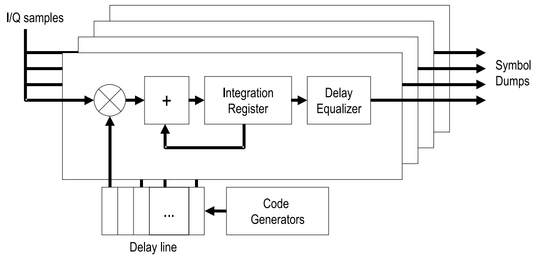


Fig. 2. Conventional Rake architecture.

ponents are acquired by the Complex Channel Estimator. MRC results in a total signal-to-noise ratio (SNR) that is equal to the sum of the SNRs of the multipath signals [3].

An important part of the Rake functionality is the multipath searcher that detects the strongest multipath components and determines the relative delays between them. This is done by correlating the received signal with known pilot bits, and detecting the strongest peaks from the correlator output. A peak's magnitude is proportional to the gain of the multipath and the distance of the peaks gives an estimate of the delays [4].

The code generators have also very important role in the Rake functionality. In WCDMA, two types of codes are used: Orthogonal Variable Spreading Factor (OVSF) codes for spreading and Gold codes for scrambling [1, 5]. The use of OVSF codes allows the spreading factor to be changed while maintaining the orthogonality between the codes. The complex Gold code is used on top of the spread signal in order to randomize it, and thus, to improve its autocorrelation properties. In the receiver, one Gold generator and one OVSF generator for each code channel are needed.

2.1. Conventional Rake Architecture

Traditionally, the Rake functionality has been implemented with a set of Rake ngers that are used for parallel reception of the multipath signals. Each nger includes a correlator that performs the despreading. The strongest multipath components are assigned to the Rake ngers by changing the code phase fed into them according to the delay measurement of the multipath component. Because the relative delays between the multipaths are potentially much longer than the symbol integration period, a delay equalizer is needed in each nger to compensate the time differences between the completed integrations.

Although this kind of architecture has been commonly employed, it features problems concerning the receiver e x-ibility. The most critical of these is the fact that adding e x-ibility to this kind of architecture requires extensive amount of additional hardware. The situation is most observable in

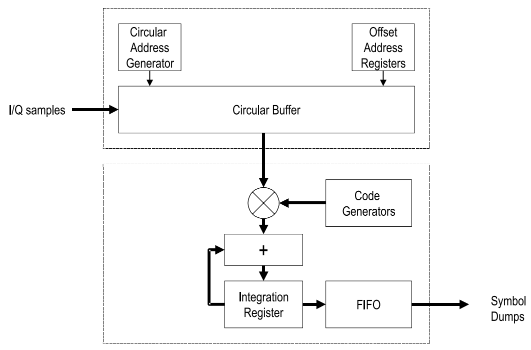


Fig. 3. The FlexRake architecture consisting of Stream Buffer (upper block) and Correlator Engine (lower block).

the case of multicode reception because one correlator per code channel per nger is required. The peak data rate 2.3 Mbps in the downlink can be achieved with three parallel code channels and spreading factor $SF = 4$ [1]. With four ngers, this equals to a total of $4 \times 3 = 12$ correlators.

Another problem is associated with the nger allocation. In a situation where the delay of the rst multipath shortens, the nger assigned to that multipath would have to rewind the sample stream in order to track this change. This is not possible, and consequently, the symbol cannot be despread and used for combining which results in decreased diversity.

The assignment of the multipath components to the Rake ngers can be done, for example, by selecting the right code phase from a delay line as depicted in Fig. 2. However, the needed code phase may fall out from the delay line range when the channel is changing rapidly, and consequently, the code phases of the code generators need to be adjusted. Depending on their implementation, this can take several clock cycles and force the reception to be suspended.

2.2. FlexRake Architecture

The FlexRake architecture is designed to avoid the shortcomings of the conventional Rake architecture [2]. A block diagram of the FlexRake is depicted in Fig. 3. It is composed of two parts: Stream Buffer and Correlator Engine. In the Stream Buffer, the sample stream coming from the receiver front-end is stored in a circular buffer which is long enough to hold the I/Q samples within the multipath searchers tracking window. The multipath components are accessed from the buffer with a special addressing method, and read to the Correlator Engine that performs the despreading of the multipath signals sequentially, i.e. one correlator is time-multiplexed between the multipath components.

The addressing method used in the Stream Buffer is controlled by the multipath searcher. Instead of using the de-

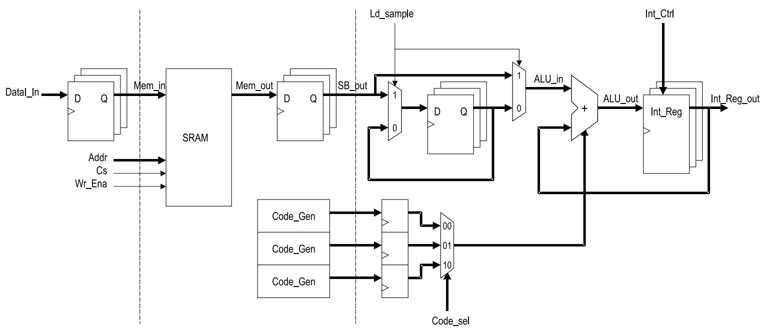


Fig. 4. FlexRake datapath with a single correlator.

lay estimates for adjusting the phases of the codes, they are stored in offset address registers and used in the address generation for the circular buffer. By adding an offset address to a cursor address generated by the circular address generator, individual multipath components can be allocated from the buffer. This method improves the e xibility of the multipath allocation. The delay window of the circular buffer can be selected so that multipath components can be tracked even if they move across the beginning of the tracking window, i.e. negative offset addresses can be used [2].

Because the operation of the Correlator Engine is time-multiplexed between the multipath components, its operation frequency has to be high enough that all correlations can be performed within a processing cycle de ned be the chip rate. In addition, in the case of multicode reception a time slot has to be reserved for each code channel. With four multipath components and three code channels a total of $4 \times 3 = 12$ correlations are done within a processing cycle. This is a requirement that dictates the selection of the clock frequency of the implementation. With the 3.84 Mcps chip rate and 12 correlations per chip period, this equals in a 47 MHz clock frequency.

The completed symbol dumps are written into a small FIFO which is used as a temporary storing buffer before the symbol dumps are read to the maximal ratio combining.

3. FLEXRAKE HARDWARE IMPLEMENTATION

The FlexRake was implemented using register-transfer-level (RTL) VHDL description and logic synthesis with standard cells. The implementation was divided into three design blocks: Stream Buffer, Correlator Engine, and Control. Complex channel estimation and the maximal ratio combining were left out of the examinations, because the modi cations introduced by the FlexRake apply only to the implementation of the Rake ngers. The functionality, and thus, the interface to the other blocks of the receiver are unaffected by these changes.

The parameters of the implementation were chosen so that the FlexRake supports the reception of four multipath components and three parallel code channels. Four times oversampling was chosen with 8-bit samples. The length of the circular buffer was chosen so that the total tracking window for the multipaths is 33 μs long, which corresponds to 128 chips. With the four times oversampling and 8-bit samples this results in 1 kbyte (8×1024) of SRAM memory. Note that the memory size is doubled because storage is needed for both I and Q samples. One-port memory is utilized due to its smaller area with respect to multiport memories.

3.1. Single-Correlator FlexRake

The data path of the FlexRake is depicted in Fig. 4. The computation is divided into three pipeline stages: address generation, memory access, and correlation. The stages are separated with a dashed line in the gure.

The address computation unit is not shown in the picture, but it is composed of a counter that produces the circular cursor address, and an adder that adds one of the offset addresses to the cursor value to form a read address. The data path, in all its simplicity, features a few interesting details. The multicode operation is carried out with a simple multiplexer/register structure. In normal operation, I/Q samples are read from the SRAM and fed directly to the ALU which performs the correlation. If multiple correlations are to be made with the same I/Q sample, it is stored in a register and fed again to the ALU, which now performs the correlation with a different code. The correlations are done in sequences consisting of up to four multipath components, and up to three correlations with each. After each sequence the code generators are incremented and the next sequence is started.

The ALU performs the complex multiplication between the I/Q samples and the code generator output, and adds the result to the partial integration result. Because the spreading codes are always sequences of 1's and -1's only (mapped to logic '0' and '1', respectively), the complex multiplication in the correlations is simpli ed to a simple sign change operation. This can be derived from the following formula:

$$(S_I+jS_Q)(C_I-jC_Q) = (S_IC_I+S_QC_Q)+j(S_QC_I-S_IC_Q)$$
$$= \begin{cases} S_I+S_Q+j(S_Q-S_I) & C_I=0,\ C_Q=0 \\ S_I-S_Q+j(S_Q+S_I) & C_I=0,\ C_Q=1 \\ -(S_I-S_Q)-j(S_Q+S_I) & C_I=1,\ C_Q=0 \\ -(S_I+S_Q)-j(S_Q-S_I) & C_I=1,\ C_Q=1 \end{cases}$$

where S_I and S_Q represent the real and imaginary parts of the input samples, and C_I and C_Q the corresponding code values. The code is the combined OVSF/Gold code obtained by $C_I = C_{Ovsf} \oplus Re\{C_{Gold}\}$ and $C_Q = C_{Ovsf} \oplus Im\{C_{Gold}\}$.

Consequently, the whole correlation can be implemented with a two stage adder/subtractor structure. The rst stage either adds or subtracts the real and imaginary parts depending on the code outputs, as derived in the equation above. The second stage then adds or subtracts the result from the partial integration registers. The adder/subtractor modules were implemented with carry-look-ahead topology.

The code generators are a design challenge themselves, particularly the OVSF generators. The properties of the OVSF codes and the rules for generating them are described by the 3GPP speci cations [6] but they take no position on their implementation in any way. The total number of the available OVSF codes is too large to consider an implementation based on look-up tables, and hence, another type of implementation is necessary. The OVSF generators were implemented with combination of a counter and a logic network which is controlled by the spreading code number. The basic idea is that the counter provides the code index, and the logic network produces the right code output from the counter value. The implementation of the Gold code generator is much simpler. It is generated as a combination of two 18-bit m-sequences with generator polynomials $1 + X^7 + X^{18}$ and $1 + X^5 + X^7 + X^{10} + X^{18}$ [6].

The integration registers are a signi cant portion of the required hardware resources. One integration register is needed for each code channel and for each multipath component for both I- and Q-branches which equals in 24 integration registers that have to be double-length to avoiding over o ws with long integrations ($SF = 512$). In addition, registers are needed for the symbol dump FIFO (not shown in Fig. 4).

The control block is of great importance in the FlexRake functionality, and the object of most of the designing effort. It was implemented as a Moore type nite state machine. In addition to the controlling signals shown in Fig. 4, it generates the control signals to the FIFO and to the receiver front-end interface. Its functionality was divided into two distinctive state machines; one controlling the memory accesses and the other controlling the correlations.

3.2. Multi-Correlator FlexRake

After the initial implementation, the FlexRake architecture was enhanced with added parallelism. This was done mainly because of the high clock frequency requirement caused by the number of sequential correlations in multicode reception. The solution was to add parallelism to the correlator engine. However, this was done in a totally different way than in the traditional Rake architecture. Instead of using the parallelism for different multipaths, it is used for different code channels.

The modi ed FlexRake data path is illustrated in Fig. 5. A dedicated correlator with a dedicated code generator is used for each code channel. Changes to the initial datapath architecture in Fig. 4 is the absence of the multiplexer/register structure and the two additional ALUs. The advantages of this approach can be appreciated when a typical receiving situation is considered. It can be assumed that when ever multipath propagation exists it is exploited by diversity reception, i.e. as many Rake ngers are utilized as possible. Parallel code channels, on the other hand, are used less frequently compared to the multipath reception. Thus, it is a better approach to optimize the architecture for the normal multipath situation, and to minimize the impact of multicode capability on the normal operation.

The clock frequency of the initial architecture had to be high enough to allow 12 sequential correlations during one chip period. Only four of these are actually needed if multiple code channels are not being used. By introducing the parallel correlators, the number of sequential correlations is always four. These are all used when four multipath components are received, which can be considered as the common case. As a result, the required clock frequency dropped from 47 MHz to 16 MHz. It is important to note that with this architecture, increasing the data rate by adding parallel code channels has minimum effect on the receiver operation.

The FIFO in the Correlator Engine was replaced with a matrix type of parallel-in/serial-out register bank. The symbol dumps from the three correlators are written in parallel into the matrix rows that have columns for four multipath components. The symbol dumps are read from the matrix row-by-row in serial form.

The amount of additional hardware introduced by the added parallelism, includes only the two ALUs, because dedicated code generators and integration registers are needed anyway. At the same time, the multiplexer/register structure needed for the multicode operation in the initial architecture

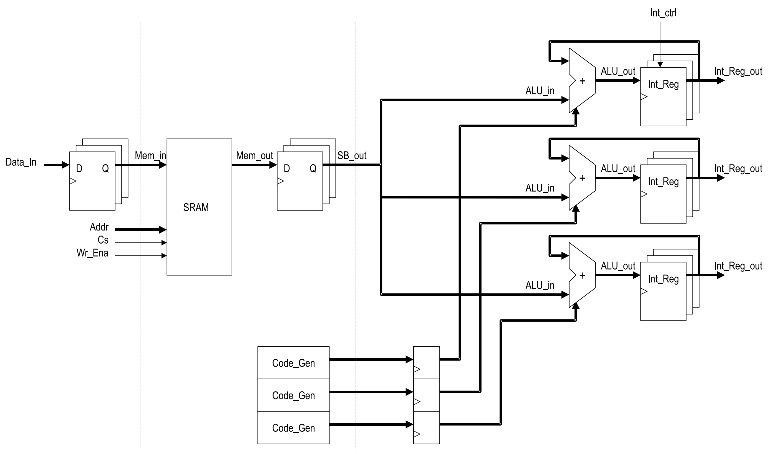


Fig. 5. FlexRake datapath with three parallel correlators.

can be left out, and most important, the control block is simpli ed greatly.

4. SIMULATION AND SYNTHESIS RESULTS

Both implementations were simulated in RTL and gate-level. The simulations were made using a VHDL testbench with multicode setup and several spreading factors. In addition to the actual FlexRake top level entity, the testbench included a symbol source, a symbol dump sink, a behavioral memory block, and a receiver control block that performed the overall con guring and controlling of the FlexRake.

The input stimulus used in the simulations was created with a Matlab model of the WCDMA transmitter and multipath channel. The output of the test-bench was veri ed by another Matlab model that compared the original data with the demodulated data. The results were not analyzed in terms of BER versus E_b/N_0 because the mathematic functionality of the FlexRake is not any different from the conventional Rake architecture. It was simply veri ed that the transmitted symbols were demodulated correctly.

After the RTL simulations were completed, a gate level model was created from the synthesis tool and formal veri cation between the two was carried out. The gate-level model was then used with the testbench and the switching activity from the simulations was back annotated to the power estimation tool. The synthesis results and power estimations of the two versions of the FlexRake are listed in Table 1. It can be seen that the only considerable difference between the two versions is the power consumption. Decreasing the clock frequency had an anticipated result on the power consumption, but at the same time adding the two

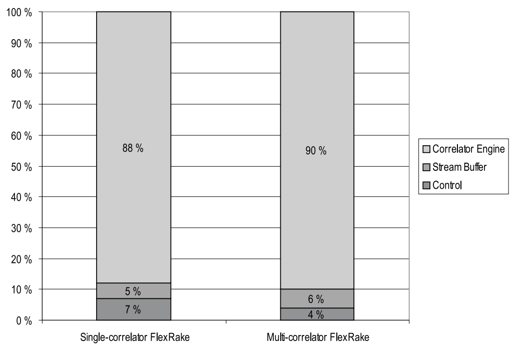


Fig. 6. The distribution of the area between different parts of the implementation.

ALUs increased the area only by a fraction. The memory needed for the circular buffer was not included in the area estimations. However, based on a reference implementation that is done for a similar 0.35 μm technology, it can be estimated that the memory consumes 0.599 mm^2 of area [7].

The distribution of the area between different parts of the architecture is illustrated in Fig. 6. The Correlator Engine clearly dominates the total area in both version. This is partly explained by the portion of the integration registers in the Correlator Engine which was about 25% for both versions. The portion of the ALUs was 7.3% for the single-correlator version and 20% for the multi-correlator one. This is a signi cant increase but it was compensated

	Single-correlator FlexRake		Multi-correlator FlexRake	
Processing technology	0.35 μm	0.18 μm	0.35 μm	0.18 μm
Supply voltage	2.7 V	1.8 V	2.7 V	1.8 V
Clock frequency	47 MHz	47 MHz	16 MHz	16 MHz
Total cell area	0.790 mm^2	0.168 mm^2	0.894 mm^2	0.190 mm^2
Power consumption	12.89 mW	1.55 mW	3.63 mW	0.44 mW

Table 1. Synthesis results and power estimations of the FlexRake. Results for the 0.18 μm technology are scaled from the simulated results according to vendor information.

by the simpli ed control block and the absence of the multiplexer/register-structure. In the whole design the portion of register elements of the total area was 50% for the single-correlator FlexRake, and 42% for the multi-correlator FlexRake.

5. CONCLUSIONS

A Rake receiver implementation was presented in this paper. Instead of the traditional Rake architecture based on parallel ngers, the proposed implementation is based on an architecture that shares resources between the multipath components, and uses parallelism for multiple code channels. The incoming sample stream is stored in a circular buffer, from where the multipath components are accessed with a special addressing method that is controlled by the multipath searcher. This method facilitates the e xibility of the multipath operation and improves modularity of the receiver. Simulation and synthesis results of the architecture were presented together with power estimates for two versions of the implementation. The results are favorable to a parallel implementation that allows the use of a lower clock frequency and at the same time, simpli es the control.

6. REFERENCES

[1] H. Holma and A. Toskala, *WCDMA for UMTS*, John Wiley & Sons, Ltd., West Sussex, England, 2001, Revised Edition.

[2] L. Harju, M. Kuulusa, and J. Nurmi, “A Flexible Rake Receiver Architecture for WCDMA Mobile Terminals,” in *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC’01)*, Tao Yuan, Taiwan, Mar. 2001, pp. 9–12.

[3] J. S. Lee and L. E. Miller, *CDMA Systems Engineering Handbook*, Artech House, Boston, MA, USA, 1998.

[4] E. Bejjani, J-F. Bouquier, and B. de Cacqueray, “Adaptive Channel Delays Selection for WCDMA Mobile System,” in *IEEE Vehicular Technology Conference*, Amsterdam, Netherlands, Sep. 1999, vol. 1, pp. 203–207.

[5] E. H. Dinan and B. Jabbari, “Spreading codes for Direct Spread CDMA and Wideband CDMA Cellular Networks,” *IEEE Communications Magazine*, vol. 36, no. 9, pp. 48–54, Sep. 1998.

[6] 3GPP Technical Speci cation 25.213, “Spreading and Modulation (FDD),” 2000, Release 4.

[7] T. Solla, R. Makela, M. Liljeroos, and O. Vainio, “Application-Speci c Filter Processor for Flexible Receivers,” in *Proc. 19th Norchip Conference*, Kista, Sweden, Nov. 2001, pp. 53–58.

PUBLICATION 3

L. Harju, M. Kuulusa, and J. Nurmi, “A Flexible Implementation of A WCDMA Rake Receiver,” in *The Journal of VLSI Signal Processing*, Springer Science, vol. 39, no 1–2, pp. 147–160, Apr. 2005.

PUBLICATION 4

L. Harju and J. Nurmi, “A Baseband Receiver Architecture for UMTS/WLAN Interworking Apllications,” in *Proc. IEEE International Symposium on Computers and Communications*, Alexandria, Egypt, Jun. 2004, vol. 2, pp. 678–685.

A Baseband Receiver Architecture for UMTS-WLAN Interworking Applications

Lasse Harju and Jari Nurmi
Tampere University of Technology, Institute of Digital and Computer Systems
P.O.BOX 553, 33101 Tampere, Finland
Tel: +358 3 3115 4365, Fax: +358 3 3115 3095, E-mail: lasse.harju@tut.fi

Abstract

This paper presents a programmable hardware platform for dual-mode WCDMA/OFDM receiver implementations. The platform is targeted for mobile terminals capable of operating in tight coupling UMTS-WLAN interworking systems. The proposed platform comprises a RISC core and three coprocessors that are used for the most intensive computation kernels. The receiver algorithms needed in WCDMA and OFDM receivers are overviewed and the needed computation resources are specified based on the analysis. The high-level architecture of the dual-mode receiver is also presented. A software development model is specified for the platform.

1. Introduction

The trend in wireless communications is towards systems where small-coverage high-bandwidth networks, such as wireless local area networks (WLAN), are employed as complementary networks to 3G systems, such as Universal Mobile Telecommunication System (UMTS). The reason for this trend is that WLANs could effectively be used in the hot-spot areas to increase the capacity of the cellular network. In such systems, subscribers could be able to access packet based services through both WLAN and UMTS networks. The effects of employing WLANs in the hot-spots are studied in [1]. The coverage, throughput, and capacity were shown to increase significantly.

Several proposals of how these networks could be tied together have been made [2], but no standards have been released yet. However, the interworking between UMTS and WLANs is under consideration within both 3rd Generation Partnership Program (3GPP) and European Telecommunications Standards Institute (ETSI) [3,4]. The focus of the published work has been on the network architectures. The fundamental requirement is, however, that the mobile terminal can operate in both networks. In this paper, we study the algorithms needed in UMTS and WLAN baseband receivers and present a programmable hardware platform for implementing these algorithms in mobile devices. The focus of this work is on the interworking between UMTS and orthogonal frequency division multiplexing (OFDM) based WLANs. The different radio technologies involved are listed in Table 1. A reduced instruction set computer (RISC) core is used as a central processing element in the proposed platform. Thus, high flexibility is provided to speed-up time-to-market and to ensure compatibility with other wireless standards using the same air-interfaces. Secondly, high computation power is provided by incorporating dedicated hardware for the implementation of the critical computation kernels found in wideband code division multiple access (WCDMA) and OFDM receiver algorithms.

Table 1. Different air interfaces involved in the interworking between UMTS and OFDM based WLANs.

	IEEE 802.11.a	HIPERLAN/2	UTRA (FDD)
Frequency band	5 GHz (ISM)	5 GHz (ISM)	uplink: 1920-1980 MHz downlink: 2110-2170 Mhz
Physical layer	OFDM	OFDM	WCDMA
Channel width	20 MHz	20 MHz	5 MHz
Modulation	BPSK, QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM	QPSK, (16-QAM)
Channel coding	Convolutional (1/2, 2/3, 3/4)	Convolutional (1/2, 2/3, 3/4)	Convolutional (1/2,1/3), Turbo (1/3)
Payload data throughput	6, 9, 12, 18, 36, 54 Mbps	6, 9, 12, 18, 36, 54 Mbps	up to 2 Mbps

The paper is organized as follows. In section 2, an interworking architecture between UMTS and WLAN is briefly introduced. In section 3, the different parts of the WCDMA/OFDM receiver are studied, followed by an overview of the needed receiver algorithms in section 4. Based on this analysis, the high-level receiver architecture

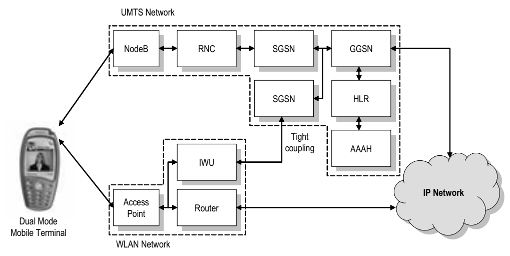


Figure 1. Tight coupling interworking architecture between UMTS and WLANs.

is specified in section 5, and the platform architecture is presented in section 6. Finally, conclusions are drawn.

2. Tight Coupling Interworking Scheme

ETSI has specified two interworking architectures between HIPERLAN/2 and 3G networks: loose coupling and tight coupling architectures [3]. The main difference between these two approaches is the point where the networks are connected in the network architecture. The focus of this paper is on the tight coupling scheme depicted in Fig. 1.

In the tight coupling interworking scheme, the WLAN network is employed as a radio access network complementary to the UMTS Terrestial Radio Access Network (UTRAN). A new user data interface between the networks is utilized that is equivalent to the Iu-interface connecting UTRAN to the Core Network (CN) in UMTS [5]. An interworking unit (IWU) is needed between the WLAN access points and the Serving GPRS Support Node (SGNS). The IWU has a similar function as a Radio Network Controller (RNC) in UTRAN. An interworking architecture based on the tight coupling scheme is proposed in [2].

The merit of the tight coupling scheme is that the existing methods for Quality of Service (QoS), mobility and security of the UMTS network can be reused. The potential drawback is, however, that the Gateway GPRS Support Node (GGSN) easily becomes the bottleneck of the architecture since all packet data traffic is routed through it.

3. Requirements for WCDMA/OFDM Receiver Architecture

In this section, the different parts of the WCDMA/OFDM receiver are studied. A key question concerning the dual-mode receiver implementation is how large part of the hardware resources can be shared between the two modes of the receiver. The different parts of the receiver, depicted in Fig. 2, are discussed in the following sections.

3.1. RF-Section

The RF characteristics for the two air-interfaces are very diverse. The bandwidth of the OFDM signal is 20 MHz centered at 5 GHz, whereas the bandwidth of the WCDMA signal is 5 MHz centered at 2 GHz. However it is possible to achieve high integration between the two receiver paths even in the RF front-end. A direct conversion RF front-end targeted for WCDMA and WLAN presented in [6] uses dedicated components for the antenna, channel selection filter, and LNA. A high bandwidth mixer and phase shift circuit are shared between the systems.

3.2. Analog-to-Digital Conversion

A shared analog-to-digital converter can be used for the systems, although the characteristics of the OFDM signal dictate the sampling rate and sample width selection. For WCDMA receiver, 4-6 bit resolution has been found adequate [7], but in OFDM based WLANs the needed dynamic range is much larger, and thus, 9-bits/ sample are needed [8]. In the case of WCDMA, the 3.84 M chip rate dictates the minimum sampling frequency, although interpolation or oversampling is required if better multipath resolution is needed. Oversampling is not that critical in OFDM systems, as they are less sensitive to timing errors. Therefore, a sampling rate of 40 Msps, two times the employed signal bandwidth, is adequate.

3.3. Digital Baseband

The digital baseband section comprises the synchronization, demodulation, channel estimation and channel equalization blocks, as depicted in Fig. 2. The algorithms used in these blocks are very similar for WCDMA and OFDM, although their throughput and accuracy requirements are very diverse. High level block diagram of the baseband sections of OFDM and WCDMA receivers are depicted in Fig. 3.

3.3.1. WCDMA Baseband. The Rake receiver is the key component of the WCDMA receiver. It comprises the multipath searcher, the Rake finger bank, channel estimator, and maximal ratio combiner. The demodulation is performed in the Rake fingers by correlating the received signal with a spreading code over a period corresponding to the spreading factor. The multipath

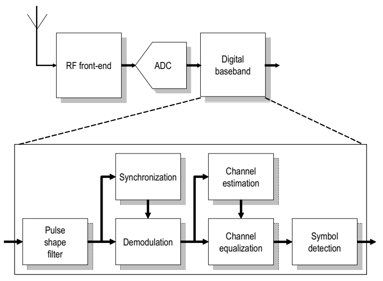


Figure 2. A block diagram of a generic receiver architecture.

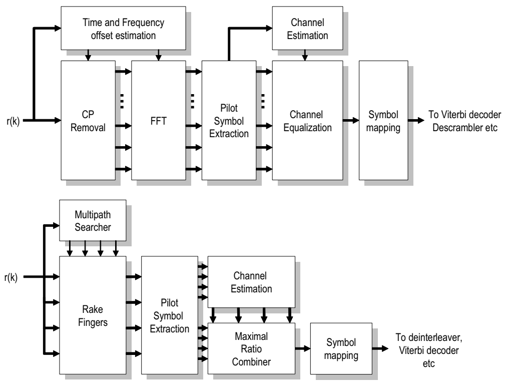


Figure 3. The digital baseband sections of OFDM (top) and WCDMA (bottom) receivers.

searcher detects the strongest multipath components and determines their relative delays. The start indices of the Rake finger correlations are determined according to this information. After the demodulation, maximal ratio combining (MRC) is applied to the symbol dumps from the fingers. In maximal ratio combining, the phases of the symbols are aligned and their amplitudes are weighted according to the complex tap coefficients acquired by the complex channel estimator. After the combining, decision of the transmitted symbol is made and the resulting bit stream is deinterleaved and decoded.

3.3.2. OFDM Baseband. In OFDM, demodulation is performed by applying FFT to the samples within the symbol window. The symbol boundaries are detected by the timing offset estimator and the cyclic prefix is removed before the FFT. The cyclic prefix is a copy of the OFDM symbols tail which is inserted in front of the symbol [9]. From the demodulated symbol stream, the pilot bits are extracted, and used to estimate the channel coefficients. After the channel equalization, the symbol mapping is performed. The bit stream is then sent to the outer receiver that performs deinterleaving, descrambling, and Viterbi decoding.

4. Receiver Algorithms

In this section, different types of algorithms needed in the OFDM and WCDMA baseband receivers are studied. The general model of the received and sampled baseband signal can be expressed as

$$r(k) = \sum_{i=1}^{L} e^{j2\pi\Delta f(k-\tau_i)} h_i(k) s(k-\tau_i) + n(k) \qquad (1)$$

where *s(k)* is the transmitted signal, $h_i(k)$ is the complex channel coefficient of the *i*-th multipath, τ_i is the delay of the *i*-th multipath (in integer multiple of samples), *L* is the length of the channel impulse response, *Δf(k)* is the frequency offset, and *n(k)* is additive white Gaussian noise (AWGN). In the case of OFDM, the expression simplifies greatly if quasistationary AWGN channel is assumed

$$r(k) = s(k-\theta)e^{j2\pi\Delta fk/N} + n(k) \qquad (2)$$

where *θ* is the timing offset caused by the channel, *Δ*f is the frequency offset common to all subcarriers, and *N* denotes the number of subcarrier.

4.1. Frequency Synchronization

Frequency mismatch between the oscillator frequencies of the transmitter and the receiver causes the constellation to rotate at constant speed. This frequency error can be estimated in both WCDMA and OFDM with a similar algorithm by exploiting the pilot sequences time-multiplexed with the transmitted data. The algorithm is based on the delay-and-correlate method where the received signal is correlated against a delayed version of itself. When two identical symbols in the sample stream separated by the length of the delay are correlated, the frequency shift can be estimated. The correlation output and the decision variable are computed as

$$c(n) = \sum_{k=n}^{n+L-1} r(k)r^*(k+D) \qquad (3)$$

$$p(n) = \frac{1}{2}\sum_{k=n}^{n+L-1} |r(k)|^2 + |r(k+D)|^2 \qquad (4)$$

$$m(n) = \frac{|c(n)|^2}{(p(n))^2} \qquad (5)$$

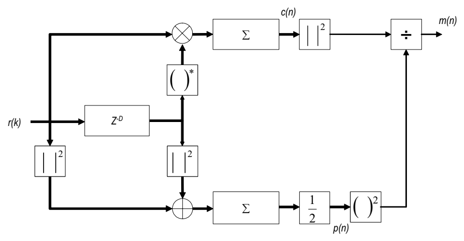


Figure 4. Signal flow representation of the delay-and-correlate algorithm.

where *r(k)* is the received and sampled baseband signal, *L* is the length of the correlation, and the delay *D* is the length of the two consecutive training symbol sequences used in the estimation. The value *p(n)* is the received signal power during the correlation period, and it is used to normalize the value of the correlation *c(n)*, so that it is not dependent on the received power. The decision is made based on the value of *m(n)*. The signal flow representation of the algorithm is shown in Fig. 4. In OFDM, the cyclic prefix can be used instead of a pilot sequence with same results [9]. The frequency offset estimate $\hat{\Delta f}$ is computed from the correlation output *c(n)* as

$$\hat{\Delta f} = -\frac{1}{2\pi D T_S} \angle c(n) \tag{6}$$

where $\angle$ denotes the argument of a complex number, *D* is the length of the delay, and T_S is the sampling interval. The arguments of *c(n)*, at the index that gives its maximum value of *m(n)* within observation window yields the frequency offset estimation. A maximum likelihood frequency offset estimation algorithm presented in [10] uses this approach. The result of the estimation can be used to construct an automatic frequency control (AFC) signal to the receiver front-end.

4.2. Timing Synchronization in OFDM Receivers

In OFDM the timing synchronization is used to determine the symbol boundaries in the sample stream. Because cyclic prefix is used, the timing estimate may vary within an interval bounded by the length of the cyclic prefix [9]. This makes OFDM systems less sensitive to timing errors.

4.2.1. Packet Detection. The first task of the synchronization procedure is to determine when a data packet has arrived. The detection can be made using the delay-and-correlate method, by exploiting the training symbols in the preamble of a data packet [9]. The value of *m(n)* in Eq. 5 is compared against a threshold value, and the detection is made when a correlation peak crosses this threshold.

4.2.2. Symbol Timing Estimation. Symbol timing estimation is needed to find the OFDM symbol boundaries. The estimation can also be done with the delay-and-correlate approach. In this case, training sequences are not needed, because the cyclic prefix can be used. The correlation *c(n)*, the weighting factor *p(n)*, and the decision variable *m(n)* are computed as in Eq. 3, Eq. 4, and Eq. 5. The index that gives the maximum value of *m(n)* inside the observation window yields the symbol timing estimate

$$\hat{\theta} = \arg max\{m(n)\} \tag{7}$$

A maximum likelihood symbol timing offset estimation algorithm presented in [10] uses this approach.

4.3. Timing Synchronization in WCDMA Receivers

In WCDMA, synchronization is needed for slot and frame timing, as well as to determine the multipath profile of the channel.

4.3.1. Cell Search. Cell search is divided into three steps: slot timing synchronization, frame timing synchronization, and scrambling code identification. Each step is performed by correlating the received signal against a different pilot sequence and detecting the correlation peaks as in the case of frequency offset estimation [5].

4.3.2. Multipath Estimation. The multipath estimation is often divided into two stages: acquisition and tracking. In the acquisition stage the received signal and the locally generated codes are synchronized. The acquisition can also be viewed as detecting the first arriving path of the received signal. In the tracking stage, the multipath searcher tracks the multipath taps inside a certain time window. The performance of different type of multipath delay estimators are compared in [11]. The best suited type for WCDMA Rake receivers is the feed-forward data-aided version.

The multipath estimation is done by correlating the received signal against a known pilot sequence and detecting peaks in the correlator output. This is very similar to the delay-and-correlate method illustrated in Fig. 4, only the delayed version of the received signal is substituted with the reference pilot sequence. The pilot symbols transmitted on the downlink Dedicated Physical Channel (DPCH) can be used for this purpose [5]. In order to minimize the effect of noise and the interference caused by other users, this correlation can be averaged non-coherently over a period of time. In principle, the first correlation peak determines the acquisition point and following peaks that are within a certain window determine the other multipath components. A peak's magnitude is proportional to the gain of the multipath, and the distance of a peak relative to the first arrival gives a measurement of the path's delay. The multipath delay estimation has to be performed at least at accuracy of one chip. Oversampling or interpolation is required if better multipath resolution is desired [11].

4.4. Channel Estimation

The channel estimator does not distinguish between the phase and amplitude fluctuations caused by the channel and those caused by synchronization errors. Thus, the channel estimator can act also as a fine-tuning synchronization. The channel estimation task is very similar in both WCDMA and OFDM, although the requirements are quite diverse. In WLAN systems it is commonly assumed that the channel is quasistationary, i.e., the channel conditions do not change during a data packet. Thus, the channel estimation in OFDM is done with the "single-shot" approach. This means that the estimation is made using the pilot symbols in the preamble of a data packet, and used for the entire packet. In WCDMA, the assumption of quasistationary channel does not apply because the signal bandwidth is larger than the coherence bandwidth of the channel, and the mobile speed is much higher than in typical WLAN scenarios. Therefore, methods for updating the channel estimates at symbol rate are needed. Furthermore, channel estimation has to be performed for each Rake finger output.

Assuming a non-frequency selective channel, the received k-th symbol after demodulation is denoted as

$$y(k) = h(k)x(k) + n(k) \tag{8}$$

where $h(k)$ is the complex channel coefficient corresponding to the k-th symbol, and $n(k)$ is additive white gaussian noise. If the transmitted symbols are known the estimate of the channel is computed as

$$\hat{h}(k) \approx \frac{y(k)}{x(k)} \tag{9}$$

In OFDM, the training symbols in the preamble of the packet are employed, and the channel estimates are computed as in Eq. 9 separately to all subcarriers. Because several identical training symbols are transmitted in the preamble, these can be averaged to diminish the effect of noise.

The time-multiplexed pilots on the downlink Dedicated Physical Control Channel (DPCCH) are used for the channel estimation in WCDMA [12]. Basically a WCDMA channel estimator is similar to the OFDM estimator, with the addition of a decision-directed (DD) stage that computes the estimates for the duration between the pilot symbols. A combined pilot-aided decision-directed (PADD) channel estimator is presented in [12]. The first stage of the estimator computes the preliminary channel estimates in each Rake finger by first averaging the pilots in one slot and applying Eq. 9. Then, the tentative estimates are computed for the data symbols using linear interpolation. After the MRC, the raw channel estimates are acquired by removing the data decision from the received symbols. Finally, a moving average filter is used to minimize the effects of noise. Dataflow representation of a feed-forward PADD channel estimation procedure is depicted in Fig. 5.

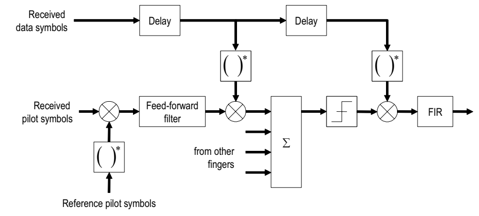


Figure 5. Dataflow representation of the PADD channel estimation.

5. Integrating the WCDMA and OFDM Receivers

From a purely functional point of view, it can be said that the receiver paths of the WCDMA and OFDM receivers can be merged to a large extent. All the receiver algorithms excluding the actual demodulation are very similar and based on the same computation kernels. However, without any of limitations on the type and volume of data traffic and mobility, the degree of integration of the receiver architectures is limited.

In order to support simultaneous reception of both systems, a separate sample stream is needed from the OFDM and WLAN front-ends, which rules out the possibility of integrating the RF section and ADCs. Furthermore, because the utilization of the hardware resources in the digital baseband is very high, sharing the functional units in the baseband section is limited to the possibility of time-multiplexing the hardware resources. This of course would lead to extensive clock frequencies.

Because of these reasons, a number of assumptions were made concerning the receiver front-end for the

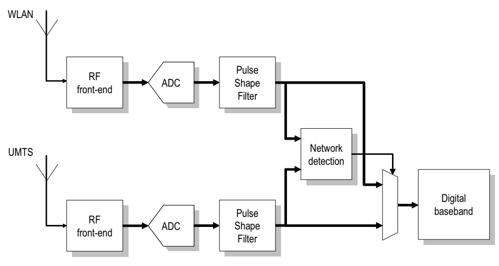


Figure 6. Receiver front-end of the dual-mode WCDMA/OFDM receiver.

purpose of this work. It is assumed that dedicated RF front-ends, ADCs, and pulse shape filters are used for both air-interfaces. The rest of the baseband section is shared between the systems, ruling out the possibility of carrying out simultaneous connection through both air-interfaces. A high-level block diagram of the receiver front-end is depicted in Fig. 6.

It is assumed that after power-up, the mobile terminal always connects first to UMTS and then starts monitoring possible WLAN access points (APs). The monitoring is performed by an additional network detection block that is parallel to the digital baseband, as illustrated in Fig. 6. When the mobile is in the WCDMA mode, this block monitors the availability of WLAN network by scanning for beacon frames transmitted by the APs periodically. When a beacon is detected, the mobile terminal switches to the WLAN mode and associates with the AP. Thereupon all control and user data is transmitted through the WLAN interface. While in the WLAN mode, the network detection block is used in turn for monitoring the Paging Channel (PCH) transmitted by the UMTS base station. This is equivalent to the Cell PCH state, one of the Radio Resource Control states in UMTS, in which the mobile terminal can only be reached through the PCH [5]. If a paging message is sent for the mobile, it terminates the WLAN connection and switches back to the UMTS mode. The WLAN connection has to be terminated also when a soft handover is to take place. This is because the mobile terminal cannot carry out the soft handover signalling with the base station while in the WLAN mode. Consequently, the mobility during the WLAN mode is limited.

6. A Programmable Hardware Platform for WCDMA/OFDM Receiver Implementations

In this section, a hardware platform suited for implementing the WCDMA and OFDM receiver functionalities described in previous sections is presented. A high-level block diagram of the dual-mode receiver was depicted in Fig. 6. The assumption was made that the mobile terminal does not have to carry on simultaneous connections to both networks, thus, only monitoring of the inactive network is required. A functional block diagram of the platform is depicted in Fig. 7.

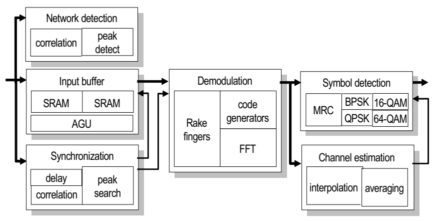


Figure 7. A functional block diagram of the WCDMA/OFDM receiver platform.

The sample stream is stored into a circular buffer which comprises two asynchronous SRAM memories and an address computation unit (AGU). The AGU computes the circular write addresses and the read addresses under the control of the synchronization.

The synchronization block executes the frequency offset estimation and timing estimation. It performs the complex valued correlations and a peak search algorithm. The output of the synchronization is used in the computation of read addresses for the input buffer. For example, the output of the OFDM symbol timing estimator is used to point to the first sample of an FFT input block. The network detection block is a simplified version of the pre-demodulation estimator.

The demodulation block reads the input samples from the buffer and performs the demodulation with the 64-point FFT or the Rake receiver. Code generators are needed for the spreading codes and downlink scrambling code.

The channel estimation block extracts the pilot symbols from the demodulated symbols and executes channel estimation. It uses averaging and interpolation to execute the feed-forward structures.

The symbol detection block performs channel equalization, the Maximal Ratio Combining (MRC), and converts the complex symbols into a bit stream.

6.1. The Platform Architecture

The above described functionality is mapped to a platform composed of a RISC core and a number of application specific coprocessors. The architecture is depicted in Fig. 8. The RISC core in question is the COFFEE RISC which is a parametric IP block that can be connected to four coprocessors through a coprocessor bus [13]. The coprocessor bus is illustrated in Fig. 9. Data

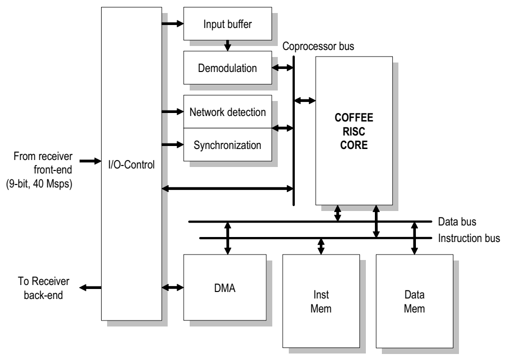


Figure 8. The WCDMA/OFDM platform architecture.

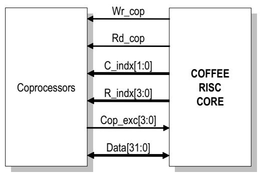


Figure 9. The coprocessor bus of the COFFEE RISC

between the coprocessors and the core is transferred through a common 32-bit *Data* signal and the direction of the transfer is selected with *Wr_cop* and *Rd_cop* signals. One of the four coprocessors is selected with the *C_indx* signal and the source or destination register is selected with the *R_indx* signal. Through these signals the core passes the input parameters to the coprocessors and initiates the computation kernels. Each coprocessor has a dedicated interrupt signal *Cop_exc*.

Three coprocessors are connected to the coprocessor bus: a synchronization coprocessor, a demodulation coprocessor, and an I/O coprocessor. The network detection block shares the same coprocessor interface with synchronization coprocessor. In addition to the core and the coprocessors, the platform comprises direct memory access (DMA), the input buffer, and on-chip memory for instructions and data.

The I/O coprocessor is used to control the connections to the receiver front-end and back-end. The input sample stream is fed through the I/O coprocessor into the network detection and synchronization coprocessors, as well as into the input buffer. The output symbol stream from the baseband is stored to the data memory and accessed via the I/O coprocessor and the DMA by the receiver back-end.

The synchronization coprocessor implements correlation-based time and frequency synchronization algorithms with a configurable FIR structure with complex valued coefficients. The input parameters for the synchronization coprocessor include the length of the correlation and selection between correlation with the delayed version of the received signal or correlation with a pilot sequence. The computation resources can be simplified as the delay-and-correlate based estimation of the OFDM receiver can be performed using only the sign bits of the samples [14]. Similarly, in WCDMA multipath searcher the correlations are performed between the samples and binary-valued pilot sequences which simplifies the complex multiplications to a simple combination of add and subtract operations. The network detection block is functionally almost identical to the synchronization coprocessor.

The Rake fingers and FFT functionality are mapped to the demodulation coprocessor. The input parameters for the demodulation coprocessor include the first read address used in the demodulation and the spreading factor, which determines the length of the despreading. The core also needs to initialize the code generators before initiating the kernel. The FFT is implemented with a pipelined single-path delay feedback (SDF) architecture [15]. The architecture is composed of 6 radix-2 butterflies (log_2N) and 63 delay elements. With this approach, the processing of the 64-point FFT takes 135 clock cycles ($2N+log_2N+1$). The Rake fingers are implemented as a bank of correlators. The Rake receiver alone needs resources for despreading up to 4 parallel channels and 4 multipath components. This means that a total of 16 parallel correlations have to be carried out. As in the case of the WCDMA multipath searcher, the correlation in the Rake fingers can be performed without complex multiplication. The functionality of the Rake receiver is based on the FlexRake architecture [16]. In addition to the Rake and FFT hardware, the demodulation coprocessor includes also the address generation unit (AGU) which is used to compute read addresses for the input buffer.

The rest of the receiver functionality is executed with the RISC instruction set. This includes channel estimation, equalization, and symbol mapping. Furthermore, several controlling tasks are executed with the RISC core.

6.2. The Software Development Approach

Application specific processors are known to be problematic for compilers, affecting negatively to the software development efficiency. With the proposed architecture the coprocessors are visible to the programmer through special function calls that are used to initiate the computation kernels in the coprocessors. The link from C-language is provided by means of a C-library that translates the function calls to assembly instructions that pass the source and destination operands and initiate the computation through the coprocessor bus.

The coprocessors can be configured to cause an interrupt whenever it completes a computation. The software written for the core executes the channel estimation and equalization tasks in the main program, and uses the interrupt service routines to initiate new kernels in the coprocessor and update the configurations.

7. Conclusions

A programmable hardware platform for implementing dual-mode WCDMA/OFDM receiver was presented. The platform is designed for mobile terminals capable of operating in a tight coupling interworking architecture between UMTS and WLANs. It was observed that dedicated receiver paths are needed for the RF front-end, ADC, and pulse shaping. Additional resources are needed also for monitoring the inactive network. The proposed platform comprises a RISC core and three coprocessors that are used for the most intensive computation kernels. The needed computation resources were specified based on a study on different OFDM and WCDMA receiver algorithms. It was shown that there are many similarities between the receiver algorithms of the two systems which can be utilized in the dual-mode receiver implementation. An efficient software development scheme was proposed for the architecture.

8. Acknowledgements

This work has been supported by the EU-project SoC-MOBINET (IST 2000-30094), the Nokia Foundation, the Finnish Cultural Foundation, Walter Ahlstrom Foundation, and the HPY Research Foundation.

9. References

[1] A. Doufex et al., “Hotspot Wireless LANs to Enhance the Performance of 3G and Beyond Cellular Networks”, *IEEE Communications Magazine*, Vol. 41, No. 7, pp. 58-66, July 2003.

[2] A. K. Salkintzis, “Interworking Between WLANs and Third Generation Cellular Data Networks”, in *Proc. IEEE Vehicular Technology Conference (VTC’03-Spring)*, Jeju, Korea, April 2003, pp. 1802-1806.

[3] ETSI Technical Report 101 957, *Requirements and Architectures for Interworking between HIPERLAN/2 and 3rd Generation Cellular Systems*, V1.1.1, Aug. 2001.

[4] 3GPP Technical Report 22.934, *Feasibility Study on 3GPP system to WLAN interworking*, Release 6, V6.2.0, Sep. 2003.

[5] H. Holma and A. Toskala, *WCDMA for UMTS*, John Wiley and Sons Ltd, West Sussex, England, 2001.

[6] M. Hotti et al., “A Direct Conversion RF Front-End for 2-GHz WCDMA and 5.8-GHz WLAN Applications”, in *Proc IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, Philadelphia, PA, USA, June 2003, pp. 45-48.

[7] L. Sumanen, K. Halonen, “A Single-Amplifier 6-bit CMOS Pipeline A/D Converter for WCDMA Receiver”, in *Proc. IEEE Symposium on Circuits and Systems (ISCAS’01)*, Sydney, Australia, May 2001, Vol. 1, pp. 584-587.

[8] T. H. Meng et al., “Design and Implementation of an All-CMOS 802.11a Wireless LAN Chipset”, *IEEE Communications Magazine*, Vol. 41, No. 8, pp. 160-168, Aug. 2003.

[9] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guid*e, Sams Publishing, Indianapolis, IN, USA, 2002.

[10] J.-J. van de Beek, M. Sandell, and P. O. Börjesson, “ML Estimation of Time and Frequency Offset in OFDM Systems”, *IEEE Transactions on Signal Procesing*, Vol. 45, No. 7, pp. 1800-1805, July 1997.

[11] E.-S. Lohan, *Multipath Delay Estimators for Fading Channels in CDMA Receivers and Mobile Positioning*, PhD Thesis, Tampere University of Technology, Tampere, Finland, Oct. 2003.

[12] A. Zhuang, E.-S.Lohan, and M. Renfors, “Comparison of Desicion-Directed and Pilot-Aided Algorithms for Complex Channel Tap Estimation in a Downlink WCDMA System”, In *Proc. IEEE Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC’00)*, London, UK, Sep, 2000, pp. 1121-1125.

[13] J. Kylliäinen, M. Kuulusa, and J. Nurmi, “COFFEE - A Core for Free”, in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2003, pp. 17-22.

[14] J.-J. van de Beek, M. Sandell, M. Isaksson, and P. O. Börjesson, “Low-Complex Frame Synchronization in OFDM Systems”, in *Proc. IEEE Conference on Universal Personal Communications*, Tokyo, Japan, Nov. 1995, pp. 982-986.

[15] T. Taskinen, *Hardware Implementation Architectures for Time-Frequency Transforms*, Master of Science Thesis, Tampere University of Technology, Tampere, Finland, 2002.

[16] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver”, in *Proc. IEEE Workshop on Signal Processing Systems (SIPS’02)*, San Diego, CA, USA, Oct 2002, pp. 177-182.

PUBLICATION 5

L. Harju and J. Nurmi, “A Programmable Baseband Receiver Platform for WCDMA/OFDM Mobile Terminals,” in *Proc. IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, Mar. 2005, vol. 1, pp. 33–38.

A Programmable Baseband Receiver Platform for WCDMA/OFDM Mobile Terminals

Lasse Harju and Jari Nurmi
Tampere University of Technology, Institute of Digital and Computer Systems
P.O.BOX 553, 33101 Tampere, Finland
E-mail: lasse.harju@tut.fi

Abstract—**A programmable system platform that enables software defined implementations of WCDMA and OFDM baseband receivers is presented. The design complexity of future wireless terminals and the shrinking time-to-market constraints are the motivators for adopting the platform-based design methodology. The presented hardware platform comprises a RISC core and three tightly coupled coprocessors that are used for the most intensive computation kernels. The application program interface of the platform is provided in the form of a library of special C-functions that are used to pass the input parameters to the coprocessors and initiate the execution. The SystemC-based simulation environment of the platform is described and the simulation results are given.**

Keywords—**Platform-based design methodology, software defined radio, WCDMA, OFDM, 3GPP-WLAN interworking**

I. INTRODUCTION

Many wireless devices today incorporate multiple short-range wireless technologies, such as Infrared, Bluetooth, and wireless local area network (WLAN), in addition to the second (2G) and third generation (3G) cellular wireless technologies. Because multiple radio technologies are being integrated into the same device, and possibly to the same chip, the design complexity of commercial wireless products has increased dramatically. Adversely, the lifespan of these products keeps on shrinking due to the emerging new features and services, resulting in very strict time-to-market constraints. The most effective way to meet these constraints and to cut the design costs is to exploit both hardware and software reuse, at highest possible level of abstraction [1]. In addition to the design time constraints, another big challenge are the increasing costs of mask sets for the latest silicon technologies [2]. Increasing the volume of the production is the only way of diminishing the impact of these costs, and consequently, only chips with guaranteed high-volume production are implemented on silicon. This requires that the chips are used in several products which is only possible if the designs are programmable. Efforts to exploit larger scale reuse and to find effective ways of incorporating programmable components in system-on-chip (SoC) architectures have led to the adoption of platform-based design methodology [2].

The multi-standard functionality of wireless systems will be taken one step further by the forthcoming interworking between universal mobile telecommunication system (UMTS) networks and WLANs [3]. The goal of this interworking is to form a heterogeneous wireless system that combines the strength of these two networks, i.e., high mobility through the cellular 2G and 3G networks, and high data throughput through WLANs in hotspots. The standardization of the interworking is currently ongoing in the 3rd generation partnership program (3GPP) [4]. The standards will enable system operators to offer WLAN access as an add-on to their general packet radio system (GPRS) services with common billing. In order to benefit from this interworking, users have to be equipped with terminals that are capable of operating in both networks. In the case of UMTS and WLANs this means that such diverse radio technologies as wideband code division multiple access (WCDMA) and orthogonal frequency division multiplexing (OFDM) have to be supported.

The multi-standard functionality of wireless communications will eventually evolve to something in accordance with the software defined radio (SDR) concept [5]. Currently, even the latest digital signal processors (DSP) employing very long instruction word (VLIW) and superscalar architectures, fabricated on the latest semiconductor technology, would not provide the needed processing power for a SDR implementation in its purest form [6]. However, it is the desirable flexibility of the SDR concept that leads the wireless world towards these sort of systems. On the other hand, it is the inevitable paradigm change in system design methodology that will eventually lead to a similar situation, where most of the multi-mode tranceiver functionality is implemented with software.

In this paper, we present a baseband receiver platform that enables a software-defined implementation of WCDMA and OFDM baseband receiver functionalities. The platform is composed of a reduced instruction set computer (RISC) core and a combination of coprocessors. The concept and the system-level issues of the platform were presented in [7].

In section II, a brief overview is given on the fundamentals of platform-based design methodology. In section III, the proposed hardware platform is presented and the coprocessor functionalities are described in detail. The interface used for programming the coprocessors is described in section IV, followed by the simulation results in section V. Conclusions are drawn in section VI.

II. Fundamentals of platform-based Design Methodology

The platform-based design methodology has evolved to serve two purposes: larger-scale reuse and separation of design concerns. Separation of design concerns means separating functionality from architecture and communication from computation. This helps to divide the design into more manageable parts, and facilitates the reuse. Although there exists many interpretation of what a platform is, it can always be divided into two parts: the hardware platform and the software platform, i.e., the application program interface (API). Together, the hardware platform and the API form a system platform [2].

A. Hardware Platform

The hardware platform can be regarded as a library of pre-designed components, optimized for a field of applications. The hardware platform is designed by first identifying the most important functions of an application field and examining their regularity and performance requirements. A library of parametrized processing elements is then constructed, that best supports these functions. This involves evaluation of several processing elements and communication primitives against the functional constraints of the application field. It is desirable to employ programmable components in the architecture, either DSPs, micro-processors, or reconfigurable logic, in order to enable the widest possible application space for the hardware platform.

B. Application Program Interface

The API is an abstraction of the hardware platform. Through the API, the target application is mapped on the hardware platform. As long as the API is kept unchanged, changes to the underlying architecture do not affect the application using the API. This enables effective software reuse. The implementation of the API may be as complicated as a real time operating system (RTOS), complemented with a set of device drivers, but a much simpler software layer may be adequate.

C. Implementation

Once a hardware platform and an API are available for a given application field, the system designer maps the functionality of the application onto the platform through the API. The hardware architecture is then derived from the platform by selecting the appropriate processing elements from the library and by exploring with the remaining parameters of the components, resulting in a platform instance. It is important to note that when the platform is oriented towards a field of applications, the design space is restricted, and thus time is saved in the implementation. Ideally, the component library that constitutes the hardware platform includes components at multiple abstraction levels. Choosing a platform instance closer to the actual implementation, results in the biggest time save, while starting from higher abstraction level results in greater level of freedom in the design space exploration.

III. A System Platform for WCDMA and OFDM Baseband Receiver Implementations

The proposed system platform is targeted for WCDMA and OFDM baseband receiver implementations. The functional profiling of the platform was carried out by simulating the physical layer procedures of WCDMA and OFDM air interfaces. The simulation were performed according to 3GPP and IEEE 802.11a specifications, respectively. Details of the identified functions and the system-level issues can be found in [7].

A functional block diagram of the combined OFDM and WCDMA baseband is depicted in Fig. 1. The main functional entities, which are identical in both air interfaces, include synchronization, demodulation, channel estimation, and symbol mapping. The hardware platform used for implementing these functions is composed of a RISC processor core and three coprocessors. The architecture is depicted in Fig. 2. The mapping of the functional blocks to the architecture is such that synchronization and demodulation functionalities are mapped to dedicated coprocessors, and the channel estimation and symbol mapping task are implemented with the instruction set architecture (ISA) of the RISC core. The core itself does not process the incoming sample stream, but only the demodulated symbols. In addition to the core and the coprocessors, the platform comprises direct memory access (DMA) and on-chip memory for instructions, data, and sample buffering. The sample buffer is long enough two store all the samples inside the expected channel delay spread. The longest supported buffer size is 1024 complex valued samples.

The RISC core in question is the COFFEE RISC which is a parametric IP block, developed at Tampere University of Technology [8]. A hardware description language implementation of the core, along with the software development tools, can be downloaded from the project website [9]. The COFFEE features a 6-stage pipeline with complete hazard detection and forwarding logic. The internal data word width is 32 bits, and the memory interface is of Harvard type. In addition to the typical implementation parameters, the core includes a configuration register that enables configuration of a number of features at run time.

The COFFEE core also features a coprocessor bus that can be used to connect the core to four coprocessors. Through this bus each coprocessor is seen as a register bank. The coprocessor bus and the internal structure of a coprocessor are depicted in Fig. 3. Data between the coprocessors and the core is transferred through a common 32-bit *Data* signal and the

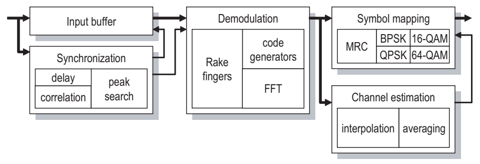


Figure 1. Functional block diagram of the combined WCDMA/OFDM baseband.

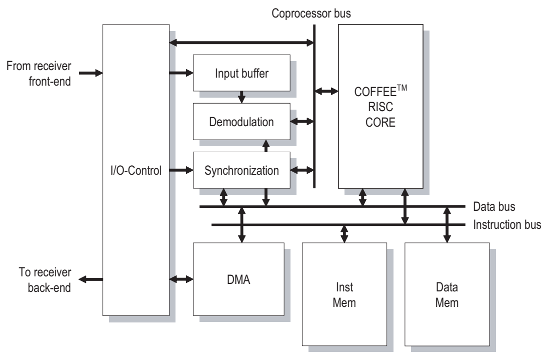


Figure 2. The architecture of the WCDMA/OFDM baseband receiver platform.

direction of the transfer is selected with *Wr_cop* and *Rd_cop* signals. The target coprocessors is selected with the *C_indx* signal and the source or destination register is selected with the *R_indx* signal. Each coprocessor has a dedicated interrupt signal *Cop_exc*. The demodulation and synchronization coprocessors are also connected to the data bus. This provides an alternative path for the less critical data traffic.

The computation kernels in the coprocessors are initiated by writing an instruction to an instruction first-in first-out (FIFO) buffer that is mapped to a specific register inside the register bank. The 32-bit instruction is composed of an operation code and a number of input parameters and control flags. The coprocessor reads the instruction from the instruction FIFO, decodes the operation code, stores the parameters to the appropriate registers, and starts executing the desired computation. When the execution completes, the return value of the function is stored to a special register in the register bank, and the possible output data is written in to a output FIFO.

A. Synchronization Coprocessor

The timing and frequency synchronization of WCDMA and OFDM receivers is performed with similar correlation based algorithms. The received signal is correlated against a pilot sequence and the timing synchronization is obtained by detecting the peaks from the correlation output. The frequency synchronization if obtained by examining the phase difference of sequential correlation outputs. Averaging is also commonly used to diminish the effect of noise [11,12].

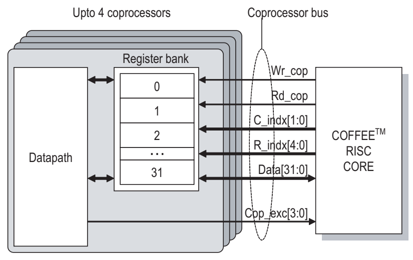


Figure 3. The coprocessor bus of the COFFEE RISC core.

The synchronization coprocessor implements the correlation based synchronization algorithms with a configurable finite impulse response (FIR) structure with complex valued coefficients. The structure can be configured to compute normal convolution algorithm, as well as delay-and-correlate algorithm. In the delay-and-correlate algorithm the received signal is correlated against a delayed version of itself. This can be used in many OFDM estimation algorithms [10].

Eight different sets of correlation coefficients can be stored into the coprocessor. These are distinguished with a coefficient handle which is passed as one of the input parameters of the correlation function. This way the coefficients can easily be switched between correlations as they are pre-stored to the coprocessor. The coefficients can be initialized by first passing the handle and the length of the coefficient vector to the coprocessor and then writing them sequentially to a specific coprocessor register. Alternatively the initialization can be performed by storing the coefficients into the data memory and by passing the address of the vector to the coprocessor.

The correlations are initiated by the core by passing the type of the correlation (normal/delay-and-correlate), the wanted coefficient handle (or the length of the delay in the delay-and-correlate algorithm), and the length of the correlation to the coprocessor. For purposes of WCDMA synchronization, a scrambling code generator is provided that can be set on. Thereupon, the correlation coefficients are automatically scrambled with the complex conjugate of the scrambling code.

After initiation, the correlation is executed in two steps. First, the correlation is computed until the first peak crossing the preset threshold occurs. The coprocessor then interrupts the core and starts the second step, if tracking parameters are specified and the tracking mode is switched on. In the second step, the correlations are computed over a given tracking window, and the output of the correlator can be averaged over a number of iterations. This is useful, e.g., when repeated copies of a pilot sequences are available for timing synchronization. The tracking parameters are set by passing the length of the tracking window, length of the gap between sequential windows, and the number of averaging iteration to the coprocessor. During the gap, the delay line of the FIR structure is shifted without any computations. This can be exploited when the correlation needs to be computed against a certain field of a transmission slot, and averaged over several sequential slots. During the last averaging iteration, the coprocessor detects all correlation peaks inside the tracking window crossing the threshold value. It stores their indices and normalized values to the output FIFO, and returns the number of detected peaks to the core. Thereupon, the core can read the peak values from the output FIFO and identify the strongest peaks and use the indices to compute the read addresses for the sample buffer. The peak values are given in angular form, which facilitates the frequency error estimation [12].

B. Demodulation Coprocessor

The biggest difference between WCDMA and OFDM basebands resides in the demodulation block. In WCDMA the demodulation is performed by a Rake receiver, whereas in OFDM it is performed with a 64-point fast Fourier transform (FFT).

A traditional Rake receiver is composed of parallel fingers that are each used to despread one multipath component of the received WCDMA signal. The despreading is performed by correlating the received signal against a spreading code over a period corresponding to the spreading factor. If multicode transmission is employed, a dedicated correlator is needed for each code channel in each finger.

The Rake functionality can be implemented effectively with the FlexRake architecture [13]. The essence of the FlexRake functionality is that in stead of using multiple Rake fingers, the despreading of the multipath components are performed with a single correlator engine. Effectively, hardware resources are shared between the multipath components, and parallelism is used for despreading multiple code channels. The relative delays of the multipath components are used for accessing the multipath components from the sample buffer that is large enough to store the samples within the multipath window.

The demodulation coprocessor includes also generators for orthogonal variable spreading factor (OVSF) and Gold codes. These are used in WCDMA for spreading/despreading and scrambling/descrambling, respectively. The OVSF code generators are initialized by passing the spreading factor, and the OVSF code number to the coprocessor. If multiple OVSF codes are initialized they are automatically assigned to the parallel correlators used in the despreading operation. Alternatively, the code generator output can be stored into the output FIFO, or to data memory, from where it can be read by the core. The Gold code generators are initialized by passing the primary and secondary scrambling code numbers to the generators. The Gold code does not have to be assigned explicitly to the correlators as it is automatically used in the despreading operation.

The despreading operation is initiated by passing the address of the first multipath component in the sample buffer, the number of despreading iterations, number of code channels, and the spreading factor (length of the correlation) to the coprocessor. The address of the first multipath component is used as a base address when computing the addresses of the other multipath components, which are allocated with offset addresses. The coprocessor reads the samples from the sample buffer, performs the correlation, and stores the symbol dump to the output FIFO. After a number of completed despreading iterations, specified by the input parameter, the coprocessor interrupts the core. Alternatively, the output can also be stored into the data memory, in which case the coprocessor returns the address of the output vector in the data memory.

In addition to the Rake functionality, the correlation resources can be used to execute any 16 parallel correlations. This feature is needed, e.g., in the second phase of the cell search procedure [14]. However, the correlations are not computed with a traditional convolution algorithm as the correlation windows do not overlap. Thus, the correlators in the demodulation coprocessors are not capable of computing matched filter type of correlations.

The computation resources are organized into four sequential groups of four parallel correlations. The four parallel correlations are always feeded with the same input samples, but the input to the sequential groups can be read from different addresses. These are defined by the offset addresses used also for allocating the multipath components in the despreading function. The correlation coefficients can be initialized similarly as in the synchronization coprocessor, and assigned to one of the 16 correlators. The Gold code generators can be switched on and off similarly as in the synchronization coprocessor.

OFDM demodulation is performed with a 64-point FFT that can be implemented efficiently with a pipelined single-path delay feedback (SDF) architecture [14]. The architecture is composed of 6 radix-2 butterflies and 63 delay elements. The FFT computation is initiated by passing the address of the FFT window, and the length of the FFT to the coprocessor. The twiddle factors are automatically initialized when the FFT is initiated for the first time or when the length of the FFT changes. The coprocessor generates an interrupt when the first output is written into the FIFO. Similarly as in the WCDMA demodulation, the FFT output can also be written into the data memory. The hardware architecture of the demodulation coprocessor is designed so that the correlations of the Rake fingers utilize same hardware resources as the butterflies of the FFT.

C. I/O Coprocessor

The I/O coprocessor is used to control the connections to the receiver front-end and back-end. The input sample stream is fed through the I/O coprocessor into the synchronization coprocessors and into the sample buffer. The core can turn the feed on and off. The I/O coprocessor also acts as a link between the receiver baseband and the receiver back-end. The output

TABLE I. COPROCESSOR FUNCTION CALLS

Function	Target coprocessor	Description	Input Parameters	Return Value
set_thr	*Sync*	Sets the threshold used for detecting the correlation peaks	threshold value (32 bits)	-
init_sync_corr_coef	*Sync*	Initializes correlation coefficients of the synchronization coprocessor	coefficient handle (3 bits), length (8 bits), source select (1 bit), memory address (14 bits)	-
set_track_param	*Sync*	Sets the tracking parameters for averaging correlations	window length (11 bits), gap length (11 bits), number of iterations (4 bits)	-
corr_x_thr	*Sync*	Starts the correlation. Writes the peak indices and the peak values in angular form to the output FIFO.	coefficient handle (3 bits), type select (1 bit), length of delay (8 bits), length of correlation (11 bits), Gold code flag (1 bit), tracking flag (1 bit)	Number of detected peaks
init_gold	*Sync/Demod*	Initializes the Gold code generator in both synchronization and demodulation coprocessors	primary gold code number (9 bits), secondary gold code number (4 bits)	-
init_ovsf	*Demod*	Initializes one of the OVSF code generators. Writes the code sequence to the coefficient memory or to the output FIFO.	spreading factor index (3 bits), OVSF code number (9 bits), output selection flag (1 bit)	The address of the code sequence in the data memory
init_demod_corr_coef	*Demod*	Initializes correlation coefficients of the demodulation coprocessor	coefficient handle (4 bits), length (8 bits), source select (1 bit), memory address (14 bits)	-
assign_corr_resrc	*Demod*	Assigns a specific correlation handle to one of the 16 available correlators	correlator number (4 bits), coefficient handle (4 bits)	-
init_multipaths	*Demod*	Sets the offset addresses of the detected multipath components	three offset addresses (10 bits)	-
despread	*Demod*	Starts the despreading, or any other correlation on the demodulation coprocessor. Writes the correlated symbols to the output FIFO or to data memory.	start address (10 bits), number of iterations (11 bits), number of parallel correlations (4 bits), Gold-code flag (1 bit), output selection flag (1 bit)	The address of the symbol dump vector in the data memory
fft	*Demod*	Starts the FFT. Writes the FFT output to the output FIFO, or to data memory.	start address (10 bits), FFT length (10 bits), output selection flag (1 bit)	The address of the FFT output vector in the data memory
rec_onoff	*I/O*	Turns on the feed to the sample buffer on and off	rec on/off flag (1 bit)	-
sync_onoff	*I/O*	Turns on the feed to the Sync coprocessor on and off	sync on/off flag (1 bit)	-
set_win	*I/O*	Sets the start address of the multipath window in the sample buffer	start address (11 bits)	-
wr_cop	*all*	Writes one 32-bit word to a register in the coprocessor register bank	coprocessor select (2 bits), register select (5 bits), data (32-bit)	-
rd_cop	*all*	Reads one 32-bit word from a regsiter in the coprocessor register bank	coprocessor select (2 bits), register select (5 bits)	32-bit data word

symbol stream from the baseband is stored to the data memory and accessed via the I/O coprocessor and the DMA.

IV. THE COPROCESSOR FUNCTION CALLS

The software layer implementing the API is provided in the form of special function calls that are exclusively used for controlling the coprocessors. These function calls are translated into sequences of assembly instructions that compose the 32-bit coprocessor instruction from the operation code of the function and the input parameters, and write the instruction to the instruction FIFO of the desired coprocessor. The link from C-language is provided by means of a C-library which effectively constitutes the API for the platform. It is important to note that since the coprocessor function calls are replaced with simple sequences of register read or write instructions, the employment of the highly application specific hardware does not constitute a challenge for the C-compiler. This is an important advantage, since poor compiler performance has been the very limiting factor in the performance of application specific instruction set processors (ASIP) [16].

The complete list of the supported coprocessor function calls is given in Table I. In addition to the input parameters listed in the table, each function call includes additional parameters that can be used to tell the coprocessor to interrupt the core after the instruction has been decoded, and after the processing of the function has been completed. These can be used to align the executions of the core and the coprocessors when necessary.

The coprocessors can be regarded as a library of subroutines that are executed on dedicated and highly optimized hardware. Although the COFFEE core is the only programmable component of the platform, the API provides enough flexibility for software defined implementation of the WCDMA/OFDM baseband receiver. Furthermore, the platform is flexible enough that it can also be used for other radio technologies employing the OFDM or WCDMA air interfaces.

V. Simulation Results

The platform was simulated with a clock cycle accurate and bit accurate C/C++ testbench running on a PC workstation. SystemC libraries were used to model the fixed-point data types, the parallelism, and timing behavior [17]. The timing behavior of the system was modelled with SystemC *wait*-statements, that can be used to stall a process for a wanted amount of clock cycles. The coprocessors were modelled at register transfer level, and the COFFEE core was modelled at behavioral level. The performance of the C-compiler was estimated and *wait*-statements were inserted to the C-code accordingly. In addition to the core and the coprocessors, a source and a sink module were used in the test bench to model the behavior of the receiver front and back-ends. A behavioral memory model was also employed.

The functional simulation of the system was done with Matlab. The same Matlab model was then used to generate the test data for the test bench. The output of the test bench was verified with another Matlab model. The minimum clock frequency of the platform, needed to run the simulation was also computed. The interface between the source and the sink modules and the Matlab model was realized through text files.

A number of simulations were carried out to verify the correctness of the testbench output, and to test the programming interface of the coprocessors. In the WCDMA mode, demodulation of a downlink dedicated physical data channel (DPDCH) with three parallel code channels and the shortest spreading factor $SF = 4$, required minimum clock frequency of 168 MHz. Other WCDMA physical layer procedures considered in the simulations included the cell search procedure, random access procedure, uplink common packet channel (CPCH) procedure, and the paging procedure [14]. In the OFDM mode, Demodulation of a WLAN data packet with QPSK modulation required minimum clock frequency of 275 MHz.

VI. Conclusion

The emerging multi mode functionality and the ever shrinking time-to-market constraints of commercial wireless devices have forced wireless system designers to adopt platform-based design methodology. The effectiveness of the methodology is based on reuse of both hardware and software, and the separation of design concerns. A system platform targeted for WCDMA and OFDM baseband receivers was presented. The platform architecture is composed of a RISC core and set of coprocessors that are used to implement the most critical computation kernels of the receiver algorithms. The coprocessors are tightly coupled to the core through a coprocessor bus. Since the execution of the critical computation kernels is done on the coprocessors the performance of the platform is not limited by the compiler. An application program interface for the platform was also presented. The API enables software defined implementation of WCDMA and OFDM baseband receivers. The platform was simulated with a SystemC-based simulation environment and the minimum clock frequency estimates were given.

References

[1] J. M. Rabaey, M. Potkonjak, F. Koushanfar, S.-F. Li, and T. Tuan, “Challenges and Opportunities in Broadband and Wireless Communication Designs”, in *Proc. IEEE/ACM Conference on Computer Aided Design (ICCAD’00)*, San Jose, CA, USA, Nov. 2000, pp. 76-82.

[2] A. Ferrari and A. Sangiovanni-Vincentelli, “System Design: Traditional Concepts and New Paradigms”, in *Proc. Conference on Computer Design (ICCD '99)*, Austin, TX, USA, Oct. 1999, pp. 2-12.

[3] K. Ahmavaara, H. Haverinen, and R. Pichna, “Interworking Architecture Between 3GPP and WLAN Systems”, *IEEE Communications Magazine*, vol. 41, no. 11, Nov. 2003, pp. 74-81.

[4] 3GPP Technical Specification 23.234, *3GPP System to Wireless Local Area Network (WLAN) Interworking; System Description*, Release 6, v6.1.0, June 2004.

[5] E. Buracchini, “The Software Radio Concept”, *IEEE Communications Magazine*, vol. 38, no. 9, Sep. 2000, pp. 138-143.

[6] R. Kokozinski, D. Greifendorf, J. Stammen, and P. Jung, “Evolution of Hardware Platforms for Mobile Software Defined Radio Terminals”, in *Proc. IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’02)*, Lisbon, Portugal, Sep. 2002, vol. 5, pp. 2389-2393.

[7] L. Harju and J. Nurmi, “A Baseband Receiver Architecture for UMTS-WLAN Interworking Apllications”, in *Proc. IEEE Symposium on Computers & Communication (ISCC’04)*, Alexandria, Egypt, June 2004, pp. 678-685.

[8] J. Kylliäinen, M. Kuulusa, and J. Nurmi, “COFFEE—A Core for Free”, in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2003, pp. 17-22.

[9] COFFEE RISC website: http://www.cs.tut.fi/~coffee

[10] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*, Sams Publishing, Indianapolis, IN, USA, 2002.

[11] E.-S. Lohan, *Multipath Delay Estimators for Fading Channels in CDMA Receivers and Mobile Positioning*, PhD Thesis, Tampere University of Technology, Tampere, Finland, Oct. 2003.

[12] J.-J. van de Beek, M. Sandell, and P. O. Börjesson, “ML Estimation of Time and Frequency Offset in OFDM Systems”, *IEEE Transactions on Signal Procesing*, Vol. 45, No. 7, pp. 1800-1805, July 1997.

[13] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver”, in *Proc. IEEE Workshop on Signal Processing Systems (SIPS’02)*, San Diego, CA, USA, Oct. 2002, pp. 177-182.

[14] H. Holma and A. Toskala, *WCDMA for UMTS*, John Wiley and Sons Ltd, Chichester, West Sussex, England, 2001.

[15] T. Taskinen, *Hardware Implementation Architectures for Time-Frequency Transforms*, Master of Science Thesis, Tampere University of Technology, Tampere, Finland, 2002.

[16] R. Leupers, “Compiler Design Issues for Embedded Processors”, *IEEE Design & Test of Computers*, vol. 19, no. 4, July 2002, pp. 51-58.

[17] A. Ghosh, S. Tjiang, and R. Chandra, “System modeling with SystemC”, in *Proc. International Conference on ASIC (ASICON’01)*, Shanghai, China, Oct. 2001, pp. 18-20.

PUBLICATION 6

L. Harju and J. Nurmi, “A Synchronization Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations,” in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2005, pp. 141–145.

A Synchronization Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations

Lasse Harju and Jari Nurmi
Tampere University of Technology
Institute of Digital and Computer Systems
P.O.BOX 553, 33101 Tampere, FINLAND
Email: lasse.harju@tut.

Abstract— **Wireless communications are evolving towards multistandard systems. The complexity of mobile terminals will increase dramatically as multiple radio interfaces need to be supported. Programmability will be essential in order to manage the increased complexity of the receiver baseband processing, and to minimize product development time. A programmable coprocessor architecture is presented that is targeted for implementing the synchronization algorithms of WCDMA and OFDM receivers. The coprocessor is a part of a programmable WCDMA/OFDM baseband receiver platform targeted for dual-mode mobile terminal implementations. The coprocessor architecture is presented and the programming interface designed for the coprocessor is explained in detail. The coprocessor has been implemented with a register-transfer-level VHDL description and synthesis with 0.13 μm standard cell CMOS technology. Simulation and synthesis results are given.**

I. Introduction

Wireless communications systems are currently evolving towards multistandard systems, where existing cellular and short range radio technologies are seamlessly integrated to form a hierarchical multistandard system. These systems, that are also referred to as 4G systems, enable the users to access the same wireless services through multiple radio technologies [1]. In a multistandard environment, the users have to be equipped with devices that support multiple air interfaces. Consequently, several radio technologies need to be integrated into a single device and into a single chip. The biggest design challenge in the future, will be the management of this increased complexity [2]. Programmability is one of the most effective ways to combat this complexity and to decrease product development time. It is imperative that the baseband receiver hardware is programmable and that processing resources can be shared in the different modes of the receiver.

In this paper we present a programmable architecture targeted for implementing the synchronization algorithms of wideband code division multiple access (WCDMA) and orthogonal frequency division multiplexing (OFDM) receivers. These two radio interfaces are required in the interworking of 3GPP and IEEE 802.11a wireless LAN networks [3]. The coprocessor is designed to work with the COFFEE processor core [4], as a part of the Espresso platform presented in [5]. The Espresso platform enables software de ned implementation of dual-mode WCDMA/OFDM baseband receivers.

The paper is organized as follows. In section II, the problems related to application-speci c programmable architectures are highlighted. In section III a brief overview of WCDMA and OFDM synchronization is given. In section IV, the Espresso platform is introduced, followed by a detailed description of the synchronization coprocessor architecture in section V. The programming interface designed for the coprocessor is described in section VI and the simulation and synthesis results are given in section VII. Finally, conclusions are drawn in section VIII.

II. Receiver Programmability

The most important design objective of the Espresso platform and the proposed coprocessor architecture has been to increase programmability of the receiver hardware. Equally important has been to retain the programming productivity by freeing the programmer from the use of low level programming languages. This burden of low-level programming is a common problem of programmable application-speci c architectures [6]. Often compilers fail to make use of the application-speci c processing units of the processor, and the programmer is forced to use low-level programming or processor speci c language extensions in order to meet the performance constraints.

Traditionally, the software development task has been facilitated with reuse of existing software libraries. Ideally, when such libraries are available, the software development task is mainly composed of making code that controls the procedures offered by the libraries. With the proposed architecture, the computation resources of the coprocessors are visible to the programmer through special C-language function calls. The programmer uses these function calls to initiate the computation on the coprocessor, similarly as using functions provided by any other software library.

III. Synchronization in OFDM and WCDMA Receivers

Synchronization in WCDMA and OFDM is acquired with similar correlation-based algorithms, where the received signal is correlated against a known pilot sequence. Timing synchronization is obtained by detecting the peaks from the correlation output and the frequency synchronization is obtained by examining the phase difference of sequential correlation outputs.

WCDMA and OFDM synchronization algorithms utilize many similar computation kernels which can be exploited to share the processing resources in the receivers. The most important synchronization tasks found in WCDMA and OFDM systems are described in the following sections. A more comprehensive study of the different algorithms can be found in [7].

A. WCDMA Multipath Estimation

Multipath estimation in WCDMA receivers is needed to determine the relative delays of the multipath components in the impulse response of the channel. This information is used in the Rake ngers to align the phase of the input signal and the spreading code used in the despreading [8].

The multipath estimation is done by correlating the received signal against a pilot sequence and detecting the correlation peaks from the correlator output. The correlation is computed with convolution:

$$y(n) = \sum_{k=0}^{L-1} c(k)^* r(n-k) \tag{1}$$

where $r(n)$ is the nth sample of the complex baseband signal, $c(k)$ is the kth element of the complex pilot sequence, $y(n)$ is the nth element of the complex correlation output vector, and L is the length of the convolution. In principle, the rst correlation peak determines the acquisition point and following peaks that are within a certain window determine the other multipath components. The magnitude of a peak is proportional to the gain of the multipath, and the distance of a peak relative to the rst arrival gives a measurement of the delay of the path. In order to minimize the effect of noise and interference caused by other users, the correlation windows can be averaged non-coherently over a number of iterations [9]. The averaging is given by:

$$\hat{y}(n) = \frac{1}{N} \sum_{i=0}^{N-1} y_i(n) \tag{2}$$

where $y_i(n)$ is the nth element of the ith correlation output vector.

The multipath delay estimation has to be performed at least at accuracy of one chip. Oversampling or interpolation is required if better multipath resolution is desired. The performance of different type of multipath delay estimators are compared in [10].

B. OFDM Symbol Timing Estimation

Symbol timing estimation is used in OFDM to determine the symbol boundaries in the sample stream. The accuracy of the estimation is less critical because in OFDM a guard period is attached in front of each symbol [11]. The OFDM symbol is cyclically extended over this guard period resulting in a cyclic pre x. This redundancy allows the timing estimate to vary over an interval equal to the length of the guard period.

The timing synchronization in OFDM can be made with a similar matched lter approach as in WCDMA multipath estimation. The pilot sequences in the preamble of each data packet can be used for this purpose. Another approach is to use so-called delay-and-correlate algorithm where the received signal is correlated against a delayed and conjugated version of itself [11][12]:

$$y(n) = \sum_{k=0}^{L-1} r(n-k) r(n-D-k)^* \tag{3}$$

In the equation, D is the length of the delay. This method exploits the redundancy introduced by the cyclic pre x. Similarly, repeated pilot sequences in the preamble of the data packet can be used in the delay-and-correlate algorithm.

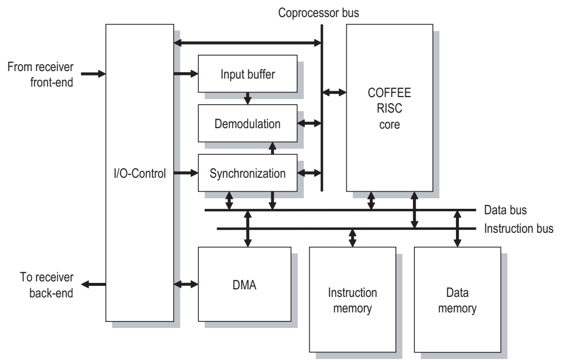


Fig. 1. The Espresso platform.

IV. The Espresso Platform

The Espresso platform enables software de ned implementation of dual-mode WCDMA/OFDM baseband receivers. The architecture, depicted in Fig. 1, is composed of the COFFEE core and three coprocessors.

In addition to the core and the coprocessors, the platform comprises direct memory access (DMA) and on-chip memory for instructions, data, and sample input buffering. The coprocessors are used for the most critical computation kernels of OFDM and WCDMA receiver algorithms. The synchronization coprocessor is used for correlation based timing and frequency synchronization algorithms and the demodulation coprocessor is used for FFT and Rake receiver functionalities. The I/O coprocessor controls the incoming sample stream and provides an interface to the receiver back-end. The channel estimation and equalization tasks are implemented with the instruction set architecture (ISA) of the COFFEE core. The core itself does not process the incoming sample stream, but only the demodulated symbols.

A unique feature of the COFFEE core is the coprocessor bus which is used for the data traf c between the coprocessors and the core. Although the coprocessors can be used with any processor that features memory mapped coprocessor connections, using a dedicated coprocessor bus spares the system bus bandwidth for other traf c.

The internal data word width of the COFFEE core is 32 bits. In the coprocessors this is divided into real and

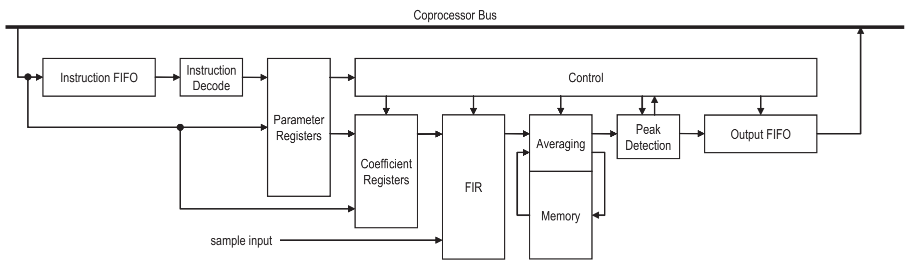


Fig. 2. The architecture of the synchronization coprocessor.

imaginary components. Consequently, the data width of the synchronization coprocessor is limited to 16 bits. However, the width of the input samples can be much less. For WCDMA receiver, 6 bits/sample resolution has been found adequate, but in OFDM, 9 bits/sample are needed [7]. In the presented coprocessor implementation, 8 bits/sample were used for both radio interfaces.

V. The Synchronization Coprocessor Architecture

The synchronization coprocessor is designed for the correlation based synchronization algorithms needed in OFDM and WCDMA receivers.

The architecture of the coprocessor, depicted Fig. 2, is composed of an instruction rst-in rst-out (FIFO) buffer, an instruction decoder, parameter registers, coef cient registers, a nite impulse response (FIR) lter block, an averaging unit, a peak detection block, control, and an output FIFO buffer. The coprocessor is connected to the coprocessor bus of the COFFEE core. The FIR block is directly connected to the I/O coprocessor that provides the sample stream to the synchronization coprocessor.

A. The FIR Block

The FIR block is used to execute convolution and delay-and-correlate algorithms, given by Eq. 1 and Eq. 3, respectively. The maximum length of the lter is $L = 256$, and the maximum delay is $D = 128$. The convolution mode of the FIR block, with convolution length $L = 4$, is illustrated in Fig. 3(a) and the delay-and-correlate mode, with delay $D = 4$ and correlation length $L = 4$, is illustrated in Fig. 3(b).

Typically in WCDMA synchronization, the value of the coef cients is always ± 1. Similarly in OFDM, only the sign bits of the coef cients can be used for the synchronization [13]. For this reason, only 1-bit complex coef cients can be used in the FIR.

The FIR block includes a switching network that feeds either the coef cients or the delayed version of the received signal to the inputs of the complex multiply-accumulate (MAC) units. Because the value of the real and imaginary parts of the coef cients is limited to ± 1, the complex MAC operation is simpli ed into a two stage add/subtract computation [8].

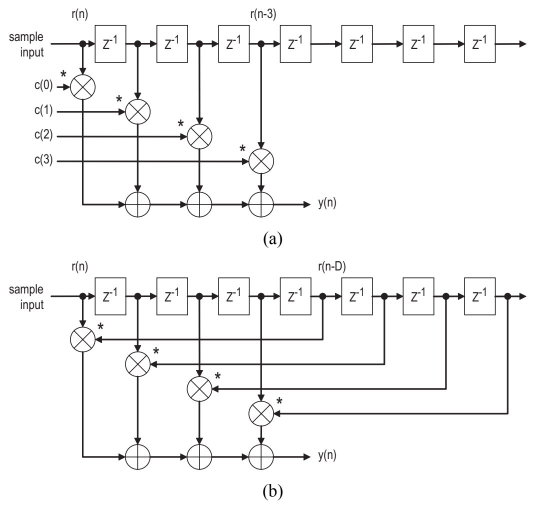


Fig. 3. The FIR block in (a) convolution mode ($L = 4$) and (b) delay-and-correlate mode ($L = 4$, $D = 4$).

In order to achieve high clock rates the FIR block was divided into 16 pipeline stages, each composed of 16 FIR taps.

B. Averaging and Peak Detection

The output of the FIR block is rst fed to the averaging unit and then into the peak detection block. These are depicted in Fig. 4. The averaging unit accumulates sequential correlation output vectors and divides the resulting vector with the number of averaging iterations, according to Eq. 2. The averaging computation is implemented with combinatory logic.

The size of the memory in the averaging unit determines the maximum length of the averaged correlation output vector, which in turn determines the length of the channel tracking window. For example, in a WCDMA receiver, a memory of 2048 bytes results in a tracking window of 16.67 μs when 16 bit correlation registers, for both real and imaginary components, and oversampling rate of four are assumed. Dual port memory is required to allow simultaneous read and write accesses.

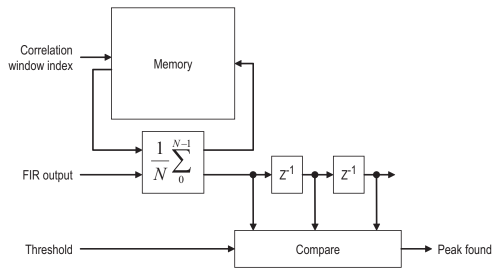


Fig. 4. The averaging unit and the peak detection block.

The peak detection following the averaging is made by comparing three sequential averaged correlation outputs against the threshold value. The comparison is implemented with combinatory logic. When the peak detector detects a local maxima crossing the threshold, it sets a signal that informs the control block that a peak has been found.

VI. The Programming Interface

The programming interface of the coprocessors is implemented with a library of coprocessor function calls. The programmer writes code for the COFFEE core and uses these function calls to initiate the computation kernels on the coprocessors. The function calls are replaced by the compiler with a sequence of assembly instructions that write the instruction and the input parameters to the instruction FIFO of the coprocessor.

The function calls used for programming the synchronization coprocessor are described in the following sections. In addition to the functions listed below, the programmer can use coprocessor read and write functions to access the parameter and coef cient registers and the output FIFO.

Scrambling Code Initialization

The synchronization coprocessor includes generators for WCDMA scrambling codes. The `init_scode` function is use to initialize the generator, by giving the scrambling code number as a parameter. The output of the scrambling code generator is fed straight to the FIR block and used with the `corr_x_thr` function as described in a following section.

Initialization of Correlation Coefficients

The `init_corr_coef` function is used to initialize a coef cient vector to one of the 8 locations in the coef cient register bank. The destination location number is speci ed with coef cient handle. The length of the coef cient vector and the coef cient handle are given as parameters. The initialization is performed by rst giving the `init_corr_coef` command and then sequentially writing the coef cients to a speci c coprocessor register.

Setting the Tracking Parameters

The `set_track_param` function is used to set the parameters for the correlation tracking mode. In the tracking mode, the sequential correlation windows are averaged. The length of the tracking window, the gap length between tracking windows, and the number of averaging iterations are given as parameters. During the gap, the delay line of the FIR block is shifted without any computations. This feature is bene cial when the correlation needs to be computed against a certain eld of a transmission slot, and averaged over several sequential slots.

Setting the Detection Threshold

The `set_thr` sets the threshold value used by the peak detector. The 32-bit threshold value is given as a parameter.

Correlation

The `corr_x_thr` function initiates the correlation between the input sample stream and the correlation coef cients. The type of the correlation (convolution/delay-and-correlate), the length of the delay, the length of the correlation, and the coef cient handle are given as parameters. The scrambling code can also be activated with an control ag, whereupon the correlation coef cients are scrambled with the scrambling code. The scrambling code index updates automatically as the correlation window moves. The execution of the correlation is divided into two stages:

1) The correlation is computed until the rst correlation peak crossing the threshold is detected. The coprocessor stores the index of the peak to the output FIFO and interrupts the core.
2) If tracking parameters have been speci ed, the tracking mode is started. The correlation output vectors are averaged over a number of iterations using the given tracking window and gap length parameters. During the last iteration all the peaks within the window length crossing the threshold are detected and their indices are stored to the output FIFO. Finally, the core is interrupted.

VII. Simulation and Synthesis Results

The platform was rst simulated with a clock cycle accurate and bit accurate C/C++ testbench, running on a PC workstation. The structure of the testbench is depicted in Fig. 5. SystemC libraries were used to model the x ed-point data types, the parallelism, and timing behavior. The coprocessors were modeled at a level equivalent to register transfer level (RTL), and a behavioral model of COFFEE core was employed. In addition to the core and the coprocessors, a source and a sink module were used to model the behavior of the receiver front-end and back-end. The necessary memory blocks were modeled at behavioral level. The test data for the test bench was created with a Matlab model, and the output of the test bench was compared with that of the Matlab model. The interface between the source and the sink modules and the Matlab model was realized through text les.

The functional simulations clearly demonstrated that the concept of the Espresso platform is a feasible and competent approach for software de ned baseband receiver implementations. The simulations also proved that the programming interface designed for the synchronization coprocessor provides

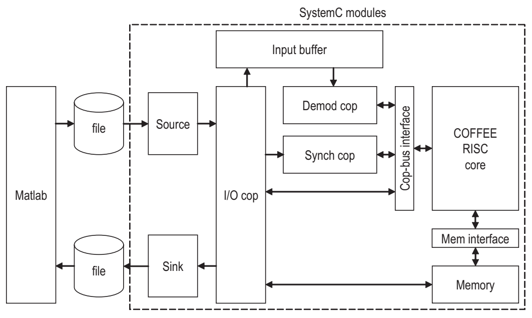


Fig. 5. The testbench used in the simulations.

the required functions, parameters, and control ags for the WCDMA and OFDM synchronization algorithms. In addition, the minimum clock frequency of the platform needed to run the computation was estimated. Simulation of the WCDMA multipath estimation resulted in a clock frequency requirement of 62 MHz, when oversampling factor of four was assumed. Simulation of OFDM symbol timing estimation resulted in a clock frequency of 80 MHz, when Nyquist sampling was assumed.

After the functional simulations, the coprocessor was implemented with RTL VHDL description. As the simulation models were already done at a level equivalent to RTL, the transition from SystemC to VHDL was facilitated. Synthesis with a 0.13 μm standard cell CMOS technology resulted in 2.02 mm^2 chip area, which is equivalent to approximately 334000 gates. A simulation of WCDMA synchronization with the synthesized gate-level circuit resulted in a power consumption of 29 mW with a clock frequency of 62 MHz and 1.32 V supply voltage. Similarly, an OFDM synchronization simulation resulted in 49 mW with a clock frequency of 80 MHz.

VIII. Conclusion

Programmability of baseband receiver architectures will be essential in order to manage the design complexity of future multistandard mobile terminals. The proposed coprocessor architecture enables programmable implementation of WCDMA and OFDM synchronization algorithms. The programming interface of the coprocessor is realized with a set of coprocessor function calls. Because these functions are only used for passing the input parameters to the coprocessor and initiating the computation, the employment of the application-speci c hardware resources does not require a customized compiler or limit the productivity of the software development. This approach solves the programming ef cienc y problem related to application-speci c processors. The coprocessor architecture has been simulated on a SystemC-based testbench and implemented with RTL level VHDL description. The area of the synthesized chip and the simulated power consumption indicate that the architecture is suitable for mobile terminal implementations.

Acknowledgment

This work has been supported by the EU-project SoC-MOBINET (IST 2000-30094), the Nokia Foundation, the Finnish Cultural Foundation, Walter Ahlstrom Foundation, and the HPY Research Foundation.

References

[1] S. Y. Hui and K. H. Yeung, “Challenges in the Migration to 4G Mobile Systems,” *IEEE Communications Magazine*, vol. 41, no. 12, pp. 54–59, Dec. 2003.
[2] Y. Neuvo, “Cellular Phones as Embedded Systems,” in *Digest of Technical Papers IEEE Solid-State Circuits Conference*, Feb. 2004, vol. 1, pp. 32–37.
[3] *3GPP System to Wireless Local Area (WLAN) Network Interworking; System Description*, 3GPP Technical Speci cation 23.234, Rev. 6.1.0, 2004.
[4] J. Kylliainen, M. Kuulusa, and J. Nurmi, “COFFEE—A Core for Free,” in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2003, pp. 17–22.
[5] L. Harju and J. Nurmi, “A Programmable Baseband Receiver Platform for WCDMA/OFDM Mobile Terminals,” in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 1, New Orleans, LA, USA, Mar. 2005, pp. 33–38.
[6] R. Leupers, “Compiler Design Issues for Embedded Processors,” *IEEE Design & Test of Computers*, vol. 19, no. 4, pp. 51–58, July 2002.
[7] L. Harju and J. Nurmi, “A Baseband Receiver Architecture for UMTS/WLAN Interworking Apllications,” in *Proc. IEEE Symposium on Computers and Communications*, vol. 2, Alexandria, Egypt, June 2004, pp. 678–685.
[8] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver,” *The Journal of VLSI Signal Processing*, vol. 39, no. 1–2, pp. 147–160, Apr. 2005.
[9] A. Huang, M. Hall, and I. Hartimo, “Multipath Channel Estimation for WCDMA Uplink,” in *Proc. IEEE Vehicular Technology Conference*, vol. 1, Amsterdam, Netherlands, Sept. 1999, pp. 141–145.
[10] E.-S. Lohan, “Multipath Delay Estimators for Fading Channels in CDMA Receivers and Mobile Positioning,” Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, Oct. 2003.
[11] R. van Nee and R. Prasad, *OFDM Wireless Multimedia Communications*. Norwood, MA, USA: Artech House, 2000.
[12] J.-J. van de Beek, M. Sandell, and P. O. Borjersson, “ML Estimation of Time and Frequency Offset in OFDM Systems,” *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800–1805, July 1997.
[13] J.-J. van de Beek, M. Sandell, M. Isaksson, and P. O. Borjersson, “Low-Complex Frame Synchronization in OFDM Systems,” in *Proc. IEEE Conference on Universal Personal Communications*, Tokyo, Japan, Nov. 1995, pp. 982–986.

PUBLICATION 7

L. Harju and J. Nurmi, "A Demodulation Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations," in *Proc. NORCHIP Conference*, Oulu, Finland, Nov. 2005, pp. 66–69.

A Demodulation Coprocessor Architecture for WCDMA/OFDM Mobile Terminal Implementations

Lasse Harju and Jari Nurmi
Tampere University of Technology, Institute of Digital and Computer Systems
P.O.BOX 553, 33101 Tampere, Finland
E-mail: lasse.harju@tut.fi

Abstract:

Programmability of the baseband receiver hardware will be essential in the future as multiple radio technologies need to be supported by the mobile terminal. A programmable coprocessor architecture is presented in this paper that can be used to implement demodulation procedures of WCDMA and OFDM receiver. The coprocessor architecture is presented and the programming interface designed for the coprocessor is explained in detail. Simulation and synthesis results are also given.

1. Introduction

Wireless communications are evolving towards multistandard systems. In the future, users can employ several existing radio technologies to access the same wireless resources [1]. In a multistandard environment, the users have to be equipped with mobile terminals that support multiple air interfaces. Consequently, the complexity of the receiver hardware increases dramatically as several radio technologies need to be integrated into a single device and into a single chip. Management of the baseband receiver complexity will be one of the biggest design challenges of future mobile devices [2]. It is imperative that the receiver hardware is programmable and that processing resources can be shared in the different modes of the receiver.

In this paper we present a programmable architecture that can be used to implement the demodulation procedures of wideband code division multiple access (WCDMA) and orthogonal frequency division multiplexing (OFDM) receivers. These two radio interfaces are conjoined the interworking of 3GPP and IEEE 802.11a wireless LAN networks [3]. The coprocessor is designed to work with the COFFEE processor core [4], as a part of the Espresso platform [5].

The paper is organized as follows. In section 2, problems commonly associated with application-specific programmable architectures are highlighted. In section 3, a short overview is given on the demodulation procedures of WCDMA and OFDM receivers. The Espresso platform is introduced in section 4, and the architecture of the demodulation coprocessor is presented in section 5. A detailed description of the programming interface is given in section 6, followed by the simulation and synthesis results in section 7. Finally, conclusions are drawn in section 8.

2. Receiver Programmability

The most important design objective of the Espresso platform and the proposed coprocessor architecture has been to increase programmability of the receiver hardware. The programming interface of the coprocessor has been designed so that the programmer will be freed from the use of low-level programming languages. The burden of low-level programming has been a common problem of programmable application-specific architectures [6]. Often compilers fail to make use of the application-specific processing units of the processor, and the programmer is forced to use low-level programming or processor specific language extensions in order to meet the performance constraints.

Traditionally, the software development task has been facilitated with reuse of existing software libraries. Ideally, when such libraries are available, the software development task is mainly composed of making code that controls the procedures offered by the libraries.With the proposed architecture, the computation resources of the coprocessors are visible to the programmer through special C-language function calls. The programmer uses these function calls to initiate the computation on the coprocessor, similarly as using functions provided by any other software library.

3. WCDMA and OFDM Demodulation

In WCDMA receivers the demodulation is performed by a Rake receiver. The received signal is correlated with a spreading code over a period corresponding to the spreading factor. In traditional Rake receivers one finger is dedicated to each multipath components [7].

In OFDM receivers, the demodulation is performed by applying 64-point fast Fourier transform (FFT) to the samples within a symbol window [8].

A more detailed study of the different receiver algorithms of WDCMA and OFDM receivers can be found in [9].

4. The Espresso Platform

The Espresso platform enables software defined implementation of dual-mode WCDMA/OFDM baseband receivers. The architecture, depicted in Fig. 1, is composed of the COFFEE core and three coprocessors.

The coprocessors are designed for the most critical computation kernels of OFDM and WCDMA receiver algorithms. The synchronization coprocessor is used for correlation based timing and frequency synchronization algorithms, and the demodulation coprocessor is used for FFT and Rake receiver functionalities. The I/O coprocessor controls the incoming sample stream and provides an interface to the receiver back-end. The channel estimation and equalization tasks are implemented with the instruction set architecture (ISA) of the COFFEE core. The core itself does not process the incoming sample stream, but only the demodulated symbols.

5. The Demodulation Coprocessor Architecture

The architecture of the demodulation coprocessor, depicted Fig. 2, is composed of an instruction first-in first-out (FIFO) buffer, an instruction decoder, parameter registers, spreading and scrambling code generators, twiddle factor generators, coefficient registers, the datapath, control, and an output FIFO. Coefficient registers are provided for storing up to 24 coefficient vectors, which can be up to 256 elements long. The coprocessor is connected to the coprocessor bus of the COFFEE core. Through the bus the programmer can access the instruction FIFO, the parameter registers, the coefficient registers, and the output FIFO. The Datapath is directly connected to the sample input buffer.

The implementation of the coprocessor Rake mode is based on the FlexRake architecture, depicted in Fig. 3 [10]. A traditional Rake receiver is composed of parallel fingers that are each used to despread one multipath component of the received WCDMA signal. The essence of the FlexRake architecture is that parallelism is only used for despreading parallel downlink channels and the multipath components are processed sequentially. The delays of the multipath components are used to compute the addresses for the input buffer read accesses. Consequently, the as the radio channel changes, only the read addresses need to be updated. The coprocessor also includes generators for orthogonal variable spreading factor (OVSF) and scrambling codes.

The implementation of the coprocessor FFT mode is based on the single delay feedback (SDF) architecture [11], depicted in Fig. 4. In this architecture, a pipeline stage is dedicated for each stage of the FFT, i.e., one butterfly operation is performed per stage at every clock cycle. The input samples are read sequentially from the input buffer and the correct input pairs for the butterflies are acquired by the delay lines at each stage. The architecture can be used to execute FFTs of length 8, 16, 32, or 64. Twiddle factor generators are also needed to implement the FFT functionality.

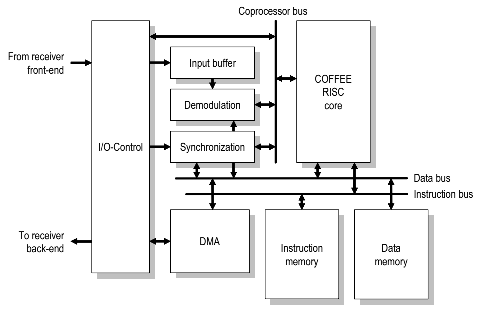


Fig. 1. The Espresso platform.

In addition to the Rake and FFT functionalities the coprocessor can be used to execute upto 24 parallel complex valued correlations. Six parallel correlators are provided which can be used in four sequential sets. The idea of the sequential sets is that each of them can be fed with a different sample input, i.e., from a different input buffer address. The parallel correlators are always fed with the same input. Up to 32 complex valued coefficients vectors, with 1-bit real and imaginary components, can be loaded to the coefficient registers. This feature is needed, e.g., in the second phase of the cell search procedure in WCDMA [7].

5.1. The Datapath

The datapath of the coprocessor includes six processing units (PU). These can execute a butterfly operation or simple a multiply-accumulate operation. In the FFT mode, the PUs are connected in the pipeline structure of the SDF architecture, and in the Rake mode, each of the PUs can be used as a parallel correlator. Each PU is connected to the sample input, twiddle/code input, the register resources, and the adjacent PUs. All the register resources are connected to a crossbar switch that is responsible for establishing the connections between the registers and PUs. In 64-point FFT, a total of 126 registers are needed to form the delay line structures of the SDF architecture and in the Rake mode 48 registers are needed for despreading six parallel code channel and four multipath components. These registers have to be double length to avoid overflows when despreading with long spreading factors.

5.2. The Processing Unit

The processing units perform the butterfly operations in the FFT mode, and the multiply-accumulate operation in the Rake mode. A butterfly operation in decimation-in-frequency (DIF) FFT is composed of one complex addition, one complex subtraction and one complex multiplication. This requires a total of four adders, two subtracters, and four multipliers. In the Rake mode, the operation of the PUs is much simpler. As the spreading

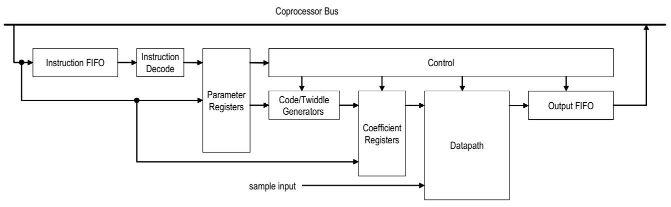


Fig. 2. The architecture of the Rake/FFT coprocessor.

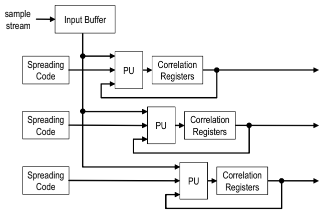


Fig. 3. The FlexRake architecture.

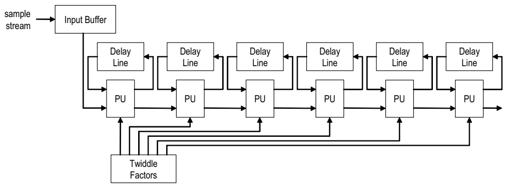


Fig. 4. A single delay-feedback implementation of the FFT.

and scrambling codes are always sequences of ±1 the complex multiplication in the correlations is simplified to a simple sign change operation. Hence, the whole multiply-accumulate operation in the Rake mode can be executed with a two stage adder/subtracter structure. The computation resources are shared between the two modes of the PU, and the selection between the two modes is made automatically by the hardware.

6. The Programming Interface

The programming interface of the coprocessor is implemented with a library of coprocessor function calls. The programmer writes code for the COFFEE core and uses these function calls to initiate the computation kernels on the coprocessors. The function calls used for programming the demodulation coprocessor are described in the following sections. In addition to these functions, the programmer can use coprocessor read and write functions to access the parameter and coefficient registers, and the output FIFO.

6.1. Spreading Code Initialization

The `init_ovsf` function initializes the OVSF code generators in the demodulation coprocessor. The spreading factor, the OVSF code number are given as parameters. The output of the OVSF generators is automatically fed to the correct correlators.

6.2. Scrambling Code Initialization

The `init_scode` function initializes the scrambling code generator in the demodulation coprocessor. The scrambling code number is given as a parameter. When the initialization is completed, the core is interrupted. The output of the scrambling code generator is fed straight to the correlators when used with the despread function and the phase of the code is automatically incremented during the correlation.

6.3. Multipath Delay Setting

The `init_multipaths` function is used to set the delay profile of the multipath channel. The delays of the multipath components are used in the coprocessor to access the correct samples from the input buffer. The delays are given as input parameters in integer multiples of samples and up to three delays can be set. If no multipath delays are set, all correlators are fed from the same input buffer address.

6.4. Initialization of Correlation Coefficients

The `init_corr_coef` function initializes a coefficient vector, other than an OVSF code or twiddle factors, to one of the coefficient registers of the coprocessor. The desired coefficient register number is as a parameter. The initialization is performed by first giving the `init_corr_coef` command and then sequentially writing the coefficients to a specific coprocessor register. The length of the coefficient vector is also given as a parameter. After decoding the `init_corr_coef` operation, the coprocessor waits until the coefficients have been written into the coefficient register.

6.5. Despreading

The `despread` function initiates the despreading operation (or any other correlations) in the demodulation coprocessor. The number of the parallel correlations is given as a parameter. The coefficients for each correlator are read from specific coefficient registers. The start address of the correlation window is given as a parameter. This address is used as the base for the offset addressing that employs the multipath delays. Other input parameters include the number of symbols to be despread (number of correlation iterations), a flag that turns the feed of the scrambling code to the PUs on and off. After the despreading execution completes, the coprocessor writes the demodulated symbols to the output FIFO and interrupts the core.

6.6. FFT Computation

The `fft` function initiates the FFT computation in the demodulation-coprocessor. The start address of the FFT window and the length of the FFT are given as parameters. The twiddle factors are automatically initialized upon the first FFT computation, or whenever the length of the FFT changes. When the execution completes, the outputs are reordered and written to the symbol FIFO, and the core is interrupted.

7. Simulation and Synthesis Results

The platform was first simulated with a clock cycle accurate and bit accurate C/C++ testbench, running on a PC workstation. SystemC libraries were used to model the fixed-point data types, the parallelism, and timing behavior.

The main focus of the functional simulations was to test the platform concept and the programming interface designed for the coprocessor. In addition, the minimum clock frequency of the platform, needed to run the simulated procedure was computed. In the WCDMA mode, demodulation of a downlink dedicated physical data channel (DPDCH) with three parallel code channels and spreading factor SF=32, required minimum clock frequency of 90 MHz. In the OFDM mode, demodulation of a WLAN data packet with QPSK modulation required minimum clock frequency of 120 MHz.

After the functional simulations, the coprocessor was implemented with RTL VHDL description. Synthesis with a 0.13 μm standard cell CMOS technology resulted in 1.32 mm^2 chip area. A simulation of WCDMA demodulation with the synthesized gate-level circuit resulted in a power consumption of 33.8 mW with a clock frequency of 90 MHz and 1.32 V supply voltage. Similarly, an OFDM synchronization simulation resulted in 47.2 mW with a clock frequency of 120 MHz.

8. Conclusions

As wireless communications evolve towards multistandard systems, managing the system complexity of portable wireless devices becomes one of the biggest design challenges. Programmability of the baseband receiver will be of paramount importance. The proposed coprocessor architecture enables programmable implementation of WCDMA and OFDM demodulation. The programming interface of the coprocessor is realized with a set of coprocessor function calls. Because these functions are only used for passing the input parameters to the coprocessor and initiating the computation, the employment of the application-specific hardware resources does not require a customized compiler or limit the productivity of the software development. The datapath of the coprocessor is optimized for executing Rake receiver and FFT functionalities. The coprocessor architecture has been simulated on a SystemC-based testbench and implemented with RTL level VHDL description. An overview of the provided coprocessor functions and results of the simulations were given.

9. Acknowledgments

This work has been supported by the EU-project SoC-MOBINET (IST 2000-30094), the Nokia Foundation, the Finnish Cultural Foundation, Walter Ahlstrom Foundation, and the HPY Research Foundation.

10. References

[1] S. Y. Hui and K. H. Yeung, “Challenges in the Migration to 4G Mobile Systems,” *IEEE Communications Magazine*, vol. 41, no. 12, pp. 54–59, Dec. 2003.

[2] Y. Neuvo, “Cellular Phones as Embedded Systems,” in *Digest of Technical Papers IEEE Solid-State Circuits Conference*, Feb. 2004, vol. 1, pp. 32–37.

[3] *3GPP System to Wireless Local Area (WLAN) Network Interworking; System Description*, 3GPP Technical Specification 23.234, Rev. 6.1.0, 2004.

[4] J. Kylliainen, M. Kuulusa, and J. Nurmi, “COFFEE—A Core for Free,” in *Proc. International Symposium on System-on-Chip*, Tampere, Finland, Nov. 2003, pp. 17–22.

[5] L. Harju and J. Nurmi, “A Programmable Baseband Receiver Platform for WCDMA/OFDM Mobile Terminals,” in *Proc. IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, Mar. 2005.

[6] R. Leupers, “Compiler Design Issues for Embedded Processors,” *IEEE Design & Test of Computers*, vol. 19, no. 4, pp. 51–58, July 2002.

[7] H. Holma and A. Toskala, *WCDMA for UMTS*, John Wiley and Sons Ltd, West Sussex, England, 2001.

[8] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guid*e, Sams Publishing, Indianapolis, IN, USA, 2002.

[9] L. Harju and J. Nurmi, “A Baseband Receiver Architecture for UMTS/WLAN Interworking Apllications,” in *Proc. IEEE Symposium on Computers and Communications*, vol. 2, Alexandria, Egypt, June 2004, pp. 678–685.

[10] L. Harju, M. Kuulusa, and J. Nurmi, “Flexible Implementation of a WCDMA Rake Receiver,” *The Journal of VLSI Signal Processing*, vol. 39, no. 1–2, pp. 147–160, Apr. 2005.

[11] T. Taskinen, *Hardware Implementation Architectures for Time-Frequency Transforms*, Master of Science Thesis, Tampere University of Technology, Tampere, Finland, 2002.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O. Box 527
FIN-33101 Tampere, Finland