Linköping Studies in Science and Technology
Dissertations, No. 1084

Design of programmable multi-standard baseband processors

Anders Nilsson

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Electrical Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2007

Design of programmable multi-standard baseband processors

Anders Nilsson

ISBN 978-91-85715-44-2

Linköping Studies in Science and Technology,
Dissertations, No. 1084
ISSN 0345-7524

Division of Computer Engineering,
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

Author e-mail: anders.h.nilsson@gmail.com

Cover illustration: 16-QAM constellation diagram. This is an example of a data-symbol recovered from a received waveform by the baseband processor.

Printed by LiU-Tryck, Linköping University,
Linköping, Sweden, 2007

To my mother

I dedicate this work to my mother, Hatice Nilsson who in her pursue of new knowledge took me along with her to Linköping University and introduced me to my current department in 1986 when I was only six years old.

Abstract

Background

Efficient programmable baseband processors are important to enable true multi-standard radio platforms as convergence of mobile communication devices and systems requires multi-standard processing devices. The processors do not only need the capability to handle differences in a single standard, often there is a great need to cover several completely different modulation methods such as OFDM and CDMA with the same processing device. Programmability can also be used to quickly adapt to new and updated standards within the ever changing wireless communication industry since a pure ASIC solution will not be flexible enough. ASIC solutions for multi-standard baseband processing are also less area efficient than their programmable counterparts since processing resources cannot be efficiently shared between different operations. However, as baseband processing is computationally demanding, traditional DSP architectures cannot be used due to their limited computing capacity. Instead VLIW- and SIMD-based processors are used to provide sufficient computing capacity for baseband applications. The drawback of VLIW-based DSPs is their low power efficiency due to the wide instructions that need to be fetched every clock cycle and their control-path overhead. On the other hand, pure SIMD-based DSPs lack the possibility to perform different concurrent operations. Since memory access power is the dominating part of the power consumption in a processor, other alternatives should be investigated.

New architecture, SIMT

In this dissertation a new and unique type of processor architecture has been designed that instead of using the traditional architectures has started from the application requirements with efficiency in mind. The architecture is named "Single Instruction stream Multiple Tasks", SIMT in short. The SIMT architecture uses the vector nature of most baseband programs to provide a good trade-off between the flexibility of a VLIW processor and the processing efficiency of a SIMD processor. The contributions of this project are the design and research of key architectural components in the SIMT architecture as well as development of design methodologies. Methodologies for accelerator selection are also presented. Furthermore data dependency control and memory management are studied. Architecture and performance characteristics have also been compared between the SIMT and more traditional processor architectures.

Demonstrator

A complete system is demonstrated by the BBP2 baseband processor that has been designed using SIMT technology. The SIMT principle has previously been proven in a small scale in silicon in the BBP1 processor implementing a Wireless LAN transceiver. The second demonstrator chip (BBP2) was manufactured early 2007 and implements a full scale system with multiple SIMD clusters and a controller core supporting multiple threads. It includes enough memory to run symbol processing of DVB-H/T, WiMAX, IEEE 802.11a/b/g and WCDMA, and the silicon area is 11 mm^2 in a 0.12 $\mu\mathrm{m}$ CMOS technology.

Populärvetenskaplig sammanfattning

Trådlös kommunikation, resursproblem

Trådlös kommunikation tar över alltmer av de funktioner som tidigare utnyttjade trådbundna nät. Idag är mobiltelefoni och trådlöst nät till datorer en självklarhet, liksom även mobil-TV kommer att vara det inom en snar framtid. Trenden är att integrera fler och fler trådlösa standarder i bärbara apparater som t.ex. mobiltelefoner. Idag är det inte ovanligt med mobiltelefoner som stödjer upp till fem olika trådlösa standarder som t.ex. GSM, 3G, Bluetooth, GPS, trådlöst LAN och digital-TV. Traditionellt implementerar man var och en av dessa funktioner på separata chip eller delar av chip vilket leder till en stor chiparea som i sin tur medför en högre implementationskostnad och strömförbrukning. Genom att istället använda en programmerbar processor för att realisera dessa funktioner kan kiselarean användas mer flexibelt genom att man laddar olika program i processorn, ungefär som på en PC. På så sätt kan en och samma processor användas för att stöjda en stor mängd standarder och samtidigt minska både kostnad och strömförbrukning för ett trådlöst system. Till skillnad från vanliga PC-program är signalbehandlingsalgoritmerna mycket beräkningskrävande. Det är inte ovanligt att programvaran för t.ex. en 3G-mottagare kräver en beräkningskapacitet större än den sammanlagda kapaciteten hos 10 pentiumprocessorer á 2 GHz. Till detta kommer kravet på låg effektförbrukning. I en mobiltelefon får processorn inte konsumera mer än en hundradel av vad en pentiumprocessor konsumerar för

att bibehålla en lång batteritid.

Ny arkitektur, SIMT

En lösning på problemet att kunna stödja flera standarder och samtidigt ha en låg effektförbrukning ges i detta doktorsarbete. En ny processorarkitektur har tagits fram som är optimerad för signalbehandling av basband och som är flexibel nog för att stödja olika trådlösa standarder. Genom den unika arkitekturen, d.v.s. den logiska uppbyggnaden av processorn, är det möjligt att exekvera många parallella uppgifter i processorn med bara ett enkelt flöde av instruktioner. Detta ger hög beräkningskapacitet även vid låg klockfrekvens hos processorn. Dessutom innebär användandet av ett enkelt flöde av instruktioner att kontrollvägen och programminnet i processorn kan reduceras vilket ger ytterligare besparingar av både kiselarea och effekt. På grund av sitt arbetssätt kallas arktekturen SIMT (Single Instruction stream Multiple Tasks). I avhandlingen presenteras uppbyggnaden av SIMT, karakteristiska egenskaper och prestanda samt jämförelser med alternativa och idag använda arkitekturer.

Ytterligare anledningar till att använda programmerbara basbandsprocessorer är flexibiliteten och framtidssäkerheten. Programmerbarheten kan användas till att fixa buggar eller uppgradera produkter till att stödja helt nya standarder bara genom att byta ut programvaran. Detta är inte möjligt med dagens lösningar, som baserar sig på kretsar som är specialbyggda för att bara klara en specifik funktion och standard.

Demonstrator

För att demonstrera SIMT-arkitekturen har en processor baserad på denna arkitektur implementerats. Processorn kallas BBP2 och stödjer många av dagens trådlösa standarder såsom 3G, WiMAX och digital-TV. Dessutom finns stöd för morgondagens mobilstandarder såsom 4G/LTE. BBP2-chipet, som hanterar modem-funktionaliteten i dessa standarder, är realiserat i en $0.12\ \mu\mathrm{m}$ CMOS-teknologi och upptar där endast $11\ \mathrm{mm}^2$, vilket är konkurrenskraftigt jämfört med traditionella lösningar.

Preface

This thesis presents my research from October 2003 to April 2007. Material from the following six papers is included in the thesis:

- **Anders Nilsson** and Dake Liu; Area efficient fully programmable baseband processors; Accepted for publication by SAMOS VII Workshop; SAMOS, Greece, July 16 - 19, 2007.
- **Anders Nilsson**, Eric Tell, and Dake Liu; Simultaneous multi-standard support in programmable baseband processors; in Proceedings of IEEE PRIME 2006, Otranto, Italy, June 2006
- **Anders Nilsson**, Eric Tell, Daniel Wiklund, and Dake Liu; Design methodology for memory-efficient multi-standard baseband processors; Asia Pacific Communication Conference, Perth, Australia, Oct 2005
- **Anders Nilsson**, Eric Tell, and Dake Liu; A Programmable SIMD-based Multi-standard Rake Receiver Architecture; European Signal Processing Conference, EUSIPCO, Antalya, Turkey, Sep 2005
- **Anders Nilsson**, Eric Tell, and Dake Liu; A fully programmable Rake-receiver architecture for multi-standard baseband processors; Networks and Communication Systems, Krabi, Thailand, May 2005
- **Anders Nilsson**, Eric Tell, and Dake Liu; An accelerator structure for programmable multi-standard baseband processors; WNET2004, Banff, AB, Canada, July 2004

The following papers, which are also related to my research, are not included in the dissertation:

- **Anders Nilsson** and Dake Liu; Multi-standard support in SIMT programmable baseband processors; Proc of the Swedish System-on-Chip Conference (SSoCC), Kolmården, Sweden, May 2006

- H Jiao, **Anders Nilsson**, and Dake Liu; MIPS Cost Estimation for OFDM-VBLAST systems; IEEE Wireless Communications and Networking Conference, Las Vegas, NV, USA, Apr 2006

- Eric Tell, **Anders Nilsson**, and Dake Liu; A Low Area and Low Power Programmable Baseband Processor Architecture; Proc of the International workshop on SoC for real-time applications, Banff, Canada, July 2005

- Eric Tell, **Anders Nilsson**, and Dake Liu; A Programmable DSP core for Baseband Processing; Proc of the IEEE Northeast Workshop on Circuits and Systems (NEWCAS), Quebec City, Canada, June 2005

- **Anders Nilsson**, Eric Tell, and Dake Liu; Acceleration in multi-standard baseband processors; Radiovetenskap och Kommunikation, Linköping, Sweden, June 2005

- Eric Tell, **Anders Nilsson**, and Dake Liu; Implementation of a Programmable Baseband Processor; Proc of Radiovetenskap och Kommunikation (RVK), Linköping, Sweden, June 2005

- Dake Liu, Eric Tell, **Anders Nilsson**, and Ingemar Söderquist; Fully flexible baseband DSP processors for future SDR/JTRS; Western European Armaments Organization (WEAO) CEPA2 Workshop, Brussels, Belgium, March 2005

- Dake Liu, Eric Tell, and **Anders Nilsson**; Implementation of Programmable Baseband Processors; Proc of CCIC, Hangzhou, China, Nov 2004

- **Anders Nilsson** and Dake Liu; Processor friendly peak-to-average reduction in multi-carrier systems; Proc of the Swedish System-on-Chip Conference (SSoCC), Båstad, Sweden, March 2004

I have co-authored two book chapters:

- **Anders Nilsson** and Dake Liu; Handbook of WiMAX; To be published 2007, CRC Press
- Dake Liu, **Anders Nilsson** and Eric Tell; Radio design in Nanometer Technologies; ISBN 978-1402048234, Springer 2006

I am also co-author of three pending US patents related to the area of baseband processing:

- Programmable digital signal processor having a clustered SIMD micro-architecture including a complex short multiplier and an independent vector load unit.
- Programmable digital signal processor including a clustered SIMD micro-architecture configured to execute complex vector instructions.
- Digital signal processor including a programmable network.

Contributions

The main contributions of the presented work can be summarized in the following points:

- Design and research on key architectural components in the SIMT framework as well as development of design methodologies for SIMT baseband processors.
- Research on aspects of the design of programmable baseband processors such as hardware/software partitioning, instruction set design, design and selection of accelerators, multi-standard execution and memory management for baseband processors.
- Development of a methodology for accelerator selection.
- Development of design methodologies and architecture support in execution units, memory system and controller core for multi-standard execution.
- Mapping and benchmarking of symbol processing functions from several diverse standards (WCDMA, DVB-H, WiMAX and Wireless LAN) to the SIMT architecture in conjunction with instruction set design, execution unit design and accelerator selection.
- Design, development and implementation of an area and power efficient programmable baseband processor suitable for multi-standard baseband processing. The flexibility as well as the low silicon area of the SIMT architecture is proven by the BBP2 processor.

In addition to the above mentioned points, significant work has been performed on algorithm design and selection for implementation and benchmarking of the SIMT architecture although this is not included in this dissertation.

Acknowledgments

It is a great pleasure for me to express my sincere gratitude to my supervisor Professor Dr. Dake Liu for his great interest in my work and for his valuable advice and encouragement during all my time at the Division of Computer Engineering. I would also like to thank my closest cooperator Dr. Eric Tell for many fruitful discussions during the last four years and for his help during the BBP2 project.

Many other friends and family members have also made my time as a PhD-student enjoyable. I would accordingly like to thank you all:

- Lic. Eng. Henrik Fredriksson for your invaluable help during the tape-out of the test chip and all interesting discussions.
- Dr. Stefan Andersson, Dr. Daniel Wiklund and Dr. Jonas Carlsson for all interesting discussions (both on- and off-topic) during late evenings and weekends.
- All my fellow PhD students; Andreas Ehliar, Per Karlström, Di Wu, Johan Eilert, Rizwan Asghar and Ge Qun and all my other co-workers at the Computing Engineering group for your ideas and comments about my work and for a wonderful working environment.
- Ylva Jernling for your invaluable support in everything from travel arrangements to course registrations.
- Anders Nilsson Sr and Niclas Carlén for your practical help with the computers and for your friendship.

- The present and past PhD students and staff at Electronic Systems and Electronic Devices. It has been a great pleasure to spend time with you.
- Greger Karlströms for your practical support and habit of luring me off in the middle of the night at the office and making me take some time off from the hard work, which was needed.
- The staff at Coresonic AB for all support during my time as a PhD student.
- All my other colleagues and friends for cherished friendship.

I would also like to thank Maria Axelsson for her love, support and patience with my working habits during the last year.

Finally I would like to thank my parents, Bo and Hatice Nilsson for their invaluable support, encouragement and love.

This work was supported by the Swedish Foundation for strategic Research (SSF) through the Strategic Integrated Electronic Systems Research center at Linköpings Universitet (STRINGENT).

Anders Nilsson
Linköping, April 2007

Contents

Abbreviations

- AGC: Automatic gain control.
- ASIC: Application specific integrated circuit.
- ASIP: Application specific instruction set processor.
- ADC: Analog to digital converter.
- BBP: Baseband processor.
- CCK: Complementary code keying.
- CDMA: Code division multiple access.
- DAC: Digital to analog converter.
- DSP: Digital signal processing or processor.
- DSSS: Direct sequence spread spectrum.
- DVB-T/H: Digital video broadcasting - Terrestrial / Handheld.
- FDD: Frequency division duplex.
- FEC: Forward error correction.
- FPGA: Field programmable gate array.
- ICI: Inter-carrier interference.
- ISA: Instruction-set architecture.

- ISI: Inter-symbol interference.
- LFSR: Linear feedback shift register.
- LTE: Long term evolution. (“4G”)
- MAC: Media access control or Multiply-accumulate unit.
- MWT: Modified Walsh transform.
- MIPS: Million instructions per second.
- NCO: Numerically controlled oscillator.
- OFDM: Orthogonal frequency division multiplex.
- OVSF: Orthogonal variable spreading factor.
- RC: Raised cosine (filter).
- RF: Radio frequency or register file.
- RMS: Root mean square.
- RRC: Root-raised cosine (filter).
- RTL: Register transfer level.
- TDMA: Time division multiple access.
- SDR: Software defined radio.
- SIMD: Single instruction multiple data.
- SIMT: Single instruction issue, multiple tasks.
- SoC: System-on-chip
- TDD: Time division duplex.
- UMTS: Universal mobile telephony system.
- VLIW: Very long instruction word.
- WCDMA: Wideband CDMA.
- WLAN: Wireless LAN.

Part I

Background

Chapter 1
Introduction

The only source of knowledge is experience.
– Albert Einstein

Baseband processing and baseband processors will become increasingly important in the future when more and more devices will be connected together by means of wireless or wire-line links. Since the number of radio standards grows increasingly fast and the diversity among the standards increases, there is a need for a processing solution capable of handling as many standards as possible and at the same time not consuming more chip area and power than a single-standard product. This trend is driven by the convergence of mobile communication devices. Many modern mobile terminals already support multiple standards such as GSM, WCDMA (3G/UMTS) as well as both bluetooth, GPS and wireless LAN. In a near future digital TV (DVB-H) and the successor to 3G, Long Term Evolution (LTE) will be included in feature rich handsets. In order to achieve the required flexibility to support all these standards and to reach optimal solutions, programmable processors are necessary in contrast to other solutions such as accelerated standard processors and similar devices.

A fully programmable baseband processor enables reuse of computing hardware, not only between different standards but also within them. Programmability can also be used to reduce the Time To Market (TTM) for

a wireless product since the software can be altered after tape-out. In the same way the lifetime of a product can be prolonged since the software stack can be updated or replaced. This allows equipment vendors to reuse existing DSP hardware in new products without having to tape-out new chips.

Programmable baseband processors are also required in order to fulfill the old dream of fully Software Defined Radio (SDR) systems. In the future, SDR systems will most certainly be used to both enable truly world-wide usable products and to efficiently utilize the scarce radio frequency spectrum available, for a wide consumer population. Furthermore existing solutions based on ordinary digital signal processors, do not have the computing power required to perform the computations needed to handle most modern radio standards, and the power consumption of such circuits is high due to their inherent flexibility. To enable programmable baseband processors, we need new processor structures which are optimized for this computing domain but still very flexible within the frame of the same domain.

The goal of this research project has been to create new such power- and area efficient architectures, suitable for future multi-standard radio networks. In this dissertation the results of the research are presented. The results include both a new processor architecture, design methods for this architecture as well as a demonstrator. The results are illustrated with a number of case studies and the fabricated demonstrator.

1.1 Scope of the dissertation

The scope of this dissertation is programmable baseband processors and how they can be designed to achieve a low power consumption and small silicon area. Special attention has been paid to four distinct areas of baseband processor design:

- System architecture, including hardware/software partitioning and instruction set design.

- Selection and design of execution units and accelerators.
- Multi-standard support.
- Scheduling and instruction issue.

The over-all goal has been to find low power and low clock rate processor solutions. General baseband processing includes many tasks such as error control coding/decoding, interleaving, scrambling etc, however this dissertation is focused on the symbol related processing, although the other tasks are also studied regarding acceleration. Symbol related processing is defined as the operations performed between the map/de-map operation and the ADC/DAC in the radio interface. The dissertation presents my research regarding the four areas mentioned above and it also gives an introduction to programmable baseband processing.

It should be noted that baseband processing also exists in many wire-line products such as xDSL and HomePlug. However, this dissertation is focused on wireless products although most of the research results would also be applicable for wire-line communication systems.

1.2 Organization

The thesis is divided into five parts. In Part I system perspective of baseband processing, research motivation and the research methodology are presented.

In Part II the unique properties of baseband processing are discussed in Chapter 5. A methodology for accelerator selection, which was developed in this project, is presented in Chapter 6 and related work is discussed in Chapter 7.

In Part III the SIMT architecture is presented. The SIMT architecture is introduced in Chapter 8. Chapter 9 describes the SIMT design flow. Simultaneous multi-standard execution is described in Chapter 10. Low power design and software development are presented in Chapter 11 and 12 respectively. The fabricated test-chip is presented in Chapter 13 and verification and emulation are presented in Chapter 14.

In Part IV extensions to the SIMT architecture are presented. Chapter 15 describes how MIMO functionality can be mapped to the SIMT architecture and how multiple SIMT processors can be integrated together in a multi-core SoC.

Finally, conclusions are drawn and future work is discussed in Part V.

Chapter 2
System environment

2.1 Introduction

A typical wireless communication system contains several signal processing steps. In addition to the radio front-end, radio systems commonly incorporate two to three different processors as shown in Figure 2.1. The processors are:

- A baseband processor.
- A Media Access Control (MAC) processor.
- An application processor.

The baseband processor is the processor closest to the radio-interface in the processing hierarchy. The baseband processor is responsible for

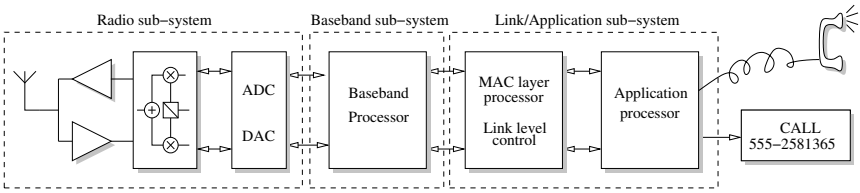

Figure 2.1: Example of a communication system.

modulating the bits received from a higher protocol layer into a discrete waveform, that is sent to the DAC and then transmitted over the air. The baseband processor is also responsible for detecting a received waveform, to synchronize to it and extract information bits from it. These bits are then delivered to the higher protocol layers which assemble the data into user services such as wireless LAN packets or voice packets in a cellular telephone. If the application only requires a smaller amount of control functions, the MAC layer functionality could be merged with the application processor into a single processor. However due to the nature of baseband processing, the baseband processing tasks are usually separated from the application processor.

2.2 Baseband processing tasks

Most wireless systems contain two main computation paths in the baseband processor, the transmit path and the receive path. In the transmit path the baseband processor receives data from the MAC processor and performs

- Channel coding
- Modulation
- Symbol shaping

before the data is sent to the radio front-end via a DAC. In the receive path, the RF signal is first down-converted to an analog baseband signal. The signal is then conditioned and filtered in the analog baseband circuitry. After this, the signal is digitized by an ADC and sent to the digital baseband processor which performs

- Filtering, synchronization and gain-control
- Demodulation, channel estimation and compensation
- Forward error correction

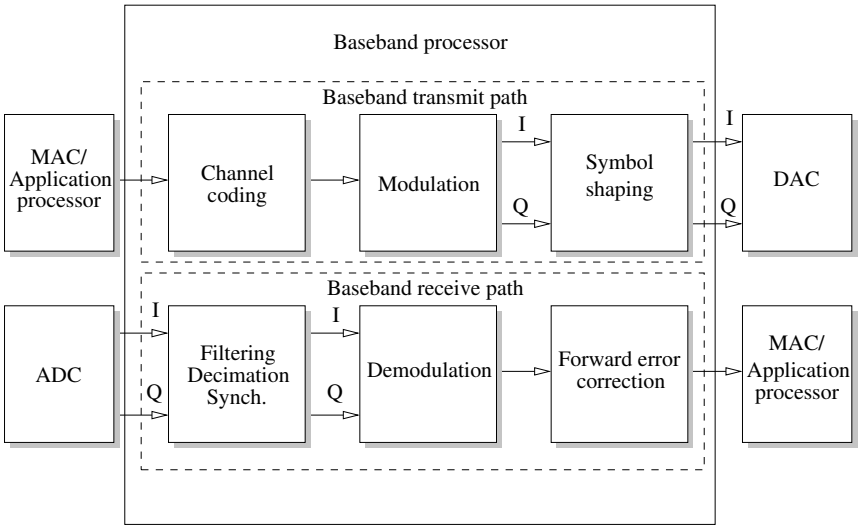

Figure 2.2: Tasks of a baseband processor.

before the data is transfered to the MAC protocol layer.

These processing tasks are illustrated in Figure 2.2. In general, this schematic view of baseband processing tasks is true for most radio systems. In the transmit path from the MAC layer to the radio, air interface data are first sent to a channel coding step which adds redundancy to the transmitted data, interleaves data both in time and in frequency and scrambles the data to remove regularities in the data stream. The binary data are fed to the modulator stage which converts it into one or many *symbols*. A symbol can be represented as one or a series of complex numbers representing a waveform or a single value. This stream of complex numbers is then sent to the symbol shaping stage which filters and smooths the signal in order to remove unwanted spectral components.

At the receiver side, all the operations are performed in reverse. The stream of complex data values from the ADC is first fed to a digital front-end stage which contains a digital filter, a sample-rate converter and correction logic for RF front-end impairments such as I/Q-mismatch, DC-

offset etc. The filtered sample stream from the digital front-end is then fed into the demodulator which performs synchronization, symbol demodulation and channel compensation on the received symbols. Binary data are then extracted from the received symbols and fed to the forward error correction unit, which utilizes the redundancy added by the channel coder stage in the transmitter to correct for any encountered transmission errors.

Chapter 3
Motivation

The best way to predict the future is to invent it.
– Alan Kay

3.1 Introduction

The idea of software defined radio has thrilled many people over the last decades. SDR promotes flexibility and hardware reuse which makes it very appealing for companies seeking to implement flexible multi-standard radios in the future. However, the limited computing capacity of ordinary DSPs has not been able to close the computing capacity gap between fixed function circuitry and standard DSPs, thus prohibiting widespread use of SDR technology. Efficient yet flexible baseband processor platforms are the key to enable true SDR systems.

3.2 Software Defined Radio

To ensure a common view of the term "Software Defined Radio", the SDR Forum [1] has defined it by use of tiers to describe the various capabilities. Each tier refers to a higher level of capability and flexibility described by its number.

The SDR Forum defines five tiers of radio solutions:

Tier 0 Hardware Radio (HR): The radio is implemented using hardware components only and cannot be modified except through physical intervention.

Tier 1 Software Controlled Radio (SCR): Only the control functions of an SCR are implemented in software, thus only limited functions are changeable using software. Typically this extends to interconnects, power levels etc. but not to frequency bands and/or modulation types etc.

Tier 2 Software Defined Radio (SDR): SDRs provide software control of a variety of modulation techniques, wide-band or narrow-band operation, communication security functions, and waveform requirements of current and evolving standards over a broad frequency range. The frequency bands covered may still be constrained at the front-end requiring a switch in the antenna system.

Tier 3 Ideal Software Radio (ISR): ISRs provide dramatic improvement over an SDR by eliminating the analog amplification or heterodyne mixing prior to digital-analog conversion. Programmability extends to the entire system with analog conversion only at the antenna, speaker and microphones.

Tier 4 Ultimate Software Radio (USR): USRs are defined for comparison purposes only. It accepts fully programmable traffic and control information and supports a broad range of frequencies, air-interfaces and application software.

Programmable baseband processors implementing SDR are necessary to enable efficient flexible multi-standard radio systems. They will also help system developers and manufacturers to reduce the Bill Of Materials (BOM) and enhance the lifetime of products by allowing software upgrades. This chapter highlights important aspects and benefits, both technical and economical, of using programmable baseband processors.

3.3 Technical aspects

The most important benefits of using programmable baseband processors are listed below. Programmable baseband processors allow:

- Hardware and software reuse within and between multiple radio standards.
- Dynamic resource allocation.
- Multi-standard support (both separate and simultaneous) within the same processor.
- Possibility to perform system updates and bug-fixes while the system is in operation.

3.3.1 Hardware and software reuse

One of the greatest benefits of programmable baseband processors is the opportunity for hardware and software reuse. Reuse can be applied on many levels:

- Reuse of computing hardware and the associated software within a standard will reduce the amount of hardware needed to implement support of a standard – hence reducing the circuit area. This is often referred to as "Hardware multiplexing".
- By reusing the same hardware and software kernels between different standards, the amount of hardware and especially the amount of program memory in the processor will me minimized.
- Reuse of hardware and software between projects will save valuable development time, reduce development costs and ensure benefits in terms of time to market.

In fixed function solutions, processing resources cannot normally be efficiently shared between different operations. However, in a programmable multi-standard baseband processor, hardware multiplexing can often achieve

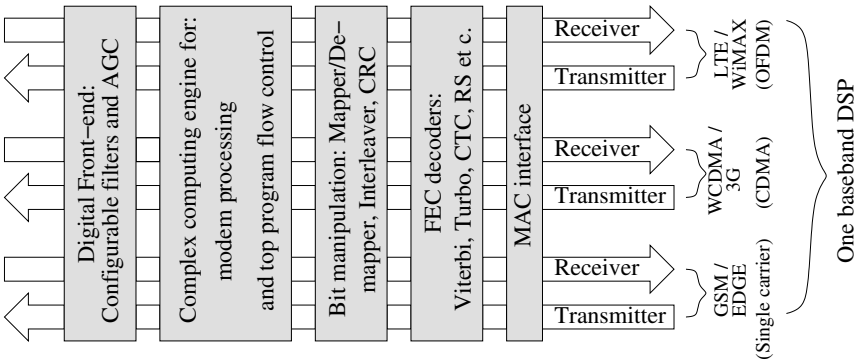

Figure 3.1: Example of hardware multiplexing of three different modulation schemes on a programmable baseband processor.

comparable power consumption and lower silicon area than a fixed function circuit [2]. This illustrates that programmability does not necessarily lead to larger hardware or higher power consumption. The concept of hardware multiplexing is illustrated in Figure 3.1.

As an example, a high-end mobile device can have support of a large number of different wireless standards from three groups of wireless standards with different properties:

Communication: The communication group includes cellular protocols such as GSM, 3G, or LTE and is used to provide packet based or circuit switched network access mainly for voice and packet data services with demanding latency requirements.

Connectivity: The connectivity group includes standards used for "peripheral connectivity" such as Bluetooth, Wireless LAN or WiMAX access. The performance requirements of these standards are often not as strict as in the communication group in terms of latency. They are further characterized by their high bandwidth.

Entertainment The entertainment class includes standards such as the mobile TV standards DVB-H and DMB and digital radio standards such as DAB and satellite radio. Only receiver functionality is needed.

From these three groups of communication standards it can be concluded that a high-end mobile terminal will have to support eight to ten different standards of which two to three will operate at the same time. The classical approach to design multi-mode systems by integrating many separate baseband processing modules, each module covering one standard in order to support multiple modes will give prohibitively large and rigid solutions for future multi-standard mobile devices.

However, by using programmable baseband processors instead of fixed function hardware all standards can be implemented on the same hardware and reuse software kernels, thus saving program memory and chip area. Through hardware reuse we can reach a smaller silicon area than a fixed function (ASIC) solution! [3]

3.3.2 Dynamic resource allocation

Another feature of programmable baseband processors is the ability to use dynamic resource allocation at runtime. By dynamically redistributing available resources, the focus can either be on mobility management or high data rate.

In Figure 3.2, the MIPS floor is limited by the top edge of the triangle. During severe fading conditions, the processor runs advanced channel tracking and compensation algorithms to provide reliable communication. In good channel conditions more computing resources can be allocated to symbol processing tasks to increase the throughput of the system. Dynamic resource allocation can also be used to reduce the power consumption in a system operating below its maximum capacity.

3.4 Market aspects

In addition to the previously presented technical reasons for using programmable baseband processors in modern wireless systems, there are also some marketing and economical reasons. Programmable processors provide:

- Prolonged product lifetime due to re-programmability.
- Reduced cost of product ownership due to the extensive possibilities for reuse.
- Reduced Non-Recurring Engineering (NRE) costs.
- Possibility of product customization after tape out.

SDR technology promotes "platform products", e.g. a fixed hardware platform capable of performing many different tasks. By using platform products, the product maintenance and personnel training costs could be shared by a number of projects.

3.4.1 Implementation flexibility

The flexibility gained by programmability can be used to "customize" platform products after tape out or be used to implement late changes to a volatile wireless standard such as IEEE 802.11n.

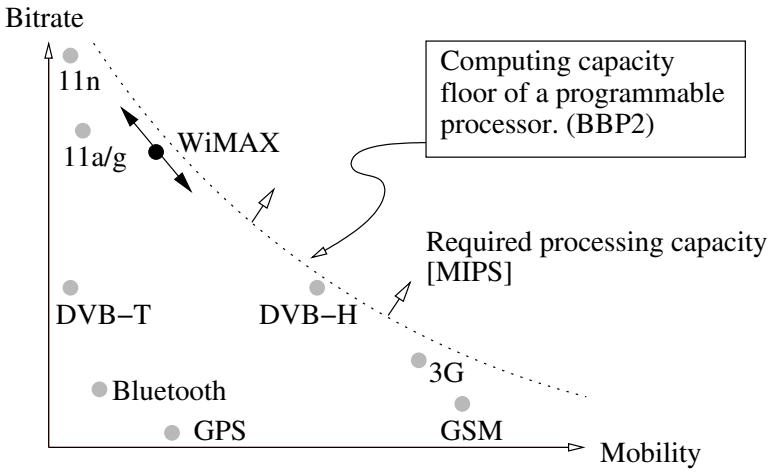

Figure 3.2: Dynamic resource allocation. High mobility requires advanced channel estimation and compensation algorithms which increases the number of operations needed per bit of received data.

There are many uncertainties in future wireless standards:

- Which standards will be used in the future?
- How will the standards evolve?
- What standards will coexist in a future product?

All the uncertainties mentioned above make it very risky to start an expensive ASIC project for a future product. By instead using a flexible solution, the decision of what features should be implemented in the product can be made much later in the development project – even after tape-out.

Product lifetimes of systems implemented using programmable processors will also be prolonged since an "old" programmable processor can still be used to new standards which were not available when the processor was designed provided that the processor has enough computing resources.

As the mask-set cost for modern CMOS processes skyrockets, the ability to fix bugs and late design changes without a re-spin is extremely valuable and further promotes the use of programmable baseband processors.

3.5 Military SDR - JTRS

A completely different segment of the SDR community is the military segment which has quite different needs. Instead of focusing strictly on low cost and bandwidth, the military community requires extreme reliability and compatibility between different platforms.

Software defined radios will be used in many military systems to both provide new features and, more important, to provide a radio platform which can communicate with all legacy military radio systems present around the world. This is important as warfare has gone from a single armed force to a coalition with armed forces from many countries - all with their own communication systems.

The American armed forces have initiated an effort to use an SDR system named "Joint Tactical Radio System" (JTRS) [4]. The JTRS effort is aimed at providing "off-the-shelf" military radio products, thus reducing overall system costs for the armed forces.

JTRS compliant radio platforms implement "waveforms". A waveform is a complete functional specification of everything between the user interface and the antenna. The basic idea of JTRS is to implement hardware independent waveforms for various communication needs. The waveform is built around the Software Communications Architecture (SCA) which specifies how software and hardware interact in a system.

There are currently nine specified waveforms implementing a diverse set of military communication systems, everything from legacy systems to modern wideband communication systems. The current waveforms are:

- Wideband Networking Waveform (WNW)
- Soldier Radio Waveform (SRW)
- Joint Airborne Networking - Tactical Edge (JAN-TE)
- Mobile User Objective System (MUOS)
- SINCGARS
- Link-16
- EPLRS
- High Frequency (HF)
- UHF SATCOM

JTRS is of interest even for civilian users since it is a very large collective effort aiming at a common goal. The development carried out in the SDR area by the JTRS effort will be of great importance to the SDR community in whole.

3.6 Bridging the computing complexity gap

As the computing capacity required to manage common wireless standards is growing with each generation of standards, traditional processors and DSPs cannot be used to implement SDR systems. A quantitative illustration of the computing complexity of common wireless standards is shown in Figure 3.3 [5]. The gap between the computing power of ASIC technology and DSPs must be bridged.

The extreme demands on computing capacity and at the same time low power consumption call for new architectures that allow an increase in the processing parallelism. High parallelism allows the processors to work at a low frequency, thus consuming lower power while providing enough MIPS. The true challenge is to provide this tremendous computing power at the same cost in terms of power consumption as a fixed function ASIC while maintaining the flexibility of a processor.

The key idea is to utilize knowledge from all the wireless standards known today and try to anticipate what new features will be required in the future and create an Application Specific Instruction set Processor (ASIP) architecture based on this knowledge.

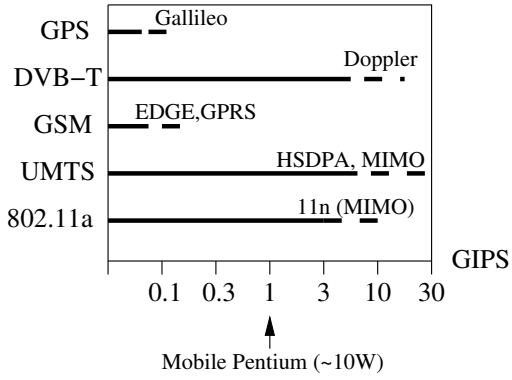

Figure 3.3: SDR Complexity. Data from Kees van Berkel et al.

ASIPs allow the processor architecture to be optimized towards a quite general application area such as baseband processing. By restricting the processor architecture, several application specific optimizations can be made. By that definition a baseband DSP capable of processing millions of symbols per second might not be able to encode an MPEG4 video stream.

3.7 Summary of challenges

Many challenges are faced during the design of programmable baseband processors – both academical and engineering challenges. A short summary of the challenges faced is presented below:

- Providing extreme computing performance with flexibility while not consuming more power than a traditional fixed-function ASIC solution.
- Having flexibility to cover current and future standards.
- Providing low computing latency.
- Interesting academic challenges such as investigation and exploration of:
  - Software and hardware co-design.
  - Hardware multiplexing.
  - Architecture space exploration.
  - Constraint aware profiling, i.e. profiling with e.g. system latency constraints in mind.

3.8 References

[1] The Software Defined Radio Forum official web site; `http://www.sdrforum.org`

[2] Eric Tell, Anders Nilsson, and Dake Liu; *A Low Area and Low Power Programmable Baseband Processor Architecture*; Proc of the International workshop on SoC for real-time applications, Banff, Canada, July 2005

[3] Anders Nilsson and Dake Liu; *Area efficient fully programmable baseband processors*; In proceedings of International Symposium on Systems, Architectures, Modeling and Simulation (SAMOS) VII Workshop; July 2007, Samos, Greece.

[4] Joint Tactical Radio System official web site; `http://jtrs.army.mil`

[5] Kees van Berkel; *Vector Processing as an Enabler for Software Defined Radio in Handheld Devices*; ISSCC07 Signal Processing Tutorial, 2007

Chapter 4
Research methodology

If we knew what it was we were doing, it would not be called research, would it? – Albert Einstein

This chapter briefly describes the research methodology used in this research project. The research methodology can be summarized by the following steps:

1. Study and analysis of general baseband processing and the related technology.
2. Survey of other research projects and related work within the same field.
3. Formalization of project goals and scope.
4. Formulation of a design and evaluation methodology for programmable baseband processors.
5. Architecture space exploration.
6. Development and refinement of a processor architecture according to the previously mentioned methodology.
7. Implementation of the created processor.
8. Evaluation.

It is very important to consider real-world effects in all stages of the research to ensure that the research results have practical relevance in the real world.

To ensure accurate research results recorded air-data have been used where available. By using real recorded data, the validity of channel models and selected algorithms used to guide the architecture design has been proven and the effects caused by a non-ideal radio have been accounted for.

Further details of how to model, design and evaluate baseband processors are presented in Chapter 9. As with any research project, many iterations are performed during the research work. However, it is important to remember the research methodology in order to ensure a high quality of the research results and findings. During the initial phases of a research project, both the scope and goal of the project will most certainly change. However, at some time early in the project a decision must be made, clearly defining what to include and what to leave out. This is essential to keep the project going, since there is no limit on how much time a researcher can spend on an interesting problem.

Part II

Programmable baseband processors

Chapter 5
Baseband signal processing

5.1 Introduction

In this chapter both properties of baseband signal processing and some of the challenges faced in a baseband processing system are described. It is only by identifying and utilizing common operations and properties of baseband processing problems that efficient programmable baseband processors can be designed. At the same time, the understanding of "real world" problems ensures relevant research and results.

5.2 Challenges

The four most demanding challenges for the baseband processor to manage, regardless of modulation methods and standards, are:

- Multi-path propagation and fading. (Inter-symbol interference.)
- High mobility.
- Frequency and timing offsets.
- Radio impairments.

These four challenges impose a heavy computational load for the processor. Besides the challenges mentioned above, baseband processing in general also faces the following two challenges:

- High dynamic range.
- Limited computing time.

To create a practically useful processor architecture all these challenges must be considered.

5.2.1 Multi-path propagation and fading

In a wireless system data are transported between the transmitter and receiver through the air and are affected by the surrounding environment. One of the greatest challenges in wide-band radio links is the problem of multi-path propagation and inter-symbol interference. Multi-path propagation occurs when there are more than one propagation path from the transmitter to the receiver. Since all the delayed multi-path signal components will add in the receiver, inter-symbol interference will be created. As the phases of the received signals depend on the environment, some frequencies will add constructively and some destructively, thus destroying the original signal. Unless the transmitter and receiver sit within an echo-free room or any other artificial environment, the transmission will usually be subjected to multi-path propagation. There is only one common communication channel which is usually considered echo-free namely a satellite link. Multi-path propagation is illustrated in Figure 5.1.

Multi-path propagation can be characterized by the channel impulse response. From the channel impulse response several important parameters can be derived. The most important parameter is the RMS *delay-spread*, σ_τ, which describes the RMS distance in time between multi-path components. The channel impulse response, also known as the Power Delay Profile (PDF) of a channel is illustrated in Figure 5.2.

The delay-spread imposes a limit on the shortest usable symbol period. This will in turn restrict the data-rate of a transmission system. A rule of thumb is to use symbol durations which are at least 10 times the delay spread ($10 \cdot \sigma_\tau$) if the system operates without advanced channel equalizers.

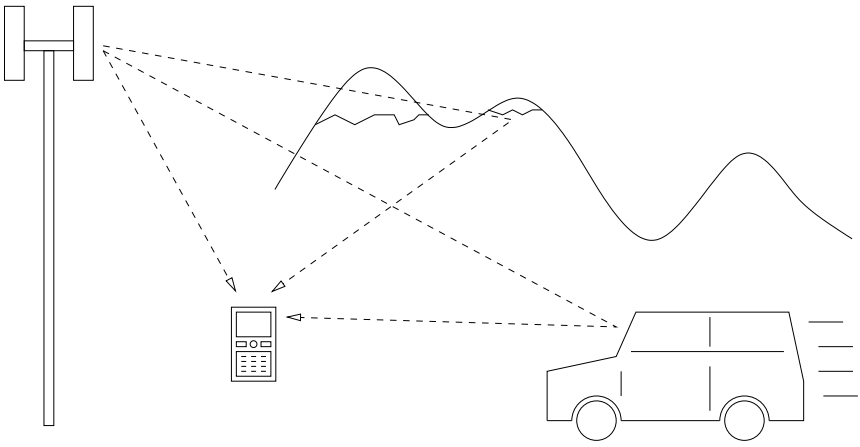

Figure 5.1: Multi-path propagation.

In the example in Figure 5.2 the shortest symbol duration would be limited to 420 ns, which gives a symbol rate of 2381 symbols$/\mathrm{s}$. In this example, a delay-spread of 42 ns was used, which corresponds to a maximum path-distance of about 12 meters. In outdoor systems the delay-spread is often in the range of several micro-seconds, which further limits the symbol rate. Common reference "channels" are specified by various institutes and standardization organs to be used in benchmarking of channel equalizers. In Table 5.1, two common channels are presented, the ITU Pedestrian A and ITU Vehicular Channel Model B [1]. The channel models represent a user walking 3 $\mathrm{km/h}$ in an urban environment and traveling in a car at 60 $\mathrm{km/h}$ respectively. The channel models specify a number of multi-path components (taps) with their delay and average power. The phase and amplitude of the multi-path component are assumed to be Rayleigh distributed.

The effects of multi-path-propagation and resulting inter-symbol interference are referred to as *fading*. For narrow-band systems (with long symbols), the effect of inter-symbol interference can be assumed to be constant over the entire frequency spectrum of the used channel (flat fading). However, in wide-band systems the effects of inter-symbol interference

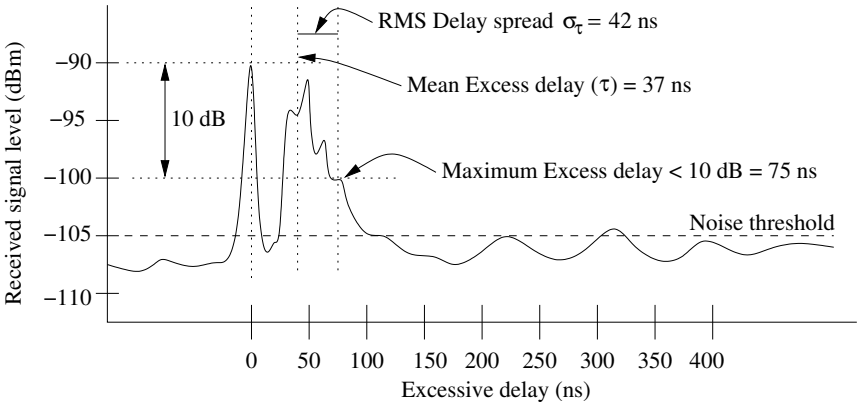

Figure 5.2: Power Delay Profile

will cause frequency dependent fading, causing parts of the transmitted signal spectrum to be destroyed. To mitigate frequency selective fading in wide-band systems, advanced equalizers must be used to compensate for multi-path channels. Another solution to avoid the problem of wide-band channels in multi-path environments is to divide the wide-band channel into many narrow-band channels and treat them as flat faded channels. This is the basic principle of OFDM transmission systems [3]. However, since it is not possible to use OFDM technology in all situations, advanced equalizers must be employed for example in CDMA networks.

5.2.2 Dynamic range

Another problem faced in practical systems is the large dynamic range of received signals. Both fading and other transmitting radio equipment in the surroundings will increase the dynamic range of the signals arriving in the radio front-end. It is common with a requirement of 60-100 dB dynamic range handling capability in the radio front-end [7]. Since it is not practical to design systems with such large dynamic range, Automatic Gain Control (AGC) circuits are used. This implies that the processor measures the received signal energy and adjusts the gain of the ana-

Table 5.1: Power delay Profiles for common channel models

	ITU Pedestrian A		ITU Vehicle B	
Tap	**Delay (ns)**	**Average power (dB)**	**Delay (ns)**	**Average power (dB)**
1	0	0	0	-2.5
2	110	-9.7	300	0
3	190	-19.2	8900	-12.8
4	410	-22.8	12900	-10.0
5			17100	-25.2
6			20000	-16.0

log front-end components to normalize the energy received in the ADC. Since signals falling outside the useful range of the ADC cannot be used by the baseband processor, it is essential for the processor to continuously monitor the signal level and adjust the gain accordingly. The power consumption and cost of the system can be further decreased by reducing the dynamic range of the ADC and DAC as well as the internal dynamic range of the number representation in the DSP processor. By using smart algorithms for gain-control, range margins in the processing chain can be decreased.

5.2.3 Mobility

Normally, the channel is assumed to be time invariant. However, if the transmitter or receiver moves, the channel and its fading will be time varying. Mobility in a wireless transmission causes several different effects, the most demanding effect to manage is the rate at which the channel changes. If the mobility is low, e.g. when the channel can be assumed to be stationary for the duration of a complete symbol or data packet, the channel can be estimated by means of a preamble or similar known sequence. However, if mobility is so high that the channel changes are

significant during a symbol period, this phenomenon is called *fast fading*. Fast fading requires the processor to track and recalculate the channel estimation during reception of user payload data. Hence, it is not enough to rely on an initial channel estimation performed on a packet or frame start.

Mobility can be described by the *channel coherence time*, T_c, which is inversely proportional to the maximum Doppler shift of the channel. For example, a WCDMA telephone operating at 2140 MHz will encounter a Doppler shift of 118 Hz when the telephone travels at 60 $\mathrm{km/h}$ towards the base-station. For a correlation of 0.5 of the current channel parameters and the channel parameters after the time T_c, the following formula applies [2]:

$$T_c = \sqrt{\frac{9}{16\pi f_m^2}} \tag{5.1}$$

where f_m is the maximum Doppler shift of the channel. This yields the channel coherence time of the previous example to be 3.5 ms. For WCDMA, there are modes specified for up to 250 $\mathrm{km/h}$, corresponding to a Doppler shift of 492 Hz and a channel coherence time of $T_c = 860$ $\mu\mathrm{s}$.

At 250 $\mathrm{km/h}$ the coherence time of the channel is roughly in the same order as the slot-time, which implies that the processor must track channel changes during the reception of a data slot. The Doppler shift will also create the same effects as frequency offsets. However, the effects of frequency offsets can easily be compensated for by de-rotating the received data [3].

Mobility together with a multi-path channel will also cause *doppler spread*. Doppler spread is caused by different echos having different doppler shifts, thus widening the spectrum. Estimation and mitigation of doppler spread is especially important in mobile OFDM systems where doppler shift will cause Inter Carrier Interference (ICI) and degrade system performance [4]. ICI mitigation is further discussed in [5, 6].

5.2.4 Radio impairments

Along with distortion and other effects added by the channel, the transmission system can in itself also have impairments. Such impairments could be:

- Carrier frequency offset (CFO).
- Sample frequency offset.
- DC-offset.
- I/Q non-orthogonality.
- I/Q gain mismatch.
- Non-linearities.

The items listed above are all common impairments in radio-front ends and they affect the performance of the whole system. Since the correction of the impairments requires extra computing resources, it is essential to include the correction of the impairments as early as possible in the design of programmable baseband processors.

One common problem in direct up-conversion transmitters is the problem of non-orthogonality of the I and Q baseband branches. Often, the quadrature-phase carrier is created by delaying the in-phase carrier to create a 90° phase-shift. However, this phase-shift will be frequency dependent and only provide 90° phase-shift at one frequency. Non-orthogonality will create severe problems for QAM modulations of high order, and create unwanted AM modulation of constant-envelope modulation schemes as well as inter-carrier interference. An expression of the received signal (I',Q') with a gain mismatch of δ, a CFO of w rad/s, a DC-offset of (I_{dc},Q_{dc}) and a leakage (non-orthogonality) of ϵ is described in Equation 5.2.

$$\begin{bmatrix} I' \\ Q' \end{bmatrix} = \begin{bmatrix} cos\,\omega t & -sin\,\omega t \\ sin\,\omega t & cos\,\omega t \end{bmatrix} \cdot \begin{bmatrix} 1+\delta & \epsilon \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} I \\ Q \end{bmatrix} + \begin{bmatrix} I_{dc} \\ Q_{dc} \end{bmatrix} \quad (5.2)$$

Further radio impairments are discussed in the book [7].

5.2.5 Processing capacity challenges

Since baseband processing is a strict hard real-time procedure, all processing tasks must be completed on time. This imposes a heavy work-load on the processor during computationally demanding tasks such as Viterbi-decoding, channel estimation and gain control calculations. In a packet based system, the channel estimation, frequency error correction and gain control functions must be performed before any data can be received.

This may result in an over-dimensioned processor, since the processor must be able to handle the peak work load, even though it may only occur less than one percent of the time. Here, in this case, programmable DSPs have an advantage over fixed function hardware since the programmable DSP can rearrange its computing resources to make use of the high available computing capacity all the time.

5.3 Modulation methods

To further understand the challenges faced in processing of different wireless standards some background information is given about each modulation scheme. Most wireless standards are based on one of the following modulation schemes:

- Single Carrier modulation
- OFDM
- CDMA

or combinations of all of them.

5.3.1 Single Carrier

The two most common single carrier modulation schemes used today are the Gaussian Minimum Shift Keying (GMSK) and 8-level Phase Shift Keying (8-PSK) modulation which is used in GSM/EDGE, bluetooth and

DECT. Common to the two modulation methods is their constant envelope which relaxes the design of power amplifiers and receivers.

Other modulation methods such as QAM can also be used "directly" in a single carrier system. Classical AM and FM radio also fall in this category.

5.3.2 OFDM

As channel equalization in single carrier systems becomes more complex when the channel bandwidth grows, several other modulation methods have been developed to mitigate this problem. One such method is OFDM modulation.

Orthogonal Frequency Division Multiplexing (OFDM) is a method which transmits data simultaneously over several sub-carrier frequencies. The name comes from the fact that all sub-carrier frequencies are mutually orthogonal, thereby signaling on one frequency is not visible on any other sub-carrier frequency. This orthogonality can be implemented by collecting the symbols to be transmitted on each sub-carrier in the frequency domain, and then simultaneously translating all of them into one time domain symbol using an Inverse Fast Fourier Transform (IFFT).

The advantage of OFDM is that each sub-carrier only occupies a narrow frequency band and hence can be considered to be subject to flat fading. Thus a complex channel equalizer can be avoided. Instead the impact of the channel on each sub-carrier can be compensated by a simple multiplication in order to scale and rotate the constellation points to the correct position once the signal has been transfered back to the frequency domain (using the fast Fourier transform) in the receiver.

To further reduce the impact of multi-path propagation and Inter Symbol Interference (ISI), a guard period is often created between OFDM symbols by adding a Cyclic Prefix (CP) to the beginning of each symbol. This is achieved by simply copying the end of the symbol and add it in front of the symbol. As long as the channel delay spread is shorter than the cyclic prefix, the effects of ISI are mitigated.

5.3.3 CDMA

Code Division Multiple Access (CDMA) is a multiple access scheme which allows concurrent transmission in the same spectrum by using orthogonal *spreading codes* for each communication channel. In this section the two CDMA based standards: Wideband CDMA (WCDMA) and High Speed Data Packet Access (HSDPA) are used as examples.

In a CDMA transmitter, binary data are mapped onto complex valued symbols which then are multiplied (spread) with a code from a set of orthogonal codes. The length of the spreading code is called the *spreading factor* (SF). In the receiver data are recovered by calculating a dot-product (de-spread) between the received data and the assigned spreading code. Since the spreading codes are selected from an orthogonal set of codes, the dot-product will be zero for all other codes except the assigned code. By varying the spreading factor, the system can trade data rate against SNR as a higher SF increases the energy per symbol.

A feature of WCDMA is the ability to scale the bandwidth for a particular user by assigning multiple spreading codes to that user. Using multiple codes is referred to as multi-code transmission. Multi-code transmission can also be used for *soft hand-over*, e.g. when the mobile station is handed over between two or more base-stations. By using one or many codes from each of the involved base-stations, the mobile station can be handed over without any interruption of the service. The WCDMA standard requires the mobile station to manage up to 3 simultaneous spreading codes, each requiring one rake finger, on 6 base stations (18 codes in total).

5.4 Baseband processing properties

By observing and benchmarking baseband algorithms in the modem stage of a baseband processor, four interesting properties can be observed:

- Complex computing: Most computations are performed on complex valued data.

- Vector property: A large portion of the computations is performed on long *vectors* of data.
- Control flow: The control-flow is predictable to a large extent.
- Streaming data: The processor operates on streaming data.

These four properties should be considered when designing an baseband processor architecture to ensure its efficiency.

5.4.1 Complex computing

A very large part of the processing, including FFTs, frequency/timing offset estimation, synchronization, and channel estimation employs well known convolution based functions common in DSP processing. Such operations can typically be carried out efficiently by DSP processors thanks to Multiply-Accumulate (MAC) units and optimized memory- and bus architectures and addressing modes. However, in baseband processing essentially all these operations are complex valued. Therefore it is essential that also complex-valued operations can be carried out efficiently. To reach the best efficency, complex computing should be supported throughout the architecture: by data paths and instruction set as well as by the memory architecture and data types.

5.4.2 Vector property and control flow

Further analysis of baseband processing tasks, especially in OFDM and CDMA transceivers reveals that most baseband processing jobs are dominated by operations on vectors of data (such as convolution, FFT, despread etc.). Furthermore there is often no or little backward dependency between the vector operations.

Two representative examples are the 8192 point FFT or the 6144 point complex multiplication performed during reception and channel equalization in Digital Video Broadcasting.

The vector property and the predictable control flow can be used to enable efficient computations in SIMD data-paths and reduced complexity of the control-path. The vector property also allows the creation of software pipelines with vector operations thus increasing the processing parallelism. By improving the computing efficiency for vector operations, valuable clock cycles could be gained and used by the controller core for control flow functions and supervision tasks.

5.5 References

[1] ITU ITU-R M.1225, Guidelines for evaluations of radio transmission technologies for IMT-2000, 1997.

[2] Rappaport, T.S.; *Wireless Communications Principles and Practice*; Second Edition 2002, Prentice Hall PTR, Upper Saddle River, NJ.

[3] H Heiskala, J T Terry, *OFDM Wireless LANs: A Theoretical and practical guide*; Sams Publishing, 2002

[4] S. Tomasin, A. Gorokhov, H. Yang and J. Linnartz; *Iterative Interference Cancellation and Channel Estimation for Mobile OFDM*; IEEE Transactions on Wireless communications, Vol 4 No 1, January 2005

[5] W.G. Jeon, K.H. Chang and Y.S. Cho;*'An equalization technique for Orthogonal Frequency-Division Multiplexing in Time-variant multipath channels*; IEEE Tr. on Comm. Vol. 47, No. 1, Jan. 1999, pp. 27-32.

[6] Y. Li, L.J. Cimini, and N.R. Sollenberger; *Robust channel estimation for OFDM systems with rapid dispersive fading channels*; IEEE, Tr. on Comm. vol. 46, pp. 902-915, July 1998.

[7] Bosco Leung, Behzad Razavi, *RF Microelectronics*; Prentice Hall PTR, ISBN 0-13-887571-5, 1998

Chapter 6
Acceleration

6.1 Introduction

As described in earlier chapters, modern wide-band modulation schemes require large computing capacity, ruling out regular implementations in general purpose processors. The key to further increased processing capacity and still maintained power and area efficiency is to introduce acceleration in the processor. An accelerator is extra hardware added to a programmable processor, which performs a certain class of pre-configured tasks while the processor could perform other operations. Acceleration could also be used to extend the functionality of single instructions of the processor. However, every extra accelerated function will increase the hardware cost, thus it is essential to select the right accelerators to cover most processing needs over the multiple standards. Acceleration can be performed on many different levels and on different types of functions:

- Function level acceleration.
- Instruction level acceleration.
  - Complex instructions.
  - Addressing support.

6.1.1 Function level acceleration

Function level acceleration accelerates a complete function such as a Viterbi-decoder, front-end filter or a complete fast fourier transform. This kind of acceleration is usually implemented as a stand-alone *accelerator*. In this case, an accelerator is defined as an execution unit connected to the internal communication network of the processor, which can perform processing tasks independently of the processor core. Function level accelerators are normally controlled via control-registers from the controller core.

Implementation of function level accelerators can be divided into two main categories. The first category includes more or less configurable hardware units such as filters, FEC decoders etc. The other category is accelerator blocks based on reconfigurable hardware such as embedded FPGAs [1] (i.e. fine grained reconfigurability). Architectures using accelerators from the later category provide more flexibility than configurable units at the expense of configuration overhead both in terms of larger silicon area of the accelerator, long reconfiguration time and high memory usage. During the remainder of this chapter, configurable accelerators are assumed. The use of FPGA technology is further discussed in Chapter 7.

6.1.2 Instruction level acceleration

Instruction level acceleration is the other form of acceleration employed by the SIMT baseband processor architecture. Instruction level acceleration implies that the processor is equipped with powerful special instructions, which accelerate a special task. This could for example be an FFT butterfly instruction, which performs one FFT butterfly operation in a single clock cycle. Other instruction level accelerators could be addressing support for different cycle consuming addressing modes such as modulo, bit-reversed and rake finger addressing or bit manipulation support in the controller core.

6.2 Accelerator selection method

Since accelerators add extra hardware and complexity to the design, it is also important that the choice of what to accelerate is made right.

In order to make good design choices and achieve an optimal design, a method to benchmark the selected algorithms and a method to decide what to accelerate are needed. In order to compare the computational load of a certain algorithm, a "MIPS cost" has been proposed and used in this project. The MIPS cost corresponds to how many million instructions-per-second a regular DSP processor would require in order to perform the specified function. The MIPS cost is calculated as follows:

$$MIPS_i = \frac{OP_i \cdot N_i}{t_i} \tag{6.1}$$

where $MIPS_i$ is the associated MIPS cost, OP_i is the number of clock cycles required by a standard DSP processor to perform the operation, N_i is the number of samples or bits to process and t_i is the maximum time allowed for the operation to complete. Unless stated otherwise, a single MAC DSP processor (with address generation support) is assumed when MIPS-costs are discussed. Furthermore only kernel operations are included in the MIPS cost which result in under-estimation of the required MIPS compared to actual DSP implementations. For symbol related operations the time to perform the operation is considered to be the symbol duration. Here it is very important that the scheduling of the tasks is performed correctly since the latency requirements are important to consider.

By analyzing all standards in terms of processing operations and kernel functions, similar operations in different standards can be identified. These operations are then used as acceleration candidates. When the MIPS cost has been calculated for all kernel functions; silicon usage must be estimated for all functions considered for acceleration.

Finally, when all kernel functions have been identified and their associated silicon area has been estimated the following three points must be considered:

1. **MIPS-cost.** A function with a very high MIPS cost must be accelerated since the operation cannot be performed by a regular processor.

2. **Reuse.** A function that is performed regularly and is used by several radio standards is a good candidate for acceleration.

3. **Circuit area.** Acceleration of special functions is only justified if there can be considerable reduction of clock frequency or power compared to the extra area added by the accelerator.

Any kernel function or operation fulfilling one or more of the previous points is a good candidate for hardware acceleration [2].

6.3 Configurability and flexibility

Since it is desirable that the chosen accelerators can be used by as many standards as possible, flexibility and especially configurability of the accelerators are important. However, as increased flexibility can increase area and power consumption it is important to analyze different degrees of configurability of the unit. Examples of flexibility and configurability of accelerators are:

- Selectable decimation ratio in an accelerated digital front-end.
- Configurable filter coefficients in the same front-end.
- Flexible polynomial in bit manipulation units based on Linear Feedback Shift Register (LFSR) structures.
- Interleavers with flexible interleaving depth and patterns.
- Map/de-map unit with variable gain and configurable constellation diagram.

The same method that is used to identify which functions to accelerate can be used in the accelerator design to find the optimal balance between circuit performance and flexibility.

6.4 Accelerator integration

The integration of accelerators is the key to maintain the advantage created by the accelerator. Function level accelerators should be connected to the core communication bus used in the processor and have the same read/write access to memories as ordinary execution units to ensure high computing throughput. The difference between a function level accelerator and an execution unit is the control method. No instructions are issued to function level accelerators, instead the accelerators are configured via a control-register interface mapped into the control register space of the controller core. Each accelerator unit can generate an interrupt signal to the controller core to signal its status or the completion of a task. The integration of function level accelerators in the SIMT architecture is further described in Section 8.8.

6.5 Case study: Acceleration in multi-standard modems

6.5.1 Introduction

Acceleration of common cycle-consuming DSP jobs is necessary in order to efficiently manage wide-band modulation schemes. In this case study it is investigated which jobs are suitable for acceleration in a programmable baseband processor supporting a common Wireless LAN [3, 4] and 3G [5] standards. The considered standards are listed in Table 6.1. By using three diverse standards in this case study, it is shown that acceleration can be successfully used in multi-standard modems.

To be able to decide which functions to accelerate the different communication standards listed previously are analyzed focusing on sample rates, required functionality of a baseband processor and cycle cost/latency requirements. To ensure fair data for decision making, the most demanding mode of each standard has been used in the calculations.

Table 6.1: Standard overview

Standard	Modulation	Type
WCDMA	CDMA	3G
IEEE 802.11b	DSSS/CCK	Wireless LAN
IEEE 802.11a	OFDM	Wireless LAN

6.5.2 Analysis

To aid the analysis, reception tasks of the different standards are divided into the following processing groups:

- Radio front-end processing.
- Symbol processing.
- Data recovery.
- Forward error correction and channel coding.

In order to compare the computational load of a certain algorithm, a "MIPS cost" is assigned to each operation according to the definition in Section 6.2. The MIPS cost corresponds to how many million instructions-per-second a regular DSP processor would require in order to perform the specified function within the specified maximum latency period. A normal software implementation effort is assumed.

To estimate the area used by a corresponding hardware accelerator, the circuits have been synthesized to the UMC 0.18 μm process using Synopsys' Design Compiler in cases where no previously published implementation exists.

6.5.3 Radio front-end processing

The first task for a baseband processor is to filter and decimate the input signal, in order to reduce interference from adjacent channels and to relax requirements on the analog-to-digital converter. The occupied band-

Table 6.2: Bandwidth and sample rate

Standard	Bandwidth	OSR	Sample rate	Filter
IEEE 802.11a	20 MHz	2	40 MHz	RC, $\alpha = 3/64$
IEEE 802.11b	11 MHz	2	22 MHz	RRC, $\alpha = 0.22$
W-CDMA	3.84 (5) MHz	4	15.36 MHz	RRC, $\alpha = 0.22$

width, the Over-Sampling Rate (OSR) and the required sampling rate are presented in Table 6.2.

As stated in the table above, the maximum sample rate the processor needs to process is 40 MHz. A raised-cosine FIR filter, which maintains the phase of the received signal, is used as decimation filter. Raised-cosine filters have a symmetrical impulse response with N/2-1 zero taps for a filter length of N. This property will reduce the computation load significantly since the processor does not need to calculate those taps.

The number of taps in the FIR filter is determined by the smallest roll-off factor. The roll-off factor determines the transition band of the filter. A lower roll-off factor yields better filter performance since the transition band is small, however the length of the filter increases accordingly. IEEE 802.11a [3] requires a roll-off factor of $\alpha = 3/64 = 0.0468$. Filter synthesis tools give a filter length of 21 taps for a filter with $\alpha = 3/64$.

For comparison, a cycle cost of 30 operations per processed input sample is assumed. This cycle cost also includes control flow instructions. Normally all 21 complex filter taps are processed. However since at most every second output sample is used, savings can be made. 30 operations per sample is a moderate estimation for such a large filter. The root-raised cosine filters used in IEEE 802.11b and WCDMA require 20 operations per sample. The resulting required MIPS cost is presented in Table 6.5.3.

This MIPS cost should be compared to the circuit area of a configurable front-end filter implemented as a function level accelerator which occupies 0.17 mm^2 in a 0.18 μm process [2].

Table 6.3: MIPS cost for front-end filtering and decimation

Standard	Required MIPS
IEEE 802.11a	1200
IEEE 802.11b	440
W-CDMA	307

6.5.4 Symbol processing

Symbol processing of the considered standards includes both channel estimation, channel compensation and (de-)modulation of data symbols. These operations are only briefly described here to highlight some of the operations needed to be performed on received/transmitted data. The MIPS cost for the operations are also estimated and compared with fixed function silicon implementations.

A more detailed discussion regarding the implementation aspects of OFDM and CDMA symbol processing is presented in the case studies in Section 9.5.

In Direct Sequence Spread Spectrum (DSSS) and CDMA based standards, channel estimation is performed by using a matched filter (correlator) to estimate the channel impulse response. In such systems the matched filter is run continuously to track channel changes. The associated MIPS cost of the continuous channel estimation is included in the MIPS cost listed for the rake unit. Moreover, in DSSS and CDMA systems, channel compensation and de-spread is performed by a rake structure [6]. The rake unit has a certain number of taps (often referred to as "fingers"), which correspond to taps in the channel impulse response. By using four "fingers", up to 90% of the received signal energy can be used to recreate the transmitted signal in an office environment [7]. A rake finger is illustrated in Figure 6.1.

However, in OFDM systems the channel is estimated and symbols are recreated in the frequency domain. The conversion from the time domain to the frequency domain is performed by an FFT. The cost of a 64

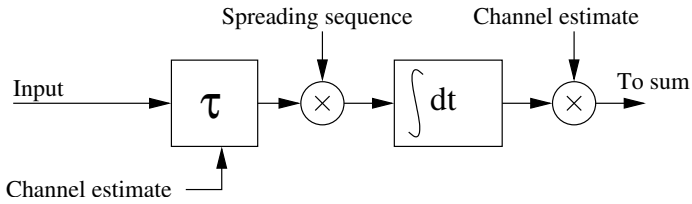

Figure 6.1: Rake finger.

point FFT is approximately 700 clock cycles in a regular processor. For OFDM systems, where the channel is corrected in the frequency domain, a rudimentary channel correction mechanism is to multiply the received frequency domain data with the conjugate of the channel estimate and use the energy information in the de-mapping stage.

The associated processing cost for initial channel estimation, synchronization, and recovery of the transmitted symbols is presented in Table 6.4.

Moreover, in the higher data rate modes of IEEE 802.11b (CCK) the data are recovered by using a Modified Walsh Transform (MWT). To extract data bits from the result of the MWT, a max-energy search is performed on the transformed data thus resulting in a higher MIPS cost.

Since the FFT, MWT and de-spread are the major corner stones of

Table 6.4: MIPS cost for initial channel estimation, synchronization, and recovery of the transmitted symbols

Standard	**Type**	MIPS for **Channel est.**	MIPS for **Data reception**
IEEE 802.11a	FFT	430	223
IEEE 802.11b	RAKE/MWT	176	657
W-CDMA	RAKE	25[1]	384

1: Assuming the terminal is already synchronized to the base station.

OFDM, CCK and CDMA processing it is beneficial to employ dedicated hardware accelerator blocks for the above-mentioned operations. Especially since the implementation of such hardware has been widely studied and efficient solutions exist, e.g. the radix-2^2 [8] implementation for FFT and FlexiRake [9] for WCDMA.

A multi-standard solution relying on function level acceleration for modem processing would need one accelerator block for each standard, thus consuming more silicon area and power. Instead, instruction level acceleration is considered which enables efficient hardware reuse.

Even hardwired FFT blocks will contain large hardware components such as complex valued multipliers and hence occupy a significant silicon area. If these resources instead are shared among different operations by incorporating them in a powerful execution unit attached to the processor core, these expensive hardware components can be reused by completely different instructions and algorithms. This type of hardware multiplexing often enables a programmable solution to reach a smaller total silicon area than a corresponding fixed function solution.

To illustrate the effectiveness of instruction level acceleration, consider a processor that could support one radix-4 FFT butterfly operation per clock cycle, then only 64 such instructions would be needed for an FFT.

This should also be compared to the silicon area for fixed function solutions which is 0.19 mm^2 for a rake unit and 0.28 mm^2 for a combined FFT/MWT accelerator implemented in a 0.18 $\mu\mathrm{m}$ process [2].

Also, as the symbol processing (modem) stage is the stage in which smart and adaptive algorithms will provide most performance gain, it further makes sense to utilize instruction level acceleration here to provide maximum flexibility of the processing solution.

6.5.5 Demappimg

When the data symbols have been recovered by the symbol processing step, they are de-mapped to extract the binary information from each symbol. Normally, the demapper uses reference levels derived from the

Table 6.5: MIPS cost for de-mapping and data recovery

Standard	Required MIPS
IEEE 802.11a (64-QAM)	144
IEEE 802.11b (DQPSK)[1]	8
W-CDMA (16-QAM)	8

1: Note that a 64 point maximum search of a complex vector also must be performed to recover all data bits in case of CCK modulated data.

channel estimate to provide thresholds for decision making. Depending on the implementation and performance requirements of the following forward error correction step either *soft* or *hard* bits are extracted. Hard bits are recovered by applying thresholds to the complex valued symbol whereas a soft bit is accompanied by a certainty value indicating the distance to the decision boundary.

For each 64 QAM symbol (6 bits worth of data) 6 arithmetic operations and 6 conditional logic operations must be performed in a hard demapper. This yields a total of at least 12 operations per received symbol in 64-QAM. The cost of de-mapping and data recovery is summarized in Table 6.5. This MIPS cost should be compared to a flexible accelerator performing hard map/de-mapping which occupies 0.016 mm^2 in a 0.18 $\mu\mathrm{m}$ process [2].

6.5.6 Forward error correction and channel coding

Forward error correction and channel coding are the most computation demanding functions performed in a baseband processor and are necessary in order to improve the data rate over noisy channels.

Several different encoding schemes are often combined to further reduce the Bit Error Rate (BER).

- **Interleaving.** Data are reordered to spread neighboring data bits in time and in the OFDM case among different frequencies.

- **Convolutional codes.** This class of error correcting codes includes regular convolutional codes as well as turbo codes. Turbo codes and regular convolutional codes are very similar. Decoding of convolutional codes is performed by the Viterbi algorithm [10], whereas Turbo codes are decoded iteratively by utilizing the Soft output Viterbi algorithm. [11].
- **Scrambling.** In several standards data are scrambled with pseudo-random data, to ensure an even distribution of ones and zeros in the transmitted data-stream.

Since all these schemes operate on bits, and since bit operations and irregular execution is very inefficient in a signal processor, the required MIPS cost is very high. Interleaving for IEEE 802.11a costs 5 op/bit whereas interleaving for WCDMA costs 32 op/bit. The very irregular memory access pattern used in most interleavers often prohibits efficient software implementations of the interleaver. Viterbi and Turbo codes are also considered large research areas by themselves. The MIPS cost for Viterbi and Turbo decoding are acknowledged according to Table 6.6. [10, 11] Scrambling is also a very cycle consuming task for a programmable DSP since it requires many bit-operations. A cycle cost of 3 op/bit for scrambling in IEEE 802.11a/b has been assumed. The MIPS costs of interleaving, Viterbi/Turbo decoding and scrambling are presented in Table 6.6.

This MIPS cost should be compared to flexible accelerators performing interleaving, Viterbi/turbo-decoding and scrambling which occupies

Table 6.6: MIPS cost for various FEC tasks

Standard	Required MIPS Interleaving	Required MIPS Viterbi/Turbo dec.	Required MIPS Scrambling
IEEE 802.11a	270	4000	162
IEEE 802.11b	–	–	33
W-CDMA	122	1964	–

0.047 mm^2, 1.1 mm^2, and 0.011 mm^2 respectively in a 0.18 μm process [2].

6.5.7 Results

Accelerators should be selected based on the MIPS cost of the considered function, the plausibility for reuse of the accelerator block and the circuit area of the accelerator. From the sub-results presented in the sections above, where the MIPS cost is compared with the estimated silicon area for accelerators capable of performing the desired function, it can be seen that a selection of accelerators that increases the efficiency is:

- A configurable front-end filter and decimator.
- A Turbo/Viterbi decoder.
- A Map/De-map unit.
- A configurable interleaver.
- A configurable scrambler.

In addition to this, it is desirable to have instruction level acceleration of FFT and other transforms in a CMAC unit, as well as efficient support of de-spread in a CALU unit.

6.6 References

[1] Bocchi, M.; De Bartolomeis, C.; Mucci, C.; Campi, F.; Lodi, A.; Toma, M.; Canegallo, R.; Guerrieri, R.; *A XiRisc-based SoC for embedded DSP applications*; Proceedings of the IEEE 2004 Custom Integrated Circuits Conference, 2004. , Vol., Iss., 3-6 Oct. 2004 Pages: 595- 598

[2] A. Nilsson, E. Tell; *An accelerator structure for programmable multi-standard baseband processors.* Proc. of WNET2004, Banff, AB, Canada, July 2004.

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

[4] IEEE 802.11b, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, 1999.

[5] 3GPP; Physical channels and mapping of transport channels onto physical channels (FDD) 3GPP TS 25.211 V3.5.0

[6] 3rd Generation Partnership Project; User equipment radio reception and transmission; 3GPP TS 25.102 V6.0.0

[7] H. Holma and A. Toskala; *WCDMA for UMTS Radio Access For Third Generation Mobile Communications*, John Wiley and Sons, Inc., 2001.

[8] H. Shousheng, M. Torkelsson; *Designing pipeline FFT processor for OFDM (de)modulation*; URSI International Symposium on Signals, Systems, and Electronics, 1998.

[9] L. Harju, M. Kuulusa, J. Nurmi; *Flexible implementation of WCDMA RAKE receiver*;Signal Processing Systems, 2002. (SIPS '02). IEEE Workshop on , 16-18 Oct. 2002 Pages:177 - 182

[10] S. Lin, D. J. Costello Jr.; *Error control coding, Fundamentals and Applications*; Prentice Hall.,1983

[11] Cavallaro, J.R.; Vaya, M.;*Viturbo: a reconfigurable architecture for Viterbi and turbo decoding*; Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. (ICASSP '03). Volume: 2, 6-10 April 2003 Pages:II - 497-500 vol.2

Chapter 7
Related work

7.1 Introduction

The programmable baseband processor research has only been around for a decade even though the concept of SDR has existed for much longer time than so. The dream of creating a true SDR system has led to the creation of a number of different processing architectures, everything from standard processors to specialized configurable execution units based on FPGA technology. In this chapter a number of different DSP architectures as well as comparable reconfigurable logic solutions such as FPGAs are presented. As most academic research projects only focus on a small part of programmable baseband processing or design-tools, very few comparable research results exist. There are however, a few exceptions which are presented here. Besides the academic research, a limited number of commercial solutions exist. As with most commercial solutions, not many details are known about their actual implementation and performance. The following DSP architectures for SDR terminals are described in this chapter:

- Sandbridge Technology - Sandblaster
- TU Dresden - SAMIRA
- Philips/NXP - OnDSP and EVP16

- Icera - DXP
- Silicon Hive - Avispa-CH1
- Morpho technologies - MS2

PicoChip also produce programmable baseband processors, however as they target base station applications they are not included in the comparison.

7.2 Traditional implementations

The most common baseband processing approach today is to combine standard RISC processors, such as ARM [1] or MIPS [2] processors or a single scalar DSP with ASIC baseband processing blocks. The drawback of this solution is the lack of flexibility of the ASIC parts and the extra overhead imposed by the superfluous flexibility of the general purpose processor core. Usually the lack of flexibility in the ASIC block cannot easily be compensated for by extra processing capability in the controller processor.

Another common design approach is to tweak an existing DSP architecture which was not intended to manage baseband processing tasks. This approach often results in inefficient solutions since it is hard to incorporate efficient hardware support of baseband tasks in a classical DSP architecture.

7.3 Silicon Hive - Avispa-CH1

The Avispa-CH1 is the baseband version of the Avispa family of DSPs from Silicon Hive. The Avispa-CH1 is targeted at OFDM-based standards such as DVB-T/H, WiFi and WiMAX.

The core architecture of the Avispa-CH1 consists of a number of Processing and Storage Elements (PSEs). Two different types of (PSEs) are present on the Avispa-CH1: The base processor PSE and complex data

type DSP PSE. The base processor PSE is used for conditional and control code. One or several PSEs are built together into a cell with its own VLIW-style control path and program memory. (Referred to as ULIW - Ultra Long Instruction Word)

Silicon Hive's main idea is to use highly specialized cells for each of the different processing tasks in a wireless system, thus taking advantage of the different baseband processing properties. The base PSE allows generic RISC and DSP calculations to be executed on the base PSE while the complex DSP PSEs are used for the actual high performance complex arithmetic DSP calculations.

No published power figures are available for the Avispa-CH processor, however since the Avispa-CH1 processor uses ordinary VLIW instructions without compaction, the program memory overhead can be quite large.

7.4 Hipersonic-1, OnDSP and EVP16

Before Systemonic was acquired by Philips/NXP, they developed a programmable baseband processor named Hipersonic I [3].
The Hipersonic processor contains three different parts:

- Hard-wired logic for computationally intensive processing.
- A flexible DSP (OnDSP).
- An optional protocol processor for scalar computations.

The OnDSP is a VLIW/SIMD based DSP where one of the issue slots in the VLIW word controls a vector SIMD unit with support for complex computing. The OnDSP uses both "horizontal" and "vertical" (Tagged VLIW) code compaction which both compacts the VLIW word when there are unused fields and uses similarities between neighboring instructions in time.

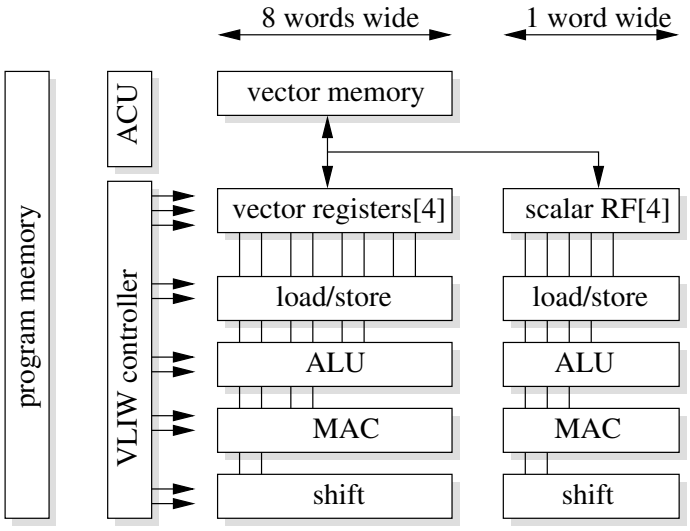

Figure 7.1: OnDSP Architecture overview

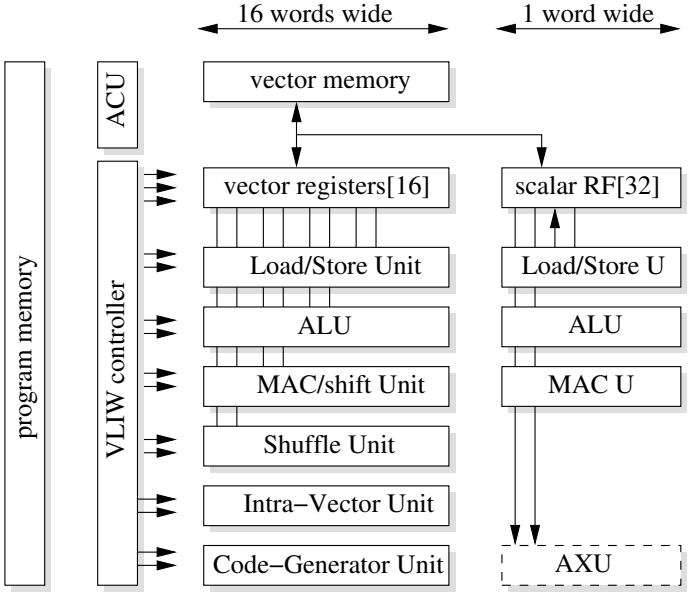

Figure 7.2: EVP16 Architecture overview

The evolved OnDSP processor from NXP, EVP16 [4].is illustrated in Figure 7.2. The EVP16 also utilizes Tagged VLIW instructions to enable compact vector programs. Both the OnDSP and EVP16 are assumed to be incorporated into a larger system-on-chip design with a host processor, dedicated hardware etc. [5].

7.5 Icera - DXP

The processors from Icera [6] use a Deep eXecution Pipeline (DXP) [7]. Icera's opinion is that a predictable execution flow can tolerate a long pipeline and that the usage of a long pipeline is necessary to utilize silicon resources efficiently.

Instructions are pointers to an entry in the configuration map which holds different configurations and constants for the 4-way deep execution data path. The configuration map is updated by control instructions. Only dynamic information is issued every clock cycle – allowing a high

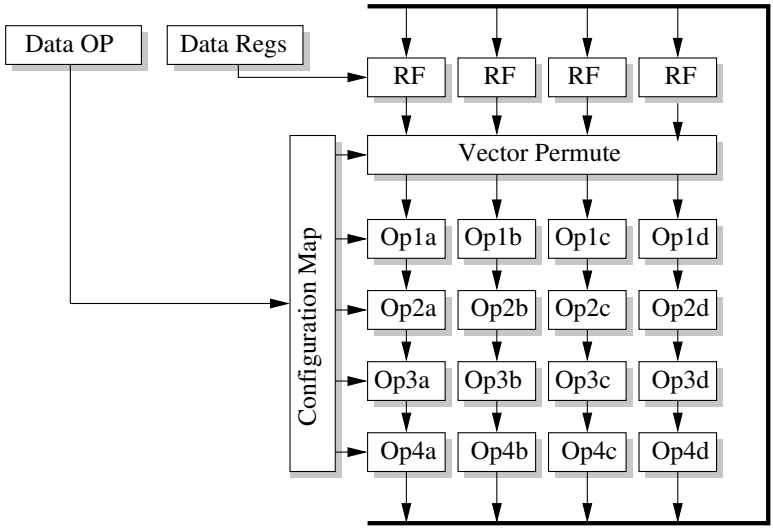

Figure 7.3: Icera Deep eXecution Pipeline (DXP)

operation count per clock cycle.

Architecture details as well as information about the memory system has not been disclosed. Processors from Icera do not use any hardware accelerators for any functionality [7]. The target market segment of processors from Icera seem to be in the WCDMA/HSPA and GSM/GPRS market segment.

7.6 Sandbridge Technology - Sandblaster

Sandbridge Technologies [8] provide a processor architecture named Sandblaster [9]. The Sandblaster DSP contains a RISC based integer execution unit and SIMD execution units. The Sandblaster platform is designed to support both communications and multimedia applications processing.

The Sandblaster architecture uses a compound instruction set for its underlying VLIW/SIMD architecture. Each instruction carries four fields: Load/Store, ALU, Integer multiplier and vector multiplier (SIMD). Thus the architecture can issue up to four simultaneous instructions where one is a SIMD instruction. To hide the effects of memory access latency, the architecture uses Token Triggered Threading (T^3). Each clock cycle of the threads can issue one instruction. The Sandblaster architecture currently supports 8 threads. In addition to this multi-threading scheme, cache memories are used in the memory system.

7.7 TU Dresden - SAMIRA

TU Dresden has proposed the Synchronous Transfer Architecture (STA) [10] which is an optimization of the Task Triggered Architecture (TTA). The basic principle of the STA is illustrated in Figure 7.4. STA based processors are built up from basic modules, (i.e. functional units) connected to other modules according to Figure 7.4. Each clock cycle, results from one module are passed to the following modules. The entire system forms a synchronous network moving data around. Hence the name STA. The

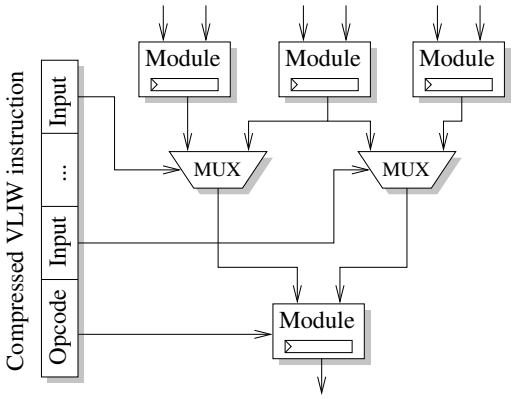

Figure 7.4: STA Architecture

simplicity of the STA architecture enables automatic generation of both RTL and simulation models from a machine description [11].

To demonstrate the STA architecture a floating point baseband processor named SAMIRA [11] was designed. The SAMIRA processor combines both SIMD and VLIW parallelism by using several SIMD floating point data-paths in parallel. To reduce control complexity of the processor, the SAMIRA processor utilizes VLIW instructions. However, to reduce the memory footprint associated with VLIW based processors, code compaction is used in the SAMIRA processor.

7.8 Morpho Technologies - MS2

Morpho Technologies [12], a research spin off from UC Irvine, provide a baseband processor architecture named "M-rDSP". The architecture consists of a 32 bit RISC processor and a reconfigurable cell (RC) array of 8 to 64 cells. Each reconfigurable cell has an ALU, MAC, register file and highly optimized specialized functional units that can be used for different wireless applications. The internal organization of an RC and its level of configurability are not disclosed. The "MS2" architecture [13], a

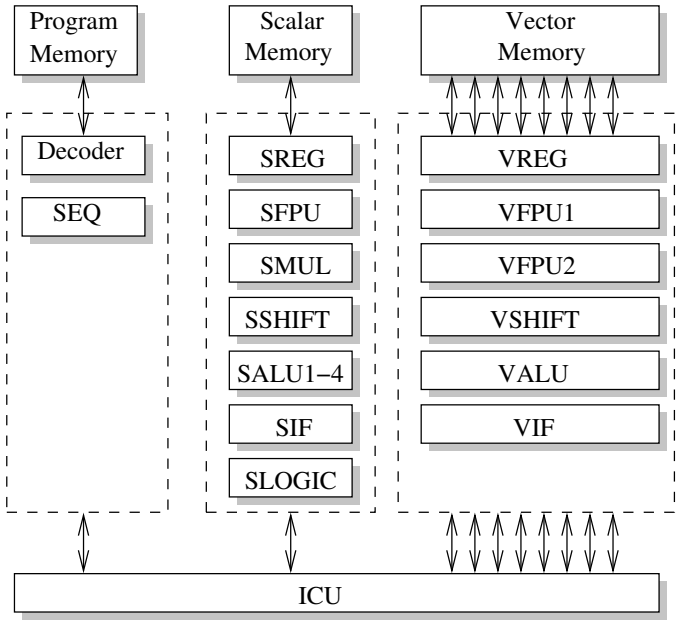

Figure 7.5: SAMIRA processor

derivate of the M-rDSP architecture, is shown in Figure 7.6.

The MS2 architecture consists of the mRISC controller with its associated instruction and data cache and memory controller, the RC Array and a DMA engine with its memory controller. The MS2 RC Array can be reconfigured cycle by cycle by using cached "computing contexts". The context memory allows the entire RC array to be reconfigured in a single clock cycle. The reconfigurable cells can load/store data from a central frame buffer. The frame buffer can supply two 8 bit values or one 16 bit value to each RC each clock cycle. In [13] it is shown that the MS2 architecture relies on external hardware accelerators for a number of tasks in a IEEE 802.11a receiver. Exactly how the accelerators are integrated in the architecture and how many cycles are consumed for data movement to/from accelerators is not disclosed.

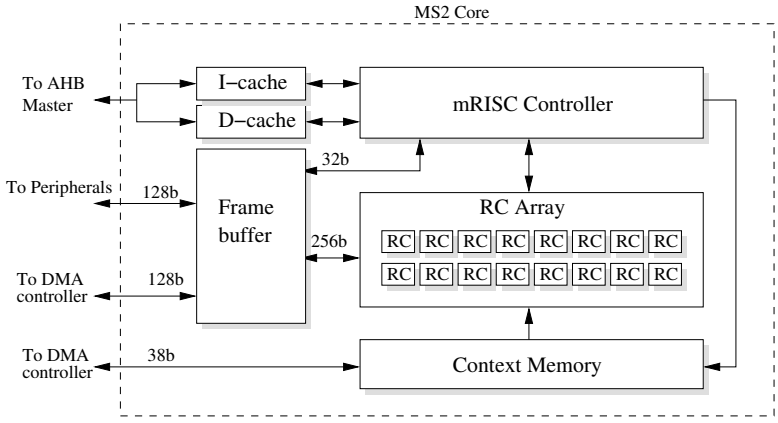

Figure 7.6: MS2 Core architecture overview

7.9 FPGA and ASIC technology

Several research groups (XiRISC [14] among others) and FPGA vendors [15, 16] suggest that multi-standard processing should be performed using reconfigurable logic such as (embedded)FPGAs. However, due to the excessive amount of resources needed for configuration [14] it is not practical to implement a complete processor using reconfigurable logic, instead certain functions are implemented in FPGA technology.

7.10 Discussion

A comparison of the SIMT architecture presented in this dissertation with the previously presented architectures yields several important differences in the memory system, instruction issue, execution units and acceleration philosophy between the architectures.

Architectures relying on homogenous arrays of processing elements such as the MS2 architecture from Morpho Technologies do not seem optimal from a system perspective since many resources will be redundant when executing heterogenous processing jobs. Furthermore the fine-

grain configurability of the array will increase the configuration overhead and thus the silicon area and power of the processor.

7.10.1 Vector instructions

Several of the described processors utilize "vector instructions", however these instructions shall not be confused with the vector instructions in the SIMT architecture. The vector instructions used in the SIMT architecture allow concurrent instructions to be issued while the vector operation executes on a SIMD unit. Also, by using several different SIMD units, resources can be balanced with the need of the supported algorithms.

7.10.2 Cache memories

As described earlier, Sandbridge Technology utilizes cache memories in the memory system. The main drawback of using cache memories is the unpredictability induced by the cache memory. If a cache miss occurs, the core needs to be stalled, prohibiting the use of a statically scheduled processor. By using a number of small memory blocks in the SIMT architecture, cache memories can be avoided since memory accesses can be performed in one clock cycle. This enables the SIMT architecture to utilize statically scheduled resources which can be used to improve software efficiency.

7.10.3 Acceleration

Both the accelerator selection philosophy and the way accelerators are integrated differ among the presented architectures. Like the SIMT architecture, several of the other architectures also utilize accelerator blocks to improve the efficiency of certain functions. The only exception to this is Icera who refuses to use accelerators. Tight integration of accelerators is also important to ensure a low communication overhead between the processor core and the accelerator unit. The accelerator integration principle used in the SIMT architecture was first introduced in the BBP1 processor

[17] and provides a very close integration with the core, while still maintaining a clean accelerator interface.

7.11 Concluding remarks

According to the authors judgment, the EVP-16 [4] architecture represents the closest match to the SIMT architecture in terms of computing efficiency. However, it is believed that the SIMT-architecture outperforms the EVP-16 architecture in the following aspects: The EVP-16 architecture is based on a VLIW scheme with code compaction (both "horizontally" and in time) whereas the SIMT architecture uses vector instructions over time, reaching the same utilization of its data-paths by only using a narrow instruction flow for a given baseband program. The SIMT architecture also reduces the complexity of the control-path and memory system compared to the EVP-16, yielding lower area and power. However, it should be noted that the silicon and memory efficiency is won by sacrificing *full flexibility*. Which architecture that is most suitable for a certain application clearly depends on the system environment and the predictability of the applications that are going to be implemented on the particular processor architecture.

The SIMT architecture presented in this dissertation is focused on baseband processing as this allows the instruction set to be restricted within a certain design space. The other presented architectures are generally more flexible outside the baseband processing domain, sacrificing power and area efficiency.

7.12 References

[1] ARM Limited, `http://www.arm.com`

[2] MIPS Technologies, `http://www.mips.com`

[3] J. Kneip, M. Weiss, W. Drescher, V. Aue, J.Strobel, M.Bolle, G. Fettweis; *HiperSonic: Single-Chip Programmable baseband ASSP for 5 GHz Wireless LAN applications*, Cool Chips IV, Tokyo, April 2001

[4] K. van Berkel, F. Heinle, P.P.E. Meuwissen, K. Moerman, M. Weiss; *Vector processing as an Enabler for Software-Defined Radio in Handheld Devices*, EURASIP Journal on Applied Signal Processing. 2005 issue 16, pp 2613-2632

[5] K. van Berkel, F. Heinle, P.P.E. Meuwissen, K. Moerman, M. Weiss; *Vector processing as an Enabler for Software-Defined Radio in handsets from 3G+WLAN onwards*, Software Defined Radio Technical Conference, Nov. 2004, Arizona

[6] Icera Semiconductor, `http://www.icerasemi.com`

[7] S. Knowles; *The SoC future is soft*; Presentation at IEE Cambridge Processor Seminar, Dec. 2005

[8] Sandbridge Technology, `http://www.sandbridgetech.com`

[9] J. Glossner, D. Iancu, J. Lu, E. Hokenek, M. Moudgill; *A Software-Defined Communications Baseband Chip*, IEEE Communications Magazine, January 2003.

[10] G. Cichon, P. Robelly, H. Seidel, E. Matus, M. Bronzel and G. Fettweis; *Synchronous Transfer Architecture (STA)*; In proceedings of International Symposium on Systems, Architectures, Modeling and Simulation (SAMOS) IV Workshop; July 2004, Samos, Greece.

[11] Matu, E. Seidel, H. Limberg, T. Robelly, P. Fettweis, G.; *A GFLOPS Vector-DSP for Broadband Wireless Applications*; IEEE Custom Integrated Circuits Conference, 2006, pp 543-546

[12] Morpho Technology, `http://www.morphotech.com`

[13] Niktash, A. Maestre, R. Bagherzadeh, N.; *A case study of performing OFDM kernels on a novel reconfigurable DSP architecture*; IEEE Military Communications Conference, 2005. MILCOM 2005.

[14] Bocchi, M.; De Bartolomeis, C.; Mucci, C.; Campi, F.; Lodi, A.; Toma, M.; Canegallo, R.; Guerrieri, R.; *A XiRisc-based SoC for embedded DSP applications*; Proceedings of the IEEE 2004 Custom Integrated Circuits Conference, 2004. , Vol., Iss., 3-6 Oct. 2004 Pages: 595- 598

[15] Xilinx Inc; `http://www.xilinx.com`

[16] Altera Inc; `http://www.altera.com`

[17] Eric Tell, Anders Nilsson and Dake Liu; *A Low Area and Low Power Programmable Baseband Processor Architecture*; in Proceedings of IW-SOC; Banff, Canada, July 2005

Part III

SIMT baseband processors

Chapter 8
The SIMT Architecture

8.1 Introduction

To summarize the requirements on a programmable baseband processor gathered from the previous chapters, the following points must be considered. The processor must have:

1. An efficient instruction set suited for baseband processing. Use of both natively complex computing and integer computing.
2. Efficient hardware reuse through instruction level acceleration.
3. Wide execution units to increase processing parallelism.
4. High memory bandwidth to support parallel execution.
5. Low overhead in processing.
6. Balance between configurable accelerators and execution units.

The above-mentioned points promote the use of SIMD data-paths for high computing parallelism and low control overhead. However, as several different types of operations are needed to be performed simultaneously, a number of SIMD execution units are needed. A common way to control several parallel execution units is to use a VLIW-based control-path, i.e. a *slot* of the long instruction word is assigned to each data-path.

However, as both the program memory overhead in terms of size and in certain cases (de-)compression logic is large for VLIW based designs, further optimizations are needed. By observing and utilizing the vector property found in most baseband programs, the instruction set and control-path can be further optimized. Other baseband properties that are utilized are the few backward dependencies in the processing flow and the predictability in the control flow of baseband algorithms.

In Figure 8.1 the evolution of processor architectures for SDR is shown, ending up in the SIMT architecture.

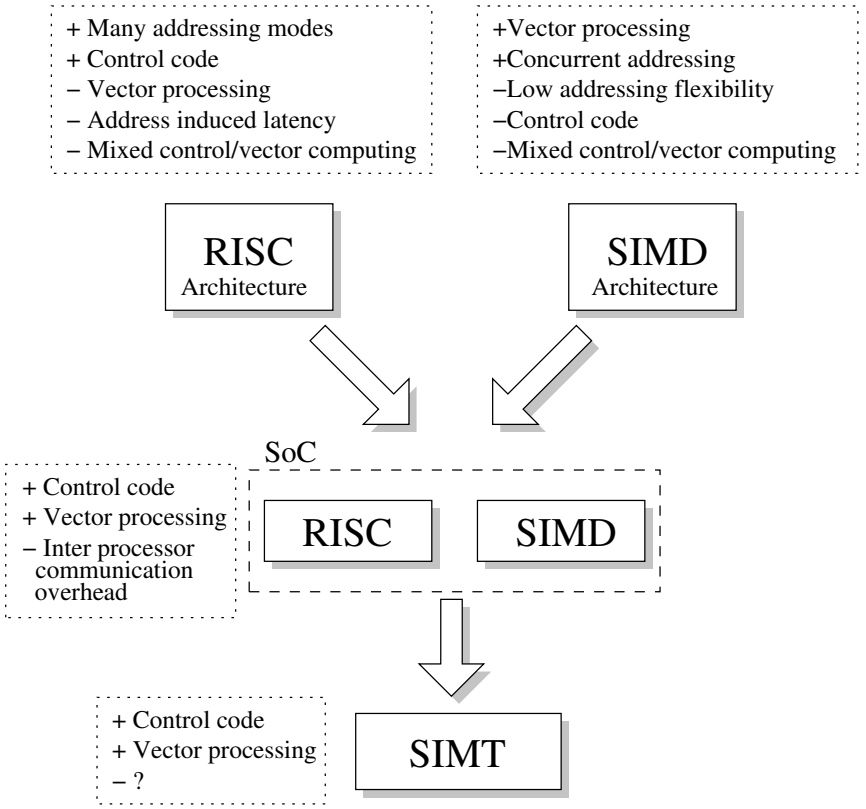

Figure 8.1: Evolution of processor architectures.

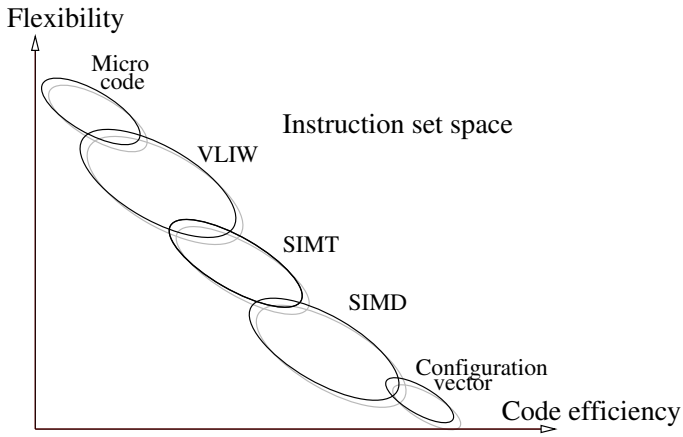

Figure 8.2: Various levels of instruction flexibility.

The key idea of the SIMT architecture is to reduce the control overhead, both in terms of hardware and program memory size by only issuing one instruction each clock cycle, while letting *vector instructions* operate on large data sets on one of several SIMD execution units **for a number of clock cycles**. Since the vector operations span over multiple clock cycles, the architecture can achieve large processing parallelism while only issuing one instruction each clock cycle. The instruction issue and details of the achieved processing parallelism are further discussed in Section 8.3. Various levels of flexibility and code efficiency are illustrated in Figure 8.2.

In short, the proposed architecture uses SIMD execution units to provide processing parallelism while using a SIMT control path to reduce the control path overhead of the processor, providing a trade-off between the flexibility of VLIW processors and the efficiency of SIMD processors.

The use of vector instructions are enabled by the underlying architecture which consists of:

1. Vector execution units (Complex MACs and Complex ALUs).

2. Small memory blocks with decentralized address generators.

3. An on-chip network.

4. A scalar data path (core) executing RISC instructions.

5. Accelerators for certain tasks.

A schematic overview of the base SIMT architecture is presented in Figure 8.3. The design and selection of each of the key components in the architecture are motivated in the following sections.

8.2 Assembly Instruction Set

From benchmarking it can be seen that a baseband processor needs support for both scalar instructions operating on integers and DSP instructions operating on complex valued fractional numbers. Based on this result, the Instruction Set Architecture (ISA) of a SIMT processor consists of three different classes of instructions:

1. **RISC instructions**, operating on integers.

2. **DSP instructions**, operating on complex fractional numbers.

3. **Vector instructions**, an extension of DSP instructions operating on large vectors of data on a particular SIMD-unit.

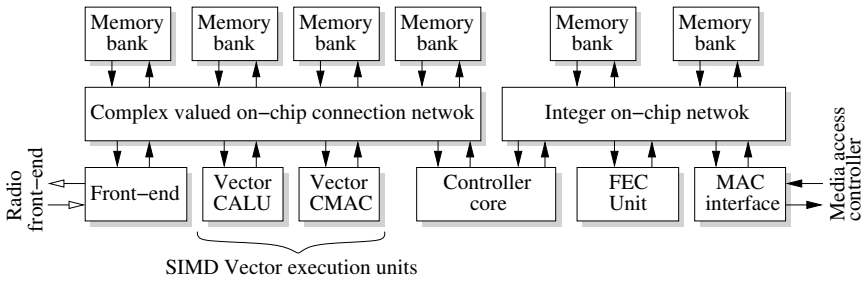

Figure 8.3: SIMT architecture.

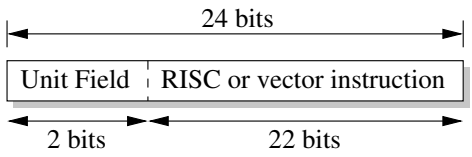

Figure 8.4: Instruction format. The bit widths are examples from the BBP2 processor.

All instructions are of the same length, typically 16-24 bit. The RISC-instruction class contains mostly control oriented instructions and this instruction class is executed on the controller unit of the processor. The DSP-instructions operate on complex-valued data and are executed on one of the SIMD units. Vector instructions are extensions of the DSP instructions since they operate on large data-sets and utilize advanced addressing modes.

Again, the vector instructions must not be confused with traditional SIMD/VLIW vector instructions. The vector instructions used in the SIMT architecture can have arbitrary vector length and will operate on a particular SIMD unit for a number of clock cycles. To reduce the program memory space, the SIMT architecture uses both RISC and CISC style instructions. However, the instruction coding is selected to maximize orthogonality between fields in the instruction word to reduce the complexity of the instruction decoder. To further reduce the control overhead of the processor, the instructions carry a *unit* field which identifies which execution unit will be executing the instruction.

The program flow controller in the controller core only decodes the unit field to determine to which execution unit the instruction will be issued. Further decoding of the instruction word is performed in the instruction decoder in the core or locally in the SIMD execution unit to reduce the control path overhead.

8.3 Single Issue Multiple Tasks

The key idea of the SIMT concept is to utilize the vector nature of most baseband processing tasks to provide high processing parallelism with low control overhead. To do so in an efficient way, vector instructions are used. Vector instructions executed on SIMD execution units will not by them self provide processing parallelism between execution units. However, as the vector instructions used in the SIMT architecture will operate for a (relatively large) number of clock cycles on an execution unit, processing parallelism can be created by issuing a number of parallel vector instructions.

By taking advantage of the fact that the vector operations will execute for a number of clock cycles on a SIMD unit, the control path of the processor can be simplified. By restricting the processor to allow only the issue of one instruction per clock cycle, the control path can be significantly simplified while the processing parallelism can be maintained due to the use of vector instructions.

This is why the architecture is named "Single Instruction stream Multiple Tasks" architecture, SIMT in short. The SIMT concept and the vector instructions are illustrated in Figure 8.5. In the example in the figure, a 128 sample complex dot-product is issued to the CMAC SIMD unit, next cycle a 32 sample complex vector addition is issued to the Complex ALU, after this miscellaneous control oriented code is executed on the scalar (control) data-path, illustrating the possibility to control a parallel machine with only a narrow instruction flow.

This type of vector instructions was first introduced in small scale in the BBP1 processor [2] implementing a Wireless LAN transceiver.

As stated earlier, the vector instructions are enabled by the underlying architecture and memory system as well as distributed address generation. To be able to sustain the computing throughput and ensure maximum usage of the execution units, vector instructions are using memory banks as direct operand sources and destinations. The SIMT architecture itself supports arbitrary vector length. The vector length is only limited

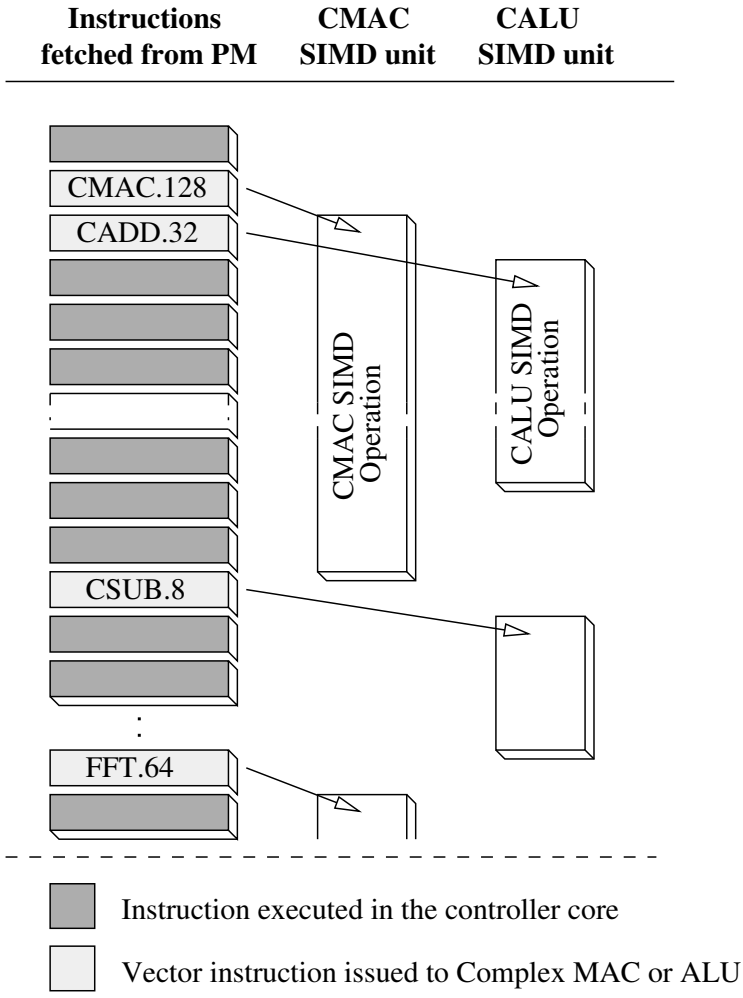

Figure 8.5: Illustration of the SIMT concept. Note how processing parallelism is achieved using only a narrow instruction stream. This is enabled by the use of *vector* instructions that operate for a number of clock cycles on a SIMD unit.

by the amount of memory and the width of the vector length field in the instruction word.

All operations that can be performed on a vector of data without inter

Table 8.1: Examples of typical vector instructions suitable for baseband processing.

Instruction	Description
cmac.n	Complex-multiply-accumulate of a vector of length N.
fft.n	Perform one layer of FFT butterflies on a vector of length N.
vadd.n, vsub.n	Perform addition and subtraction on complex valued vectors.
sqrabs.n	Calculates the square absolute value of a complex vector.

vector data dependencies are suitable for implementation as vector operations in the SIMT architecture. Benchmarking of baseband algorithms has indicated a number of operations that are suitable for implementation as vector instructions. Examples of typical vector instructions are shown in Table 8.1. Considering the low instruction width, both the processing efficiency and code-density can be significantly increased by using vector instructions compared to traditional approaches.

8.4 SIMD execution units

The design of the execution units is important to achieve an efficient yet flexible processor. The flexibility of the data-paths is a trade off between complexity of the execution unit and the instruction set of the processor. In general, the SIMT architecture can support any width of the instructions and use a custom coding for each execution unit. This allows the designer full flexibility when designing the instruction set. However, the SIMT architecture promotes the use of "narrow" instructions to keep the programs compact.

After the sub-instruction word has been issued to an execution unit it

is further decoded in the instruction decoder of the respective execution unit. The instruction decoder used in the SIMD execution units is a single-stage decoder together with a vector length counter. The vector length can either be taken from the instruction word or from a register.

From a flexibility point of view, micro code could be used in the execution units to further increase the flexibility. Micro code is however not currently used in the SIMT architecture since a one-stage instruction decoder is sufficiently flexible. Operations that require the use of micro coded execution units are more suitable to be implemented as stand alone micro coded function level accelerators.

The execution units used in SIMT based processors range from complex valued MAC units capable of executing a radix-4 FFT butterfly in one clock cycle to complex valued vector ALUs. By using homogenous SIMD data-paths within the execution units and at the same time having several different execution units attached to the on-chip network, the processing throughput and diversity are increased while maintaining the high processing efficiency.

For maximum design flexibility, the SIMT architecture can support any width (i.e. number of data-paths) of the execution units. It is also feasible to mix different execution units of different widths to increase the design-space for the system designer during initial processor design. The SIMT architecture also supports the use of floating-point execution units. Some benefits of using floating-point arithmetics are:

- Simplified programming of the processor since scaling requirements can be relaxed compared to the use of fixed-point arithmetics.
- Large dynamic range within computations.

However, the drawbacks of using floating-point arithmetics are:

- Larger execution units due to the extra complexity added by the floating-point arithmetics compared to fixed-point arithmetics.
- There is still a need for conversion of floating-point representation to fixed point representation which costs both chip area and power.

For the sake of discussion, the SIMT architecture is assumed to use fixed-point arithmetics throughout the rest of this thesis.

Since the execution units operate on long vectors of data, the internal precision and dynamic range of the execution units must be considered. In a fixed-point execution unit, the internal dynamic range of the accumulator registers can be increased by using *guard bits*. Guard bits are prepended to the accumulator registers to allow them to represent larger values than the native representation of the processor. The number of guard bits is a trade-off between hardware cost and the maximum vector length.

The following subsections will both exemplify common SIMD execution units used in the SIMT architecture and methods of reducing the amount of memory accesses for a large class of vector operations.

8.4.1 Vector management

The SIMD units will read and write complex valued data from the on-chip network (see Section 8.5). A number of baseband algorithms such as multi-point correlation will use neighboring data items (previously fetched vector elements) during consecutive clock cycles. In the same way vector data will be reused between data-paths when a complex valued vector is multiplied by multiple vectors in a wide data-path. Normally, the processor would have to fetch all data items again, causing extra memory accesses and thus wasting power. However, by equipping all SIMD execution units with a vector load unit that distributes previously fetched data among the data-paths, the memory bandwidth can be reduced significantly.

The vector load unit can be used to load vector data into the execution unit in two different ways. In the first mode, multiple data items are loaded from a bank of memories. In the other mode, data are loaded one data item at a time and then distributed to the individual lanes of the SIMD unit. This later mode is used to reduce the number of memory accesses when consecutive data are processed by the SIMD cluster. If

consecutive data are processed, the load unit can reduce the number of memory fetches by $3/4$ in a 4-way execution unit.

Similarly a vector store unit is used to create local feedback between data-paths within the execution unit. This is used to post process data in the accumulator registers without having to move data to a memory for intermediate storage.

8.4.2 Complex MAC SIMD unit

A complex multiply-accumulate unit is necessary in a programmable base-band processor. As described earlier in Chapter 6, support for certain functions such as FFT-butterflies is needed in a CMAC unit.

An overview of a 4-way CMAC unit is shown in Figure 8.6.

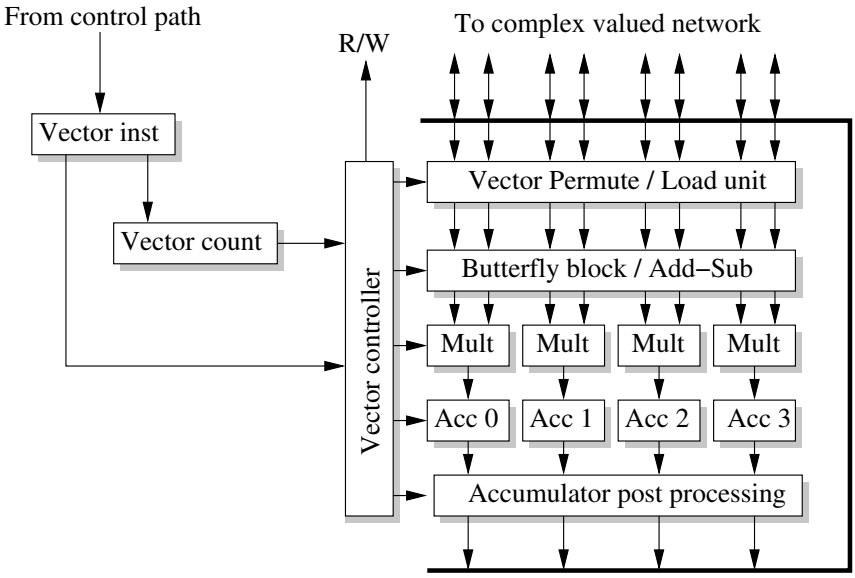

Figure 8.6: SIMT CMAC unit. (Example from the BBP2 processor.)

8.4.3 Complex ALU SIMD unit

Although most baseband algorithms have a large part of convolution based operations, there are many operations performed on vector data that can be performed in a complex valued ALU instead of in a large CMAC unit. Of course vector addition, subtraction, conjugation and accumulation can be performed in an ALU unit with accumulator registers. What is more important to realize is the possibility to use a complex ALU unit with accumulators for de-spread operations used in WCDMA for example. Since the de-spread operation is a dot-product of a complex valued vector and a complex valued sequence with values from the set: $\{0, \pm 1; 0, \pm i\}$, power could be saved by executing this operation in the ALU unit instead of in the CMAC unit as the complex ALU has significantly lower complexity than the CMAC unit.

By implementing a 4-way complex ALU unit with accumulator registers, the unit can either perform four parallel correlations or de-spread of four different CDMA channelization codes at the same time (see Section 9.5). These operations are enabled by utilizing the ALU to form a "short" complex multiplier capable of only multiplying by $\{0, \pm 1; 0, \pm i\}$.

To ensure the throughput of the execution unit, the short complex multiplier can either be controlled from the instruction word or take coefficients from integer memories or accelerators through the integer network. The over-all micro-architecture of the Complex ALU is very similar to the CMAC unit with the same sort of Instruction decoder, VLU/VSU and Vector controller. A part of the micro-architecture of the vector ALU is illustrated in Figure 8.7.

8.5 On-chip network

The SIMT architecture utilizes an on-chip network to enable communications between the execution units, the memory system or any other units attached to the network. The on-chip network is essential to enable vector instructions as the memory system needs the flexibility to quickly recon-

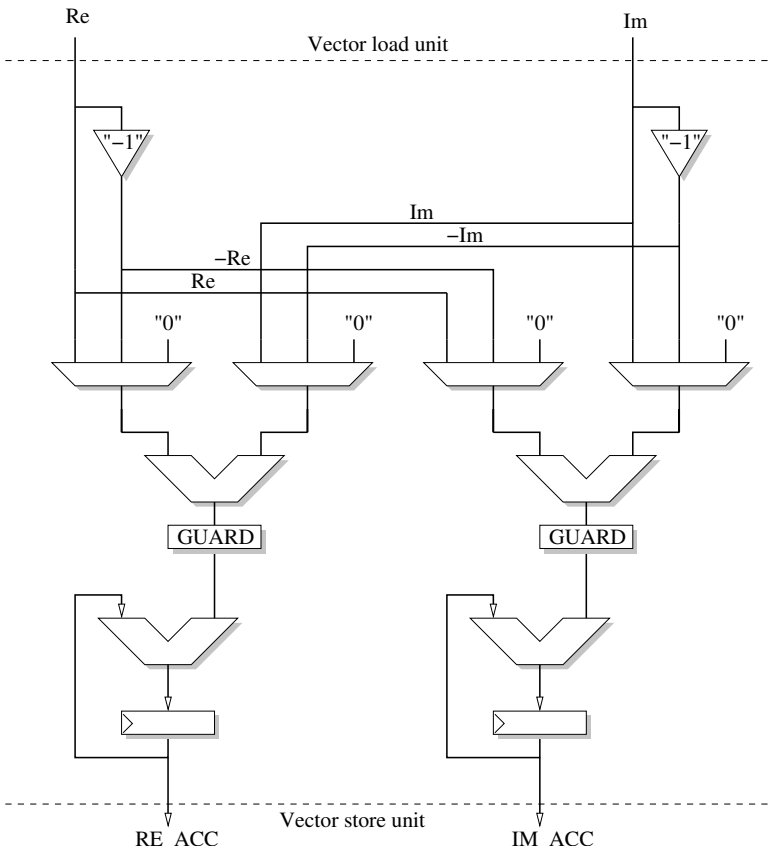

Figure 8.7: Part of vector ALU data path micro-architecture.

nect different memory banks to different execution units.

Ideally the on-chip network would be implemented as one single large on-chip network. The SIMT architecture requires a number of ports of a certain data width to be available to the execution units, memory banks and accelerators. A single network will provide ultimate flexibility since it will allow every unit attached to the network to communicate with any other unit. However, in practical implementations it is preferred to split the single network into smaller networks and thus restricting the architecture to allow only connections that make sense and thus reduce the

complexity, power consumption and chip area of the network. For the sake of discussion, two separate networks are considered during the remainder of this thesis.

A representative implementation of the SIMT architecture would have its on-chip network realized as two separate parts, one used for complex valued data and one for integer data. The two networks have different functionality to facilitate the different data access patterns and data-widths of scalar computations and vector operations. Both networks are completely combinatorial and have pipeline registers at the edge of the network thus relaxing the timing requirements of the network. Both networks are under direct software control and reconfiguration of the network only consumes one clock cycle and does not affect other communications occurring in the network. Since the networks are implemented as crossbar switches, the network will always provide "private" access to memories and execution units avoiding resource conflicts and bus contention found in other network-on-chip architectures. The crossbar switch ensures predictable execution time which is necessary to ensure good optimization during compile time of programs.

8.5.1 Complex network

The complex valued network is mainly used to connect execution units to the memory system in an efficient way. Execution units and certain accelerators (digital front-end for example) are masters on the network whereas memories are slave units. In a processor supporting radix-4 based algorithms, each network port is required to be at least 128 bits (4 complex words $\times$ 32 bits) wide.

Besides the data signals, the switch transports read and write signals from the master units to the respective recipient. As all slave units (mostly memories) are assumed to be ready to deliver data each clock cycle no handshaking is necessary. This further reduces the number of set-up cycles of vector operations and promotes predictability.

8.5.2 Integer network

The integer network has different requirements from the complex network. First, as accelerators or bit-manipulation execution units with unpredictable execution time will be attached to the integer network, handshaking is necessary. Secondly, as each port on the integer network is narrower than a normal port on the complex valued network, extra interfacing logic would be required to interface the units efficiently. Third, it does not make sense to connect the FEC part of the processor directly to the complex valued part.

These three points motivate the use of split networks in most SIMT based processors. As introduced and demonstrated in the BBP1 [2] processor, handshaking can be used to create "processing chains" of accelerators attached to the network connected in series. The handshaking signals are used to automatically synchronize the operation of the accelerators without involvement from the processor core.

8.6 Controller core

The RISC controller core is the unit which contains the program memory and the Program Flow Controller (PFC). The controller core executes ordinary RISC style instructions on its integer units. To enable efficient support for control software and operating system functions the controller core needs to be equipped with at least the following minimal set of units:

- A program flow controller (sequencer) supporting multiple contexts.
- An arithmetic/logic unit.
- A real-valued multiplier.
- A local integer memory used for operation system support and software stack.
- Register files with multi context support.
- A network interface unit connected to the on-chip network(s).

A schematic illustration of a controller core suitable for use in the SIMT architecture is illustrated in Figure 8.8.

8.6.1 Multi context support

To enable efficient multi-task support and to simplify programming of SIMT processors, the controller core has support for multiple computing contexts. Multi-contexts support and fast context switches are necessary to efficiently be able to run unrelated control software on the controller core while executing high priority vector tasks on the SIMD units. Single cycle context switches are enabled by the use of dedicated resources for each context in the controller core. The following resources are dedicated to each context:

- Program Counter.
- Stack pointer.

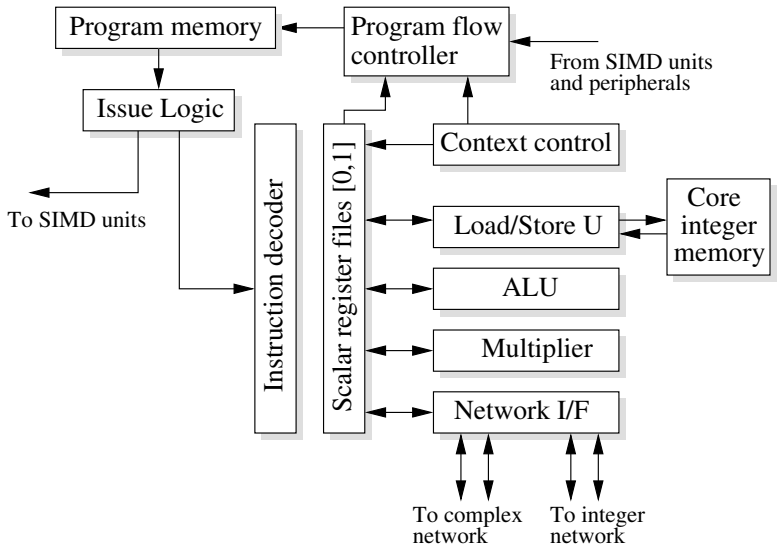

Figure 8.8: Illustration of the controller core used in the SIMT architecture.

- Register File. (Only for the computing contexts 0 and 1.)

Here three different contexts are assumed, two of the contexts are used by normal baseband programs and one context is designated for interrupt service routines enabling efficient operating system support. The context can be switched in one clock cycle with dedicated context switch instructions. A program can either switch context of all resources or only the register file. This feature combined with the possibility of automatic context switch upon interrupts or completion of vector instructions are used to enable simultaneous multi-standard support which is described in Chapter 10.

8.7 Memory system

The amount of memory needed is often small in baseband processing, but the required memory bandwidth may be very large. As seen in previous chapters, FFT calculations in DVB-T alone may need a memory bandwidth of several hundred Msample per second, averaged over the entire symbol time. In practice the peak memory bandwidth required may be up to 1024 bits per clock cycle for a processor running at a few hundred MHz. High memory bandwidth can be achieved in different ways – using wider memories, more memory banks, or multi-port memories – resulting in different trade-offs between flexibility and implementation cost. It is also important that the memory architecture balances the memory bandwidth against the access rate to reduce the overall power and area cost for the memory sub-system.

Baseband processing is characterized by a predictable data flow with few data dependencies and regular addressing, which means that flexible but expensive multi-port memories often can be avoided.

Based upon these observations, the SIMT architecture uses a number of *memory banks* connected to the on-chip network. The usage of memory banks connected to execution units via the on-chip network provides a very high memory bandwidth which is required by parallel vector operations. Another feature of using an on-chip network is the possibility

to "hand over" memory blocks to other execution units or accelerators – eliminating unnecessary data moves between memories and thus reducing the power consumption of the processor.

The number of memory banks used in a SIMT based processor is a trade-off between achievable computing parallelism and complexity of the on-chip network. To exemplify the memory bandwidth requirement, consider the case of a processor receiving data from the ADC while performing two parallel vector operations and at the same time sending data to the de-mapper accelerator. In this case eight memory banks are needed, two source buffers and one destination buffer for each vector operation in addition to the memory buffers needed by the ADC data and the de-mapped data.

Generally the memory sub-system accounts for a large portion of both the area and power consumption in digital signal processing systems [1]. It is therefore very important to optimize the memory system in such a way that the memory bandwidth is balanced against the power consumption resulting from a higher access rate.

To achieve a high enough memory bandwidth and avoid using dual-port memories for operations such as FFT, each memory bank is formed by a number of memories operating in parallel. The data width of each individual memory is wide enough to hold one complex valued sample. This implies that a SIMT processor with 4-way SIMD units would need to have four memories in each memory bank to provide enough memory bandwidth.

Another key design goal during the development of the SIMT architecture has been to avoid the use of cache memories, both from a area and power consumption point of view, but also from a software point of view since the absence of cache memories ensures predictable execution time. Predictable execution time enables the use of statically scheduled programs – thus reducing the control software overhead.

8.7.1 Addressing

To sustain the high computing throughput enabled by vector instructions, the memory system with its address generating logic must be able to generate addresses in the same pace as the execution units access the memory. To enable this high throughput in the addressing system, decentralized addressing is used for most vector operations. Each memory bank contains its own address generator capable of generating a set of common addressing patterns:

- Linear addressing.
- Modulo addressing.
- Bit-reversed addressing (used for radix-2 FFT for example).
- Non-collision radix-4 addressing (used in radix-4 FFTs to avoid memory conflicts).

The addressing modes mentioned above cover most of the addressing needs of ordinary baseband programs. There are however exceptions of this, for example Rake-finger addressing used in WCDMA channel compensation and pilot addressing used to extract pilot sub-carriers in OFDM based standards. Common to these addressing patterns is their irregularity. To accommodate support for these operations, the memory blocks can also take addresses from the integer on-chip network where the addressing pattern can either be stored in an integer memory or be taken from an address calculation accelerator.

In this way the benefits from centralized addressing are combined with the efficiency of decentralized addressing. Decentralized addressing provides the following benefits:

- By using decentralized address generators, each address generator could be optimized for the size and layout of the corresponding memory bank, thus reducing the over-all processor complexity.
- There is no need to distribute centrally generated addresses to each memory block over the on-chip network. This reduces the number

of network transactions which in turn reduces over-all power consumption.

One drawback of using distributed addressing is the case when a certain execution unit needs a special addressing mode. This drawback is however mitigated by the possibility to use either integer memories or accelerators to address the complex valued memories.

In a vector processor with SIMD units, care must be taken to facilitate the possibility to read and write misaligned data, i.e. data that are not placed on an addresses which is a multiple of the number of lanes in the data-path. The SIMT architecture uses a small reordering crossbar switch placed in the memory bank to provide this ability. The reordering switch interfaces each memory block with the on-chip network. This crossbar switch allows any single memory to be accessed from any of the ports. This feature is especially useful when the memories are written by a single port device such as the mapper or the analog interface. Later, the memory content can be accessed in parallel in order to facilitate higher radix based algorithms and transforms. The reordering crossbar switch is also used to provide collision-free memory accesses during radix-4 based transforms (such as FFT). A complete memory bank is illustrated in Figure 8.9.

8.8 Accelerator integration

Acceleration of certain functions is used in the SIMT architecture. Design and selection of accelerators are described in Chapter 6 and in [3].

Function level accelerators are easily integrated in the SIMT concept by using the same type of interface towards the on-chip network as is used by memories and execution units. The analysis of function accelerators shows that configuration and control of function acceleration units only require few control bits. It is therefore feasible to control and monitor function level accelerator blocks by means of control registers mapped to the control register space of the controller core. Another way of controlling accelerators is to use special instructions that allow immediate

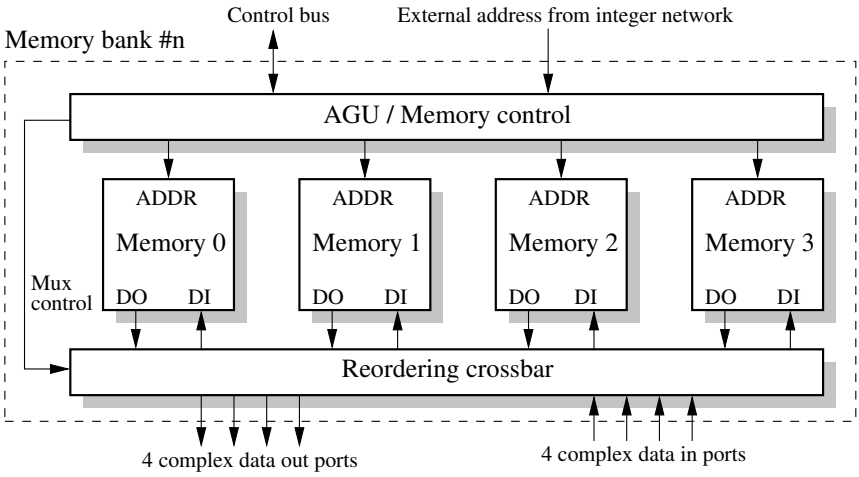

Figure 8.9: Example of a memory bank in the SIMT architecture.

operands coded in the instruction word to be sent to an accelerator. These instructions were first introduced in the BBP1 processor and provide single cycle set-up of accelerators connected to the on-chip network.

8.9 References

[1] Lidsky D.B.,Rabaey J.M.; *Low-power design of memory intensive functions*, Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium, Oct 1994.

[2] Eric Tell, Anders Nilsson and Dake Liu; *A Low Area and Low Power Programmable Baseband Processor Architecture*; in Proceedings of IWSOC, Banff, Canada, July 2005

[3] A. Nilsson, E. Tell; *An accelerator structure for programmable multi-standard baseband processors.* Proc. of WNET2004, Banff, AB, Canada, July 2004.

Chapter 9
SIMT Design flow

9.1 Introduction

In this chapter a methodology to create SIMT based processors is presented along with two case studies illustrating various design aspects to take into consideration when selecting execution units and memory organization in a SIMT based processor. The first case study highlights how to map a rake receiver to the SIMT architecture and the second case study exemplifies a mapping of a multi-mode OFDM receiver to the SIMT architecture. The second case study also describes the dimensioning of execution units and the memory system to reduce the amount of memory accesses.

9.2 Design methodology

In order to create an efficient baseband processor architecture, both in terms of memory usage, area, and power efficiency, a methodology has been developed that consists of the following steps:

1. Analysis and high-level modeling of the application.
2. Algorithm selection.
3. Mapping of the selected algorithms to the SIMT architecture.

4. Benchmarking and analysis of the algorithms on the architecture to find limiting factors on memory usage, memory access rate, latency requirements and computing capacity.

5. Selection of memory sizes, memory organization and width of the execution units.

6. Selection of accelerators (if needed).

7. Instruction set specification.

8. Implementation.

9.2.1 Analysis of the covered standards

In this first stage of the method, the covered standards are analyzed in terms of estimated memory usage, roughly which algorithms to use and which precision of the executions units is needed. Then, this information is used to determine a good starting point for the base configuration of the SIMT architecture to use in the later stages of the method. In this stage, high-level (system) models are created in either Matlab or C++. The models are used to get further knowledge and information about system and hardware requirements.

9.2.2 Algorithm selection

When the covered standards have been analyzed and high-level models have been created, it is time to do the detailed algorithm selection. In a high level model, it does not make any difference how a correlation is performed, but if the model is to be used to select and dimension hardware resources it is important to know exactly how it is going to be implemented. Algorithm selection is one of the most important tasks during the design of a baseband processor since the result will have a great influence on the rest of the project. In general, telecommunication standards are only specified on the transmitter side. Often no hints about suitable

receiving algorithms are given. This further complicates the algorithm selection part and illustrates the importance of good algorithm knowledge.

9.2.3 Mapping and benchmarking

Now, when algorithms have been selected, kernel functions must be identified and mapped to the SIMT architecture. The goal of the mapping and benchmarking step is to determine parameters of the design such as:

- Required clock frequency.
- Memory size and number of memory banks.
- Required execution units. (e.g. ALU and CMAC)
- Number of lanes in the execution units. The memory organization will follow the number of lanes in the execution units in order to fulfill memory access requirements.

In addition to kernel functions, control code should also be benchmarked to identify possible bottlenecks in the architecture. Benchmarking suites containing both kernel and application benchmarks are available from both EEMBC [1] and BDTi [2].

Cycle and memory access cost

In order to compare the computational load of a certain algorithm and the amount of memory accesses used, a cycle and memory access cost (much like the MIPS cost used for accelerator selection) must be calculated for each kernel function implemented on different architecture variations. These measures can be used to provide a trade-off between the number of memory accesses and the silicon area used by the processor.

Latency

The cycle cost must be compared to the target operating frequency and the available maximum computing latency. In most wireless systems there are four factors limiting the computing latency:

- **Channel equalization.** In a packet based system, the channel estimation performed on the preamble must be complete before the processing of the user payload symbols is started.
- **High level acknowledge packets.** In time division duplex systems and TDMA systems such as WLAN and WiMAX the standard stipulates the maximum allowed time from the end of a received packet to the beginning of the transmitted response packet.
- **Memory limitation.** Increasing task level parallelism will require more memory space for intermediate data storage. Since on-chip memory resources are very expensive, this will impose a limit on the maximum allowed latency.
- **User delay.** User protocols such as voice services limit the overall latency in a communication system.

9.2.4 Component selection

When the baseband tasks have been analyzed both in terms of latency requirements, number of memory accesses and cycle cost, the results can be used to select vector execution units, memories, and accelerators.

Vector execution units

The cycle cost and memory access rate yielded from the analysis of the worst case load will be used to select the number of lanes in the execution units. The width of the execution units will be a trade-off between circuit area (more memories are needed in higher radix circuits) and desired clock rate. The total number of memory accesses must also be taken into account when selecting execution units since this has a large impact on power efficiency.

Memory organization

The number of memory banks is mainly determined by the chosen task level parallelism whereas the width of each bank is determined by the

memory bandwidth required by the execution units.

The following example illustrates one trade-off related to the implementation of execution units capable of performing FFT butterflies. A radix-2 compatible MAC unit needs to read three complex data items and write two complex data items each butterfly operation during FFT calculations. A radix-4 unit will on the other hand require a total of 11 memory accesses per butterfly operation. However, by using a radix-4 based FFT algorithm, the overall number of memory accesses will still be lower than the a radix-2 implementation. This is a direct result of the number of butterfly operations needed in the respective case. Radix-4 based FFTs require $\frac{N}{4}log_4 N$ butterflies whereas a radix-2 implementation requires $\frac{N}{2}log_2 N$ butterfly operations.

Accelerators

Accelerators should be selected according to the methodology presented in Chapter 6

9.2.5 Instruction set specification

When the algorithms and execution units have been selected, the instruction set of the processor is fixed. The instruction set architecture acts as an interface between software and hardware development within the project. When instructions are selected, careful investigation and benchmarking are necessary to avoid an excessive number of instructions. An initial set of instructions is created by allowing all legal combinations of control signals to an execution unit, thus providing full flexibility. The initial instruction set is then refined by performing kernel and application benchmarking on the initial instruction set. Superfluous instructions are then removed from the instruction set. This procedure is iterated until the number of instructions and thus the instruction width, is balanced against the overall code size. The number of instructions is a trade-off between ease of software implementation and hardware complexity (see also Section 9.4).

9.3 Evaluation

In reality there exist several (both forward and backward) dependencies between the stages of the previously presented method. Often several iterations of the method must be performed in order for the processor development to converge towards a good solution. To help the design to converge, it is of importance to have good evaluation methods to help and guide the design process towards "a better solution". Good evaluation methods are also important in order to produce *comparable* results between different configurations of the processor architecture. In the area of programmable baseband processors it is extremely hard to compare two different solutions with each other. In general there are only three practical measures, which could be used in a comparison:

1. Required clock frequency.
2. Chip area.
3. Power consumption.

One problem which was faced during the design process, was to accurately estimate the above mentioned properties from a high level design. The earlier in the design stage, the less accurate results can be tolerated, however many of the design choices made early in the design phase rely on these estimations. Since the effects of early architectural decisions are large on the rest of the design, it is crucial to know the effects of an early design decision. The three measures mentioned above however can be accurately estimated using results from simulations. When the system has been scheduled, the required clock frequency of the design can be estimated with good accuracy. This frequency can then be used to calculate memory access rates, access power and power of execution units. By using all available data from the simulation environment, power and area figures can be estimated with good accuracy. These figures can then be checked with known data from previously manufactured chips with roughly the same topology. Last, the area of the chip can be estimated

fairly accurately by modern design tools without having to run the complete place-and-route stage. Actually this approach yields fairly accurate estimations which could be used both as comparisons with other solutions and as guidance when improving the design.

9.4 Multi-mode systems

To ensure architectural support for multiple standards, each standard must be analyzed, benchmarked and mapped to the SIMT base architecture. Then, each resource in the processor architecture must be dimensioned to manage the most demanding requirement imposed by the standards. Resource sharing and requirements from different standards are illustrated in Figure 9.1. The functionality or resource usage in the design space required by each standard is represented by a circle in the figure. To ensure support for all standards, the processor must provide resources corresponding to the union of the requirements imposed by each separate standard. By using this representation, hardware "overhead" imposed by a single standard can be illustrated.

By varying the flexibility of the execution units, the reuse (and thus the

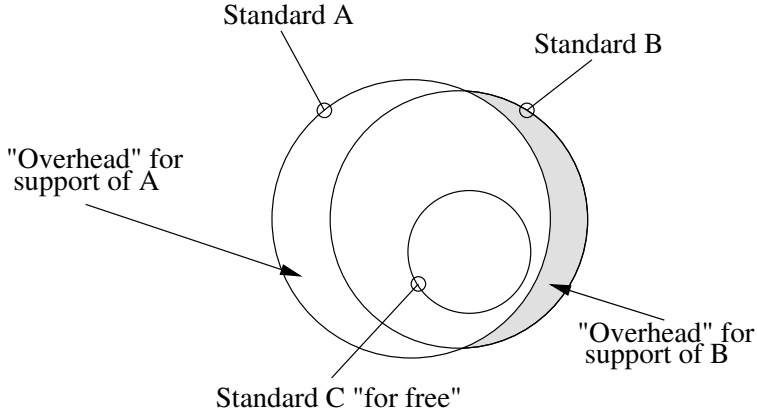

Figure 9.1: Illustration of hardware function coverage.

overlap of the circles) can be traded-off against hardware complexity. This trade-off must be considered when specifying the instruction set for any given execution unit in a processor supporting multiple standards. To exemplify this, let A represent computing resources for OFDM processing, B resources needed for CDMA processing and C resources needed for single carrier processing. Then, by implementing support for OFDM and CDMA, many resources could be shared. Furthermore single carrier processing could be managed within the resources needed for OFDM and CDMA processing.

By creating an architecture that supports the worst load in all corners of the design space, architectural support for new and diverse standards which fall inside the limitations imposed by the worst case is ensured. Standards and operations that fall within the maximum requirements will be supported "for free".

9.5 Case study: Rake receiver

9.5.1 Introduction

In this case study a mapping of a rake receiver onto the SIMT architecture is presented and analyzed. The goal of this case study is to illustrate how a specific function can be efficiently implemented on SIMD execution units in the SIMT architecture. Both channel estimation and Maximum Ratio Combining (MRC) based channel equalization for several WCDMA based wireless transmission systems are performed in software. Unlike other flexible Rake architectures [3], the SIMT processor will run both the multi-path search, Rake fingers and MRC in software [4, 5]. The created flexibility can further be used to trade-off channel compensation capability versus multi-code capability in software. For example, if channel conditions are good, computing resources can be re-allocated from channel search to multi-code de-spread yielding a higher throughput for the end user.

In this case study both WCDMA-FDD (Frequency Division Duplex)

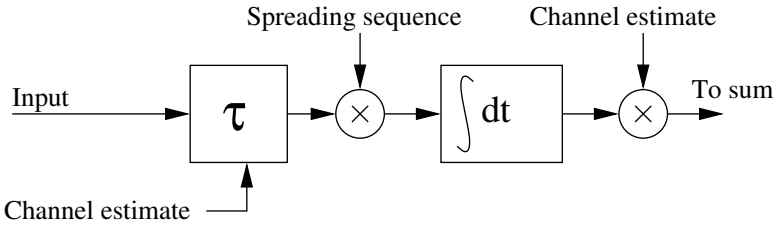

Figure 9.2: Example of a Rake finger.

and WCDMA-TDD (Time Division Duplex) modes as well as HSDPA are considered. The SIMT architecture is proven by mapping the most demanding algorithms to the architecture. This is accomplished by implementing the High Speed Data Packet Access (HSDPA) [6] mode since it contains heavy processing tasks and requires a short computing latency. By proving the HSDPA mode, architectural support for the less demanding modes can be ensured.

9.5.2 Rake based channel equalization

All the investigated standards are based on 3.84 Mcps WCDMA operating either in TDD or FDD mode. An oversampling rate of 4 was used resulting in a sampling frequency of 15.36 MHz. By using an OSR of 4, the multi-path searcher will provide quarter-chip resolution without the use of fractional delay filters. Rake receivers are used in CDMA systems for two reasons; the first reason is inter-symbol interference cancellation and the second reason is to utilize the multi-path diversity [7]. The idea of a Rake receiver is to identify a number of different multi-path components and align them constructively, both in time and phase, thus utilizing the created multi-path diversity. The function of delaying a certain transmission path and correct its phase is often referred to as a "Rake finger". A Rake finger is illustrated in Figure 9.2.

The number of Rake fingers needed is determined by both the multi-path environment and the use of multi-codes. In a Rayleigh fading out-

door environment (ITU Pedestrian B) [8] up to 93% of the scattered energy could be utilized by four Rake fingers per code used.

9.5.3 Review of processing challenges

There are several processing challenges in multi-standard WCDMA systems, namely channel estimation, the use of multi-codes and soft hand over. To further improve the data-rate to a user in a WCDMA network, the user can be assigned several spreading codes in parallel. This will allow the user to receive and decode each of the codes in parallel, thus increasing the data-rate. In most 3G standards de-spreading is accomplished by first de-scrambling the data by a scrambling sequence (Gold code), then by separating each data channel by multiplication by an Orthogonal Variable Spreading Factor (OVSF) code.

Since the de-spread operation in a classical Rake finger uses a compound code, one Rake finger is needed for each code used. This obstructs the use of multi-code transmission. If the number of Rake fingers is limited, the system must trade-off channel compensation capability against data-rate. Multi-code transmission is necessary to achieve the higher data-rates used in the WCDMA and HSDPA systems.

The second challenge is *soft hand over*. Since CDMA networks can be built as single-frequency networks, hand over between base-stations can be done by soft hand over. Soft hand over can be explained as: when a user is on the border between two base-stations, the hand over is performed gradually by transmitting data from both base-stations instead of switching base-stations instantly. This is possible due to multi-code transmission. A further development is *softer hand over* where the mobile terminal is handed over between sectors in the same base-station. Then the same code-set is used so that the user cannot distinguish the new transmitter from a strong echo. The 3GPP WCDMA standards require mobile stations to handle up to 6 simultaneous base-stations during soft hand over. To handle a soft hand over scenario with 3 base-stations and three multi-codes, at least eighteen traditional Rake fingers are necessary in the

mobile terminal in order to de-spread the data.

9.5.4 Function mapping

The four main kernel functions performed by a Rake receiver are:

1. Delay equalization.
2. De-scramble and de-spread.
3. Maximum Ratio Combining.
4. Multi-Path search and channel estimation.

These operations are mapped to the SIMT architecture. As an initial hardware assumption, a 4-way complex ALU and a 4-way complex MAC unit are assumed to be used. By separating the de-scrambling operation from the de-spread operation, the same hardware can be re-used between the different operations in WCDMA systems. The mapping result of the Rake finger functionality to the different execution units is shown in Figure 9.3. First the complex ALU unit is used to de-scramble the received data. By later feeding each accumulator within the complex ALU unit an OVSF-code, the architecture can run de-spread on several simultaneous OVSF codes.

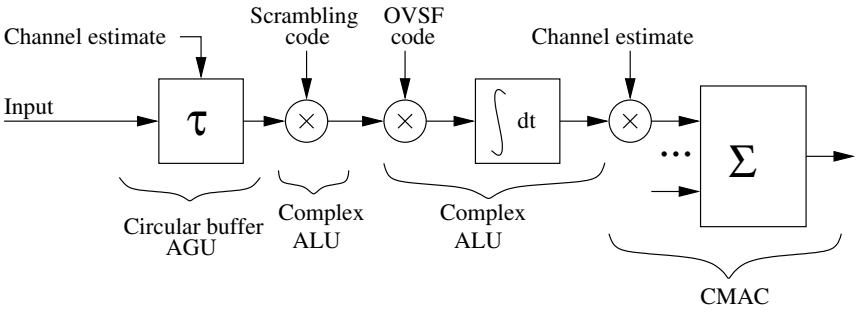

Figure 9.3: WCDMA Rake mapping.

Table 9.1: Overview of kernel functions and their cycle cost on 4-way execution units

Kernel function	Length	Cycle cost	
`vabsqr`	64	22	$\|x_i\|^2$
`vmul`	16	10	$c_i \cdot x_i$
`vmac`	256	70	$\sum c_i \cdot x_i$
`vsmac`	64	22	$\sum (\pm 1 \pm i) \cdot x_i$
`vsmac4`	$64 \cdot 4$	70	$4 \| \sum (\pm 1 \pm i) \cdot x_i$

Multi-path search is performed by using matched filters implemented as software running on the same SIMD units as the Rake finger processing. Correlation with the spreading/pilot sequence is performed by the Complex ALU, whereas peak detection and absolute maximum search is performed by the CMAC unit.

To achieve optimal resource usage, the number of computing elements is balanced between SIMD clusters so that each subtask consumes approximately the same number of clock cycles. Also, the inputs of an execution unit are masked to reduce unnecessary activity on internal logic when the unit is not in use.

As shown in Figure 9.3 the complex vector ALU along with address and code generators are the main components used for Rake finger processing. In addition to having efficient vector de-spread, efficient memory management is essential in order to efficiently use the SIMD execution units. To facilitate efficient Rake addressing the memory banks can use an external address source in addition to the AGU associated with each memory bank. Since the memory access patterns are irregular, a special AGU is used to control the delay equalizer buffer used in the "Rake finger" processing. The kernel functions used by Rake receiving algorithms are benchmarked on the hardware and the result is listed in Table 9.1 together with the associated cycle cost.

Implementation of all kernel functions associated with a rake receiver for the standards discussed requires at most a clock rate of 76 MHz. This limit is given by the memory access rate for WCDMA. To mitigate finite word-length effects during execution of the kernel functions, both guard bits and precision bits are used in the accumulator as well as scaling logic.

9.5.5 Results and conclusion

In this case study the main focus has been to investigate how to map Rake-based channel equalization for common and future WCDMA based communication standards to the SIMT architecture. Care has been taken to verify the architectural support for soft and softer hand over and multi-code transmission.

Benchmark results show that the SIMT architecture equipped with 4-way CALU and a 4-way CMAC is capable of performing all kernel functions associated with a Rake receiver for the standards discussed at a clock rate not exceeding 76 MHz de-spreading three multi-codes while performing soft hand over to six base stations. In addition to the fully flexible SIMD execution units in the SIMT architecture, only spreading code generators and a flexible address generator are needed to be implemented specifically for rake functionality. Furthermore, assembly simulations of a WCDMA Rake receiver show that the cycle cost of control code execution is masked by vector operations running in parallel.

9.6 Case study: Memory efficiency in multi-mode OFDM systems

9.6.1 Introduction

The focus of this case study is to highlight certain design considerations and illustrate what trade-offs could be performed during selection of execution units to reduce the power consumption of the memory system in a processor supporting symbol processing in multiple OFDM standards.

The case study is split in two parts, the first part presents OFDM reception algorithms and benchmark results for the algorithms [9, 10]. The second part of the case study illustrates a mapping of kernel functions to the SIMT architecture and describes the impact of execution unit selection on the memory system.

The case study uses the IEEE 802.16e [11] (WiMAX) and DVB-H [12] (Digital Video Broadcast - Handheld) standards as examples.

9.6.2 Application analysis and mapping to the SIMT architecture

Following the SIMT design methodology, high level models of the discussed standards have been developed, kernel functions have been identified and analyzed and mapped to the SIMT architecture. The processing flow of a typical OFDM transceiver is illustrated in Figure 9.4.

Since reception is more computationally demanding than transmission, only tasks from the reception flow are considered in this case study. Based on previous knowledge and MIPS cost analysis, two accelerators are proposed for the symbol processing part (a configurable digital front-end implementing channel filtering and frequency error compensation and an NCO used for timing error cancellation). These tasks fulfill all requirements for implementation as accelerators. They are also used among both discussed standards and can be efficiently implemented in hardware, see Chapter 6.

Analysis and implementation of the symbol processing tasks shown in Figure 9.4 identifies three different kernel operations which are frequently used:

- **Complex FFT** is used to transform time domain data into frequency domain data. The FFT is also used for cyclic correlation when performing synchronization.

- **Complex MAC** is used for dot-product calculations.

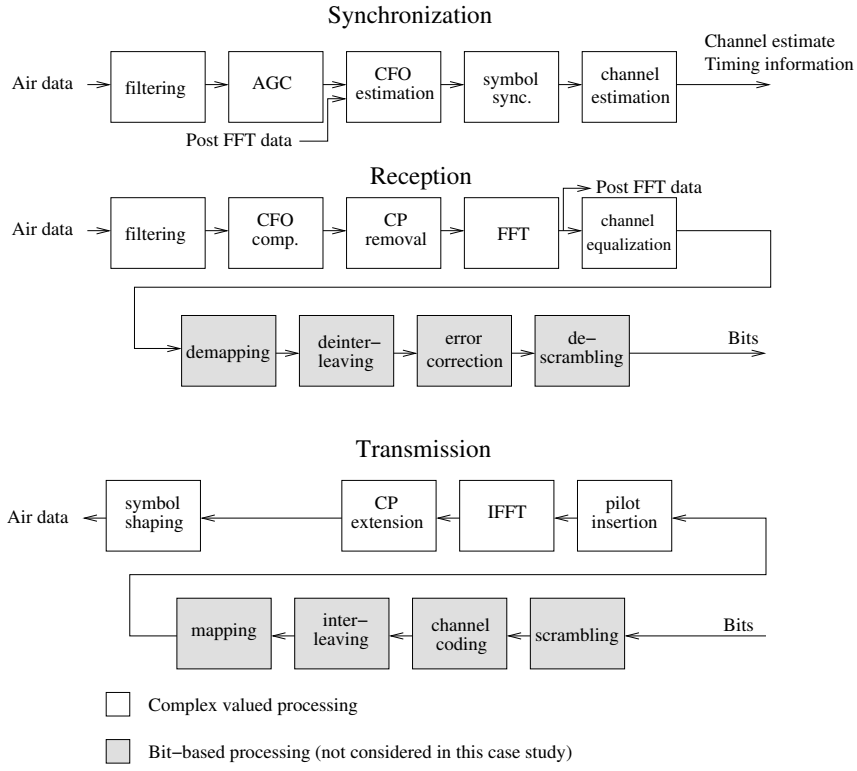

Figure 9.4: OFDM processing flow

- **Complex vector multiplication** is used for channel compensation and cyclic correlation.

All of the previously listed kernel operations can efficiently be executed on a CMAC SIMD unit with FFT support, and can therefore be implemented as software kernels.

9.6.3 Vector execution units

To ensure the efficiency of a baseband processor, all kernel functions need to run as efficiently as possible. As FFT is a cornerstone of OFDM processing, an efficient FFT implementation is necessary to maintain good

Table 9.2: Overview of memory access rates and system parameters

Operation	IEEE 802.16e 256 pts symbol	DVB-H 4k pts symbol
FFT (r2)	5120 op	122880 op
FFT (r4)	2816 op	67584 op
Channel comp.	768 op	12288 op
De-map	256 op	4096 op
Symbol length: (incl. max GI):	320 pts	5120 pts
Symbol duration: (incl. GI):	23.15 μs	462μs
Avg. access rate (r2):	265 Msamples/s	268 Msamples/s
Avg. access rate (r4):	165 Msamples/s	148 Msamples/s

overall processing efficiency. As described in section 9.2.4 the width of the execution unit (and thereby the radix of the FFT butterfly operation) can be varied to provide a trade-off between execution time, number of memory accesses and circuit area. In Table 9.2 the number of memory accesses are listed together with each operation.

It can be seen that a higher radix used in FFT calculations drastically reduces the number of required memory accesses at the cost of extra hardware in the CMAC unit.

9.6.4 Memory banks

Furthermore, by analyzing the kernel operations required in OFDM reception and all data dependencies, it can be shown that four memory banks of the same size as the received data symbol including guard period are sufficient in a flexible baseband processor. Latency requirements for all the discussed standards impose a limit which requires all symbol processing tasks to be executed within a symbol period. An example of

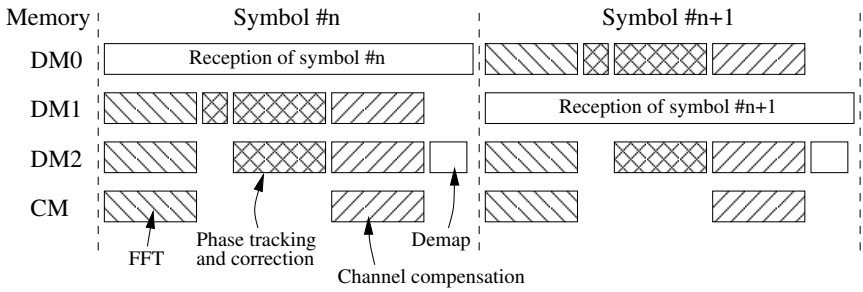

Figure 9.5: Memory scheduling for OFDM reception.

memory scheduling for an OFDM receiver mapped onto the SIMT architecture is presented in Figure 9.5.
The memories are scheduled as follows:

1. DM0: Used to store incoming symbols.
2. DM1: Used as source for calculations.
3. DM2: Used as destination of calculation results.
4. CM: Used as coefficient memory for calculations and for storage of constant vectors.

This scheme and the memory crossbar switch eliminate the need for memory moves. A possible design alternative is to use a dual-port memory instead of the two data memories that are reserved for computations. However, since the power consumption and the area for a double-port memory is roughly twice of a single port memory, only single port memories are considered.

9.6.5 Results and conclusion

Application of the presented methodology on the two discussed standards yields the following architecture parameters:

1. Four memory banks of totally 28k words of memory.

Table 9.3: Organization of memory banks

Bank	Amount	Organization	Total area
DM0:	4	2048x32	0.92 mm^2
DM1:	4	2048x32	0.92 mm^2
DM2:	4	1024x32	0.51 mm^2
CM:	4	2048x32	0.92 mm^2

2. A radix-4 compatible CMAC unit.

3. Acceleration of front-end tasks and mapping/de-mapping.

The accelerators were selected based on results from the accelerator selection methodology presented in Chapter 6. The resulting memory requirements for the multi-standard OFDM processor architecture are presented in Table 9.3. The maximum performance figures are related to the IEEE 802.16e standard whereas the maximum memory requirement is derived from the most memory demanding standard, DVB-H, which uses up to 5120 samples per symbol that is stored alternatingly in DM0 and DM1. In Table 9.4 peak memory power consumption and access statistics are presented for main receiving tasks in IEEE 802.16e implemented on this architecture.

Table 9.4: Cycle and memory access costs

Operation	Clock cycles	Accesses per cycle	Memory power	Energy
256 pts FFT	256	11	48 mW	154 nJ
Ph. tracking and comp.	264	1	4 mW	14 nJ
Ch. comp.	256	3	12 mW	42 nJ
De-map	64	4	16 mW	13 nJ

Area and power figures are collected from commercially available memory generators for a standard 0.13 μm digital CMOS process. Mapping and simulations yield a maximum required clock rate of only 80 MHz when the presented architecture runs reception tasks of the discussed standards. This maximum operation frequency is a result of latency requirements of the IEEE 802.16e standard which specifies the longest turnaround time between two packets.

9.7 References

[1] Embedded Microprocessor Benchmark Consortium. `http://www.eembc.com`

[2] Berkeley Design Technology, Inc.; *Evaluating DSP Processor Performance*; `http://www.bdti.com`

[3] L. Harju, M. Kuulusa, J. Nurmi; *Flexible implementation of WCDMA Rake receiver*; Signal Processing Systems, 2002. (SIPS '02). IEEE Workshop on , 16-18 Oct. 2002 Pages:177 - 182

[4] A. Nilsson, E. Tell and D. Liu; *A Programmable SIMD-based Multi-standard Rake Receiver Architecture*; European Signal Processing Conference, EUSIPCO, Antalya, Turkey, Sep 2005

[5] A. Nilsson, E. Tell and D. Liu; *A fully programmable Rake-receiver architecture for multi-standard baseband processors*; Networks and Communication Systems, Krabi, Thailand, May 2005

[6] 3rd Generation Partnership Project; Physical channels and mapping of transport channels onto physical channels (FDD) 3GPP TS 25.211 V6.0.0

[7] H. Holma and A. Toskala; *WCDMA for UMTS Radio Access For Third Generation Mobile Communications*, John Wiley and Sons, Inc., 2001.

[8] ITU ITU-R M.1225, Guidelines for evaluations of radio transmission technologies for IMT-2000, 1997.

[9] A. Nilsson, E. Tell, D. Wiklund and D. Liu; *Design methodology for memory-efficient multi-standard baseband processors*; Asia Pacific Communication Conference, Perth, Australia, Oct 2005

[10] A. Nilsson and D. Liu; *Handbook of WiMAX; Chapter 1: Programmable baseband processors for WiMAX systems*; To be published 2007, CRC Press

[11] IEEE Standard for local and metropolitan area networks, Part 16: Air Interface for Broadband Wireless Access Systems. IEEE802.16e-2005: WirelessMAN-OFDM/OFDMA

[12] ETSI EN 300 744, Digital Video Broadcasting. DVB-T/DVB-H

Chapter 10
Simultaneous multi-standard execution

10.1 Introduction

A natural next step in the development and use of programmable baseband processors is to extend the multi-standard support with support for simultaneous multi-standard execution. Traditionally the research focus has been on enabling multi-standard baseband processors and the issue of being able to efficiently execute several standards simultaneously has not been covered.

By using a single programmable baseband processor to execute several baseband processing programs at the same time, the following benefits are gained [1]:

- Increased hardware reuse. (Save silicon area.)
- Shared software kernel functions. (Save program memory.)
- Use of shared information, such as link state and channel parameters. (Improve performance.)

Simultaneous multi-standard execution on a SIMT baseband processor is exemplified by a case study of Unlicensed Mobile Access (UMA) in Section 10.4.

10.2 Hardware implications

No additional hardware except one Digital Front-End for each ADC/-DAC converter needs to be added to a SIMT based processor to support simultaneous multi-standard execution. A part of a system capable of operating with two different radio front-ends simultaneously is shown in Figure 10.1. As the controller core used in the SIMT architecture has support for multiple execution contexts no extra logic needs to be added to the system for context management, which limits the additional hardware to the digital front-end of each standard.

10.3 Scheduling and task analysis

As baseband processing is a *hard real time* problem with latency requirements on 1 μs scale and is computationally heavy, special attention must

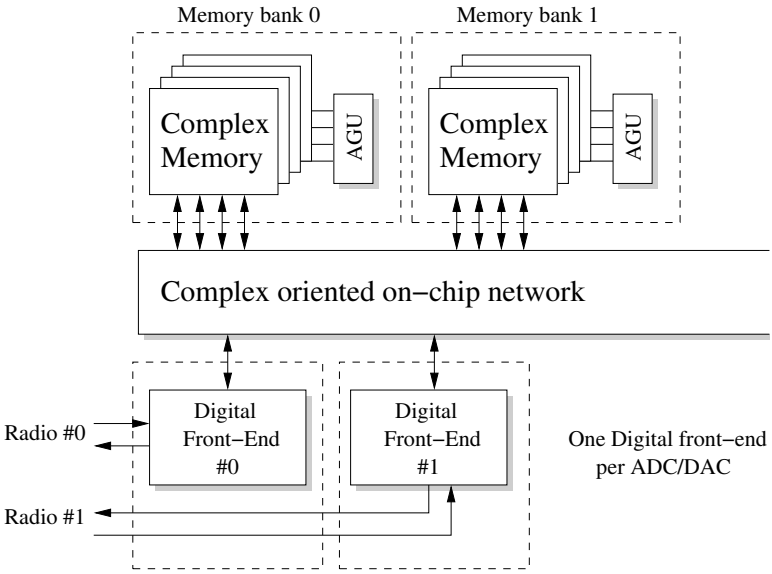

Figure 10.1: A SIMT based processor with two radio interfaces.

be paid to low scheduling overhead to achieve efficient multi-standard support. In this work, only very low complexity schedulers are considered due to the limited computing time available in programmable baseband processors. To avoid data loss, dropped packets or frames, the processor must have enough resources to support the worst case load in all supported standards simultaneously.

10.3.1 Task analysis

To aid profiling of algorithms in baseband programs and to further facilitate scheduling in the processor, the *task* concept is introduced. This "task" shall not be confused with the "task" in SIMT which refers to a vector operation. Besides being used in scheduling, division of a program into a chain of tasks helps identifying data dependencies and maximum tolerable latency of different operations during the profiling stage. In this context a task is defined as an atomic operation that is executed on one execution unit. Hence a program is considered to be a sequence of atomic tasks.

To assist profiling and scheduling, the concept of task-groups, e.g. a set of tasks with a common dead-line is defined. A task-group is further divided into a chain of atomic tasks. Some task-groups, typically associated with the radio configuration will have very short dead-lines (Typically 2-5 μs). Common to those tasks, is that they need to be performed before any data can be stored in memory. Such tasks are referred to as *immediate tasks* since they need to be serviced immediately. Automatic Gain Control (AGC) and similar tasks often fall into this group.

The immediate tasks often start a chain of further processing tasks. To handle these unscheduled events, the processor must have enough resources to manage all possible combinations of immediate tasks within the maximum latency time for each of them.

The maximum computing latency is often limited by the standard specification. For example, in most packet systems such as WLAN, the standard stipulates the allowed maximum time from the end of a received

packet to the beginning of the transmitted response packet.

The computing latency is also limited by the amount of memory available as more buffers must be used to store incoming samples while the processor is busy with the previous task.

10.3.2 Context management

To allow efficient simultaneous multi-standard execution of different baseband programs, the controller core utilizes its multi-context mechanism. The difference between the SIMT architecture and architectures without multi-cycle vector support is that the context switch will only affect the controller core and its data-path. E.g. a task could issue a vector operation to a certain vector execution unit and in the next clock cycle perform a context switch while the vector operation continues to execute on the vector unit. Control is then returned to the original context when the vector operation is completed, thus enabling a very efficient multi-tasking environment. This is especially useful in order to simplify programming and to hide the cycle cost of control oriented code with vector operations. As the hardware will not prohibit issuing new conflicting vector instructions to the SIMD units, a task scheduler must be used to coordinate the usage of the vector execution units. The usage of automatic context switching is illustrated in Figure 10.2.

10.3.3 Lightweight scheduler

As described in the previous section, a small scheduler must be used to switch between the different baseband tasks. In this work, only a very simple scheduling technique has been used to illustrate the hardware performance. As the baseband processing tasks consume about 40-100 cycles, the scheduling and task switch operations must be performed with a minimal cycle count not to dominate the processor load. The earliest deadline first scheduling principle is used by the lightweight scheduler used in this project. As indicated by the name, the scheduler will give priority to the task with the earliest deadline. To efficiently support scheduling,

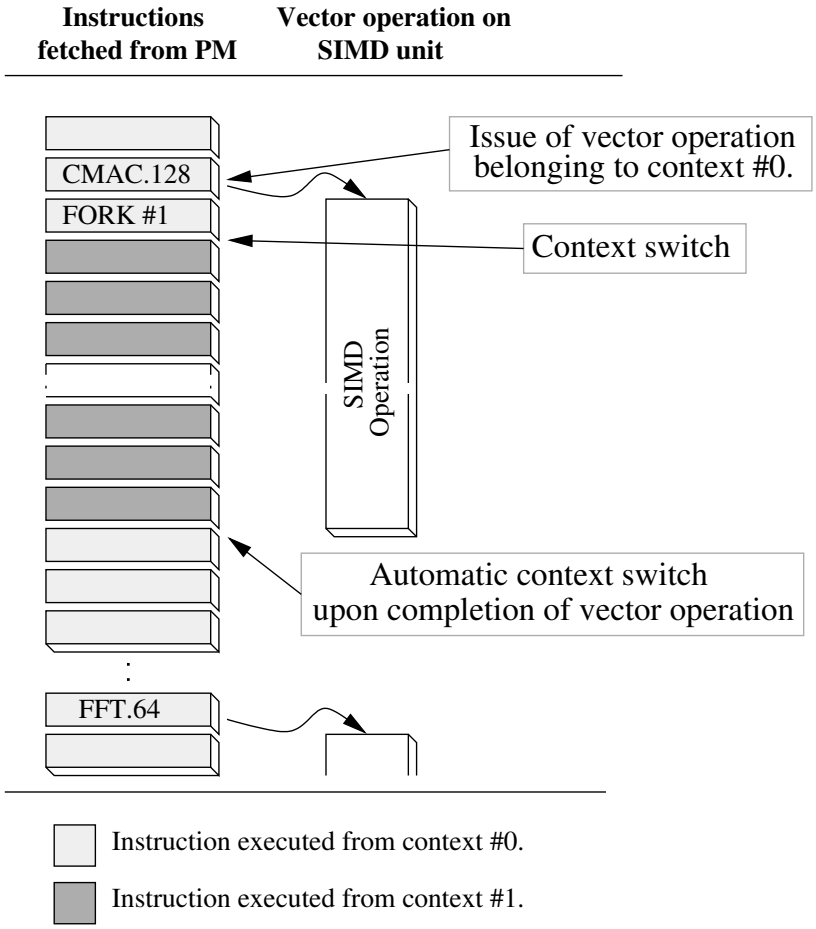

Figure 10.2: Using automatic context switches to improve control parallelism.

context switch instructions are inserted between tasks in the software at compile-time. This task partition could be performed automatically or by the programmer. Upon execution of such an instruction the processor performs a context switch to the supervisory code which then performs another context switch to issue the next task. The scheduling principle is further described in the case study in Section 10.4.3 together with esti-

mations of the scheduling overhead in a UMA system. This lightweight scheduler is not claimed to be efficient nor complete, however it serves its purpose since the design of schedulers is a complete research topic in itself.

10.4 Case study: UMA

10.4.1 Introduction

Unlicensed Mobile Access (UMA) [2, 3] technology allows end users to use GSM/GPRS mobile services over unlicensed spectra such as 2.4 or 5 GHz wireless LAN bands using IEEE 802.11 protocols. UMA also provides methods for handing over a cellular GSM connection to a wireless LAN connection without service interruption. This enables network operators to offer mobile solutions as a replacement of "fixed line" telephones in a very efficient way.

UMA technology is also interesting for network operators since they can offer new services and improve coverage by using the customers ordinary Internet access when possible instead of always using scarce resources in licensed bands. At the same time, UMA will improve the experienced network quality for users due to the extra coverage achieved from the Wireless LAN. The UMA network architecture is illustrated in Figure 10.3.

The goal of this case study has been to investigate what resources are needed to implement a UMA solution consisting of a GSM and Wireless LAN (IEEE 802.11g) receiver on a single SIMT baseband processor and to validate the usefulness of the SIMT architecture. In this work, only reception tasks in the GSM and Wireless LAN firmware are considered as reception is generally more demanding than transmission.

10.4.2 Profiling and mapping

During profiling and mapping of receiver algorithms in GSM and WLAN (IEEE 802.11g-OFDM) on the SIMT architecture the following machine

parameters have been used:

1. Four memory banks with 1024 words of complex data.

2. A 4-way Complex MAC SIMD unit capable of performing a radix-4 butterfly each cycle.

3. A controller core with a real-valued ALU and multiplier as well as private memory for software stacks etc.

4. Accelerators for cycle intensive bit-manipulation tasks such as Viterbi decoder, scrambling, interleaving etc.

The profiling of WLAN and GSM receiver programs visualizes resource usage and highlights task dependencies. During profiling, the programs are divided into chains of tasks that execute on the initially assumed hardware. Profiling results could then be used to enhance and dimension the initially used hardware architecture accordingly. In the UMA case, the profiling results suggest that the size of each memory bank could be reduced by a factor of two. In Table 10.1 basic properties of the selected standards are listed.

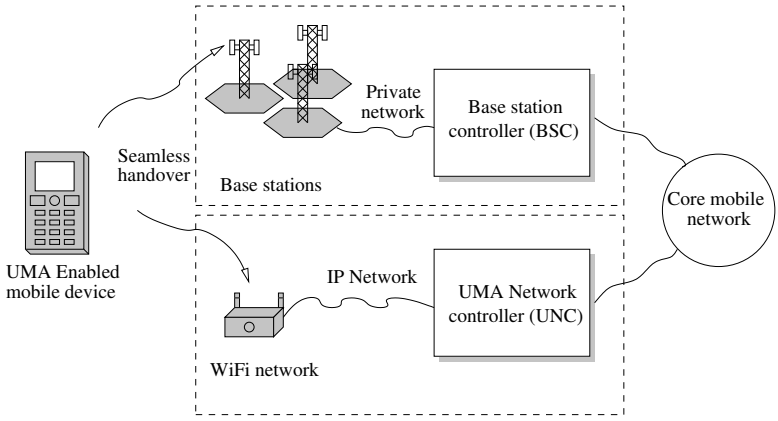

Figure 10.3: UMA network architecture.

Table 10.1: Standard overview.

Parameter	GSM	IEEE 802.11g (OFDM)
Symbol period:	3.69 µs	4 µs
Symbols/sub-carriers per frame/slot:	156.25	64
OSR:	4	2
Sample Rate:	1.083 Msps	40 Msps
Maximum latency: for packet/frame reception	576 µs	10 µs

Algorithm selection and analysis of WLAN and GSM reception yield the task division presented in Table 10.2. The cycle count cited is the number of cycles consumed by the corresponding kernel operation on the hardware. Immediate tasks need to be performed before any data can be stored in memory. Dependency analysis gives the processing latency requirements presented in Table 10.3. This table also indicates the peak memory usage by a task-group. (Local memory in accelerators is however not included.)

10.4.3 Scheduling

In this case study a lightweight scheduler is used as described in Section 10.3.3. The scheduling principle used by the scheduler is based on knowledge of the period in which GSM and WLAN tasks are initiated.

From the GSM standard it is known that a data burst might be received every 576 µs. WLAN bursts can however be received more often. WLAN activity can in worst case be back-to-back on the radio channel. With this in mind and the latencies from Table 10.3, it can be concluded that all WLAN tasks will have precedence over GSM tasks (earliest deadline first scheduling). This will require the processor to be able to serve the largest GSM task within the respective latency time. The scheduler will be called

Table 10.2: Task division and cycle usage.

#	Task	GSM normal burst	WLAN Preamble	WLAN Data
1	**AGC*:**	32 cc	48 cc	-
2	**CFO estimation*:**	-[a]	88 cc	-
3	**Synch.:**	62 cc	220 cc	-
	FFT:	-	70 cc	-
	Multiplication:	-	16 cc	-
	IFFT:	-	70 cc	-
	Max Search:	-	64 cc	-
4	**Channel est.:**	36 cc	16 cc	-
5	**Viterbi ch. dec.:**	25984 cc	-	-
	116 $\times$ Mod.:	224 cc	-	-
6	**FFT:**	-	-	70 cc
7	**Ch. tracking:**	-	-	120 cc
8	**Ch. comp:**	-	-	16 cc
9	**Demap:**	-	-	64 cc (Acc)
10	**Interleaving:**	18 cc (Acc)	-	29 cc (Acc)
11	**Conv. decoder:**	464 cc (Acc)	-	228 cc (Acc)
12	**Scrambling:**	116 cc (Acc)	-	228 cc (Acc)

[a] GSM uses a special frequency correction burst. * Immediate task.

between each task in the GSM task flow. If a WLAN packet is detected, the task scheduler will start to execute the WLAN flow.

10.4.4 Results

Table 10.2 and Table 10.3 shows that the maximum memory usage case will occur when a GSM burst is interrupted by a WLAN packet. This will require a total of 1359 words of complex memory (671 words from GSM, 208 words from WLAN and 480 constant words. Stacks etc. in the

Table 10.3: Task latency requirements and memory usage.

Task group	Maximum latency	Maximum memory usage
GSM:		
1:	3.6 µs	32 words
3-5,10-12:	570 µs	671 words
WLAN:		
1-2:	6.4 µs	120 words
3-4:	4 µs	144 words
6-8:	1.2 µs	208 words
9-12:	2.8 µs	64 words
Coefficients:		480 words

controller are not included). A detailed breakdown of the resource usage is presented in Table 10.4. In Figure 10.4 the required computing capacity v.s. time is illustrated.

The peak load of the processor will occur if a long WLAN packet is received just after the start of the synchronization task in GSM. Then the processor needs to simultaneously support 47 MIPS (GSM) and 196 MIPS (WLAN) during peak conditions as illustrated in Figure 10.4. The worst case load originates from the overall computing requirements over a GSM slot with full WLAN traffic. The case where WLAN tasks 1-2, see Table 10.2 and 10.3, occur precisely after a subtask in GSM task 5 is issued, will only require 56.25 MIPS. (224 + 48 + 88 cycles in 6.4 µs).

To reduce the power consumption, clock gating support is essential. As shown in Figure 10.4, not all computing capacity is needed all the time. Turning off the clock of individual execution units, accelerators and memories can thus save power.

A lightweight scheduling scheme has deliberately been chosen in this

Table 10.4: Overview of required MIPS and memory.

Task group	Required MIPS	Required Memory
GSM:		
1:	9 MIPS	32 words
3-5,10-12:	47 MIPS	671 words
WLAN:		
1-2:	22 MIPS	120 words
3-4:	59 MIPS	144 words
6-8:	171 MIPS	208 words
9-12:	196 MIPS	64 words
Constants:		480 words

case study to keep the scheduling overhead low, at the cost of a slightly over-designed hardware in order to maintain a guaranteed performance. By enhancing the core with single cycle context switch instructions and mechanisms for resource allocation, scheduling overhead can be kept to a minimum. Under full load in both GSM and WLAN there will be maximally 126 task switches per GSM slot (which is the largest scheduling period). According to the peak load calculated previously, the peak load of the processor is 243 MIPS. Each task switch will consume maximally 12 clock cycles (this includes context switching, network setup, et.c). This will add another load of maximally 1512 cycles per 576 μs which corresponds to 2.6 additional MIPS.

10.4.5 Conclusion

It is possible to manage both GSM and WLAN reception on the presented architecture at only 245 MHz using 1359 words of complex data memory. The scheduler and task switch mechanism adds only 1.1% cycle over-

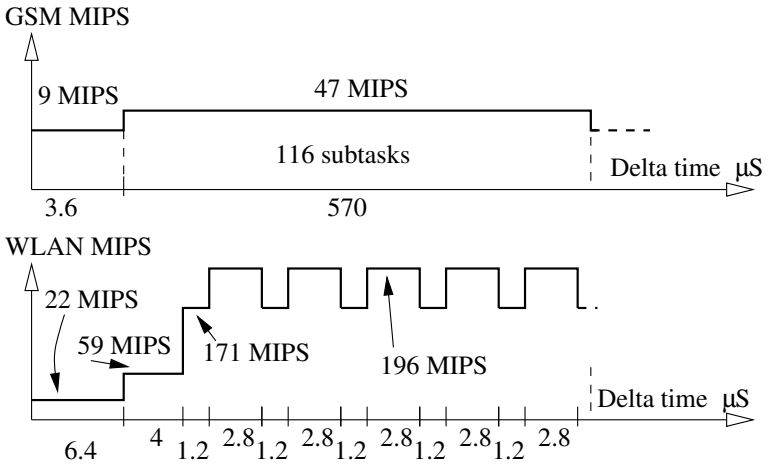

Figure 10.4: Required computing capacity v.s. time.

head. This extra overhead is compensated for by the hardware reuse. This makes SIMT based baseband processors well suited for simultaneous multi-standard processing in mobile terminals.

Profiling and mapping GSM and WLAN (IEEE 802.11g) to the SIMT architecture shows that simultaneous support for the above mentioned standards can be accomplished with 245 MHz clock frequency and 1359 words of complex data memory on a SIMT enabled processor with the execution units described previously.

10.5 References

[1] A. Nilsson, E. Tell and D. Liu; *Simultaneous multi-standard support in programmable baseband processors*; in Proceedings of IEEE PRIME 2006, Otranto, Italy, June 2006

[2] 3GPP; Technical specification: 3GPP-TS-43.318; Generic Access to A/Gb interfaces for 3GPP Release 6.

[3] UMA Technology; `http://www.umatechnology.org`

Chapter 11
Low power design

On the other hand, we cannot ignore efficiency.
– Jon Bentley

11.1 Introduction

This chapter describes several low power techniques applicable for baseband processors. Both methods to create low power processors and methods for reducing power consumption during operation are discussed.

The possibility to employ low power design techniques is greatest at design time. At design time, the following four areas have large impact on the resulting power consumption of the design:

- Memory efficiency.
- Hardware multiplexing.
- Precision.
- Use of multiple-threshold voltage CMOS technologies.

To dynamically reduce the power consumption of a system with variable load the following techniques can be used:

- Clock gating.

- Power gating and voltage scaling.
- Data width masking.

11.2 Low power design

11.2.1 Memory efficiency

Generally the memory sub-system accounts for a large portion of both the area and power consumption in digital signal processing systems [1]. It is important that the memory architecture balances bandwidth against the access rate to reduce the overall power and area cost for the memory sub-system. In order to reduce power consumption in the processor it is also important to ensure that the data is accessible in the right memory. This will eliminate unnecessary data moves between memories. Memory efficiency in a baseband processor relates to two different areas:

Code density: The code density reflects how much memory (Kbit) that must be used to store a program which implements a particular function. Code density can be measured for whole applications, kernel functions and on control code.

Memory efficiency: Memory efficiency is much harder to measure since it is algorithm dependent. Of course if an architecture is tailored for a certain application or kernel function, memory efficiency can be quite high. On the other hand, if the application contains kernel functions which are not well supported on the particular architecture the memory efficiency will be quite low. Memory efficiency is best measured by implementing a kernel function or a complete application and execute it on the processor.

By using narrow vector instructions, the SIMT architecture can achieve a high code density. Since a vector operation in the SIMT processor is fetched only once, program memory accesses are reduced compared to architectures relying on hardware loops to implement operations on vector data.

A related issue is the power consumption arising from to memory accesses. The on-chip network used in the SIMT architecture allows reconfiguration of the network in one clock cycle, hence allowing hand-over of memory banks between execution units and thereby reducing the amount of memory

By using multiple small memory blocks in parallel as opposed to using a single big memory block, the memory bandwidth can be increased and at the same time the memory access power be decreased in the SIMT architecture. Data (under NDA) from commercially available memory generators indicate that the memory access power is proportional to the size (length) of the memory array. The reduction in memory access power can be explained by the lower capacitance of the bit-lines in a smaller memory block compared to a larger memory. The parallelization of memory blocks must be balanced by the static power consumption and area overhead of using multiple memory banks.

11.2.2 Hardware multiplexing

By use of extensive hardware multiplexing, the amount of logic gates in a processor can be reduced due to the hardware reuse between different functions. This reduction, directly leads to a lower leakage power of the processor since the leakage is proportional to the number of gates in the design, assuming that low leakage cells are not switched to high-speed cells during synthesis of the RTL code.

11.2.3 Data precision optimization

Power and silicon area could also be saved by carefully selecting the precision of all execution units. Detailed computing precision selection at design time is however of limited use since the execution units in the processor often are used for many tasks – resulting in maximum precision in all execution units. However, in accelerators or similar units where the precision requirements are known at design time, data precision optimization can reduce power significantly [2]. See Section 11.3.1 where

dynamic data width masking is presented.

11.2.4 Low leakage standard cells

At low level design, the use of a multiple-Vt CMOS technology enables the use of low leakage cells for logic that is not in the critical path. This will reduce the amount of leakage without affecting the performance of the rest of the design. Multi-Vt cells allow the designer to trade-off cell leakage versus speed. As a rule of thumb; if the design is not constrained by timing constraints it is probably too parallel and therefore consuming excessive amount of silicon area.

11.3 Dynamic power saving

The large difference in computing complexity required by different wireless standards will cause excessive power consumption in a programmable processor when a "simple" standard is run on a machine dimensioned for a demanding standard unless power reduction techniques are used.

A processor designed to support mobile WiMAX would for example idle more than 90% of the time during reception of a GSM burst due to the difference in complexity. However, the overhead of running a low complexity task on a powerful processor can be reduced by using clock and power gating.

In the same way dynamic data masking can be used to reduce the precision of execution units to save power when the precision is not needed.

11.3.1 Dynamic data width

When full precision is not needed in arithmetic execution units, the precision can be reduced by *masking* a number of LSB bits, i.e. setting bits to zero in the input data word when not full precision is needed. Investigations originally presented in [2] and [3] illustrate that the dynamic power consumption of a 16 bit MAC unit can be lowered by more than 50% by

Table 11.1: Power reduction in a MAC unit by data masking

Masked 16-bit MAC		Variable MAC unit precision	
Masked Precision	**Power**	**Data path precision**	**Power**
16-bit (no masking)	1.00	16-bit	1.00
12-bit (mask 4-LSB)	0.47	12-bit	0.41
10-bit (mask 6-LSB)	0.31	10-bit	0.26
8-bit (mask 8-LSB)	0.18	8-bit	0.12
6-bit (mask 10-LSB)	0.09	6-bit	0.06
4-bit (mask 12-LSB)	0.04	4-bit	0.02

only masking 4 bits. The resulting power reduction versus number of masked bits is presented in Table 11.1.

11.3.2 Clock gating

There are two main strategies used when implementing clock gating in a processor design:

- Coarse grained clock gating.
- Fine grained clock gating.

Coarse grained clock gating is used to disable entire blocks in the design such as execution units or memory blocks. Fine grained clock gating is on the other hand applied to individual registers. Coarse grained clock gating is easier to implement than fine grained clock gating since it can be applied to a pre-verified and pre-compiled synchronous block (black box). However, fine grained clock gating has the largest potential for power savings since it can be applied more accurately. An example of coarse grained clock gating is to disable the clock to an entire CMAC unit when no CMAC-operation is executing on the unit while fine grained clock gating disables individual pipeline stages in the data paths. In addition to clock gating, an on-chip phase locked loop together with a voltage

controlled oscillator can be used to vary the clock frequency according to the load of the processor.

11.3.3 Power gating

The idea of power gating is to shut down the power supply for a particular part of a design. This allows the system to reduce the effects of leakage, as unused logic can be turned off. The use of power gating is however not as widespread as the use of clock gating due to a number of reasons:

- There is a lack of reliable design-tools for power-gated logic.
- Power-gated logic has a significant "power-up" delay before the logic becomes functional. Detailed scheduling knowledge is essential in order to power up a power gated execution unit in time for processing deadlines.

However, power gating will be more and more important in the future due to subthreshold leakage in CMOS devices. Instead of switching logic blocks on and off to throttle the performance of the processor, dynamic voltage scaling could be used. Dynamic voltage scaling is used to vary the power supply to logic blocks in order to trade-off their speed with the power consumption.

11.3.4 Integration in the SIMT architecture

The on-chip network used in the SIMT architecture provides possibilities to efficiently introduce coarse grained clock and power gating on execution units attached to the network. This is illustrated in Figure 11.1. Coarse grained power and clock gating are controlled by the controller core whereas fine grained clock gating is performed by local control logic in order to reduce the complexity of the clock control logic.

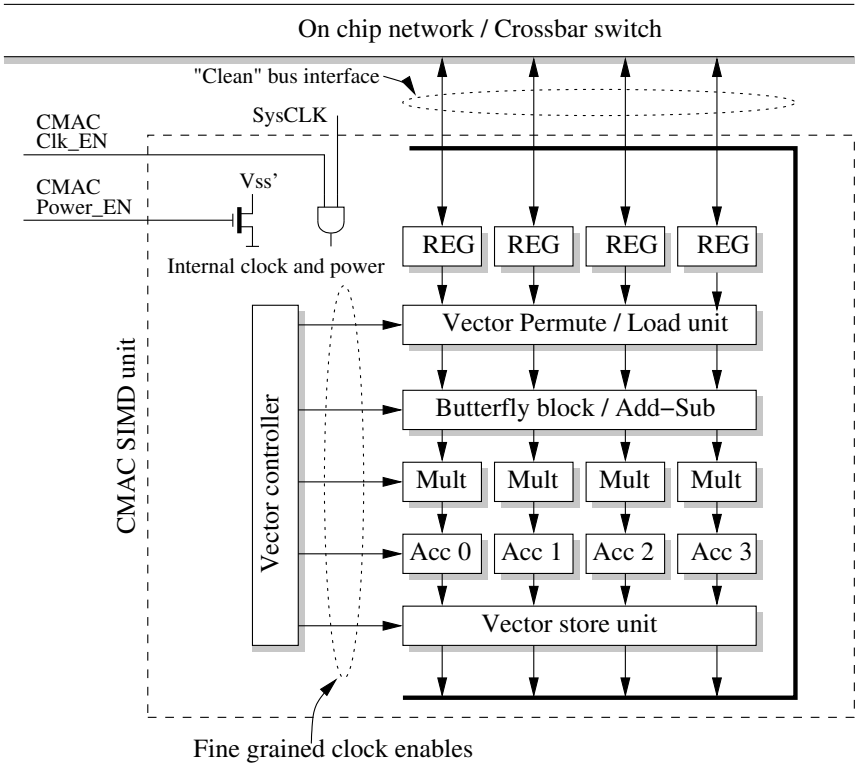

Figure 11.1: Implementation of clock and power gating of execution units in SIMT based processors.

11.4 References

[1] Lidsky D.B.,Rabaey J.M.; *Low-power design of memory intensive functions*, Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium, Oct 1994.

[2] D. Liu and E. Tell, *Low Power Electronics Design*; Chapter 23 - Low Power Programmable Baseband Processors; Ed. Christian Piguet, CRC Press, 2004

[3] E. Tell; *Design of Programmable Baseband Processors*; Ph.D. thesis, Linköping Studies in Science and Technology, Dissertation No. 969, Linköpings universitet, Sep 2005

Chapter 12
Software development

Controlling complexity is the essence of computer programming.
– Brian Kernighan

12.1 Introduction

Efficient tool support is essential for any DSP architecture. A complete tool-set for a DSP processor includes a vectorizing C-compiler, a source code profiler and a complete assembly and simulation flow. As creation of high level software tools such as compilers and source code profilers for parallel DSP processors is a research topic of its own, the work related to software tools used and presented in this dissertation has been limited to an assembler, a bit and cycle true simulator and a C-compiler.

12.2 Software development methodology

The software development flow for the baseband software developed in this research project is illustrated in Figure 12.1. The flow starts with a high level (C, C++, Matlab) source code implementing the desired functionality. The high level source code is then profiled to identify kernel functions, data dependencies and an approximation of the resource usage. This information together with hardware information is used to

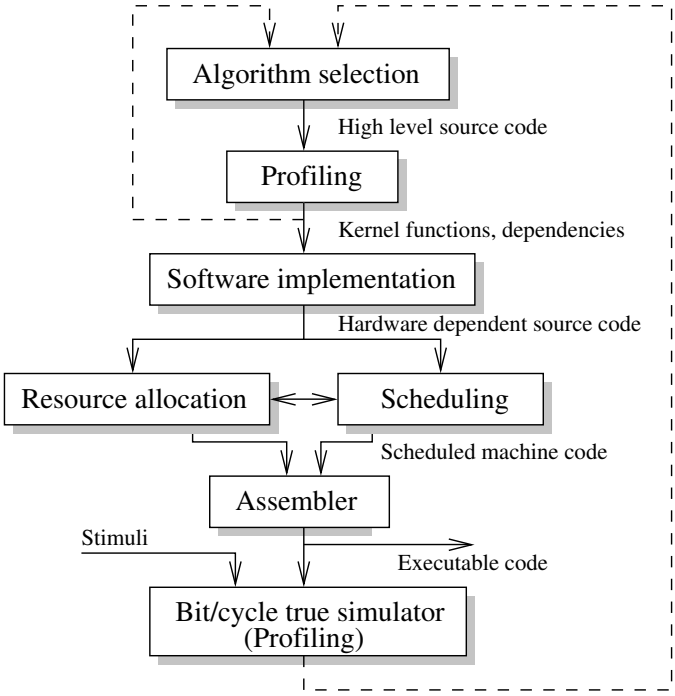

Figure 12.1: Software development flow.

create a hardware dependent source code. The difference between the hardware dependent source code and the original source code is the implementation detail level, e.g. a "FFT" operation in the original source code has been replaced by a more detailed function call such as "radix-4 decimation in time FFT". This code transformation is driven by using hardware templates.

The hardware dependent source code is then processed by a resource allocator and a scheduler before it is finally translated into executable machine code by the assembly flow. As a final step, the code can be used in the bit and cycle true simulator to verify finite word-length effects and detailed scheduling. To aid decisions taken during the software development, profiling and benchmarking are performed during several stages of

the flow. Major feedback paths of information are indicated with dashed lines in Figure 12.1.

12.3 Software development

12.3.1 Algorithm selection

As good algorithm selection is essential for an efficient baseband processing system, this stage is of great importance. During algorithm selection, both finite word-length effects and resource limitations must be considered [1]. Here, the available execution units are used to guide the selection of algorithms to utilize the available hardware maximally.

12.3.2 Profiling and benchmarking

Two different kinds of profilers are used in the software development flow. First, a *source code profiler* is used to identify kernel functions, data dependencies, possibilities for parallelization and approximations of resource usage. The second profiler is the *resource usage* profiler, which is used to estimate the amount of memory accesses, memory usage and cycle cost from the hardware dependent source code. The resource usage profiler is useful when improving algorithms in terms of the number of memory accesses and data dependencies. The difference between the resource usage profiler and the simulator is that the resource usage profiler estimates parameters without actually executing the code. This improves the overall speed of the tool, however at the cost of accuracy.

12.3.3 Hardware dependent behavior modeling

To model algorithms without using a bit and cycle true instruction set simulator, the algorithms can be implemented using bit true arithmetics in C++ for example. This behavior modeling allows algorithm designers to take finite word-length effects into consideration during algorithm

design. By using subroutines with the same bit-true behavior as the vector instructions available in the processor, finite word-length effects can accurately be modeled.

12.3.4 Scheduling

There are two different types of schedulers used for software development in programmable baseband processors. The first type is the static scheduler which is used to statically schedule a program onto a certain hardware. The second type of scheduler is the dynamic scheduler which is used to control the execution of several concurrent programs on a processor. Since static scheduling is performed offline, the scheduling proposed by the scheduling tool can aid the designer to improve the source code at design-time of the program.

To support dynamic multi-standard execution, static scheduling cannot be used, hence there is a need to incorporate operating system functions in the firmware of the processor. Depending on the application, this scheduler can vary from a hundred lines of assembly code to a complete real time operating system. Dynamic scheduling can be performed in many ways, and the most common scheduling principles are discussed in [2]. Scheduling is simplified for both types of schedulers since the runtime and resource usage is known for all instructions and operations in the SIMT architecture. Currently, an automated task scheduler is under development for the SIMT architecture. The task scheduler analyzes the hardware dependent source code and creates a dynamic scheduler that can be used to run simultaneous programs on a SIMT processor.

12.3.5 Simulator and Assembler

A bit and cycle true simulator is essential for software development, algorithm evaluation and system debugging. By using a bit and cycle true simulator, both finite word-length effects and data dependencies due to pipeline delays can be accurately modeled. The simulator used in this project is entirely written in C++ where all architectural components in

the SIMT architecture are implemented as separate classes. Furthermore custom accelerators are easily integrated due to the unified software interface used. The assembler used in this project was implemented using C++ and can generate both binary code for the BBP2 processor as well as symbolic assembly code used together with the simulator for architecture space exploration.

12.3.6 C-compiler

A C-compiler for the SIMT architecture is currently under development. The C-compiler is based on the LCC retargetable compiler [3, 4]. Currently, vector instructions are implemented by means of library function calls and compiler known functions. Together with a resource allocator and the hardware dependent source code flow, an automatic code gener-

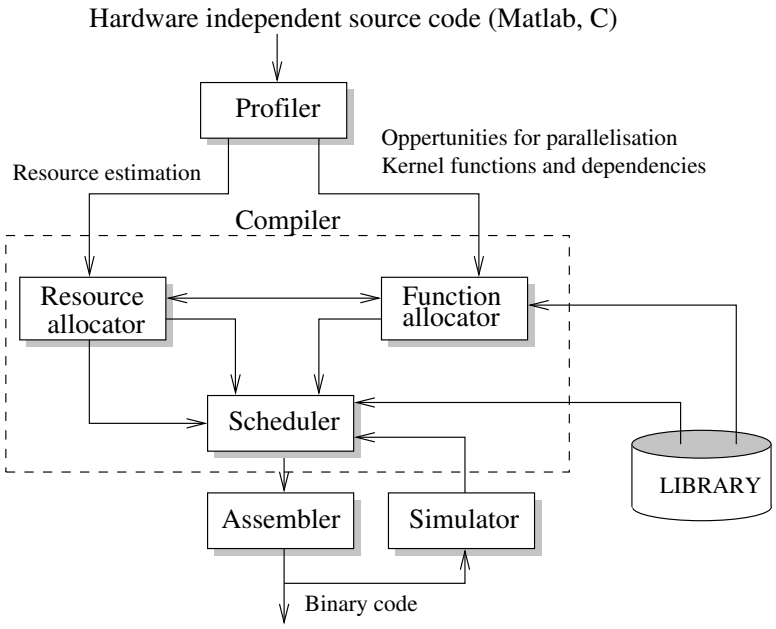

Figure 12.2: Ideal tool-chain for software development.

ating flow could be implemented.

12.3.7 Ideal tool suite

An ideal tool suite for SIMT processors is illustrated in Figure 12.2. By using a function allocator which matches identified kernel functions with functions from a library, vectorization of high level source code could be performed automatically.

12.4 References

[1] Dake Liu; *Embedded DSP Processor Design: Application Specific Instruction Set Processors*; Reading, Elsevier (Morgan Kaufmann), under final editing to be published 2008.

[2] Bacon and Harris: *Operating Systems - Concurrent and Distributed Software Design*, Addison-Wesley, 2003

[3] C. W. Fraser and D. R. Hanson;*A Retargetable C Compiler: Design and Implementation*; Addison-Wesley, 1995. ISBN 0-8053-1670-1.

[4] The World-Wide Web page at `http://www.cs.princeton.edu/software/lcc`

Chapter 13
The BBP2 processor

Design is not just what it looks like and feels like.
Design is how it works.
– Steve Jobs

13.1 Introduction

The SIMT architecture is demonstrated by the BBP2 processor which was fabricated using the ST Microelectronics 0.12µm HCMOS9 process. The BBP2 processor was designed to support symbol processing of a broad range of wireless standards thereby demonstrating the flexibility of the SIMT architecture. Symbol processing of the following diverse wireless standards is supported:

- DVB-T and DVB-H
- WiMAX, IEEE 802.16d,e
- Wireless LAN, IEEE 802.11a,b,g
- WCDMA R6, including HSDPA

The above-mentioned standards include OFDM, OFDMA and CDMA modulation schemes.

13.2 Architecture

To support the previously listed standards the BBP2 processor is equipped with the following core components:

- Five complex valued memory banks organized as four banks of 8k complex words (16+16 bit) and one larger bank of 10k complex words. Each bank is divided into four memory blocks enabling read and write of four consecutive complex values each clock cycle.
- One integer memory of 4k $\times$ 16 bit used for data buffering between the Media Access Control layer and the processor.
- A controller core using 24 bit instructions with multi-context support, a real valued multiplier and a 512 byte integer stack.
- A 4-Way Complex Multiply-accumulate SIMD unit capable of performing one radix-4 FFT butterfly per clock cycle.
- A 4-Way Complex ALU SIMD unit capable of performing multi-code de-spread in WCDMA.

The components were selected according to the methodology presented in Chapter 9. The target maximum frequency of the processor was set to 240 MHz. With the above mentioned configuration the BBP2 processor achieves the following performance:

- 1.92 G complex OP/s, @ 240 MHz (Complex MAC, Complex Arithmetics).
- Memory bandwidth of 153.6 Gbit/s @ 240 MHz. (640 bits/CC)

The complete processor architecture is illustrated in Figure 13.1.

13.2.1 Instruction set

The BBP2 processor has 158 instructions divided among:

- 92 RISC instructions.

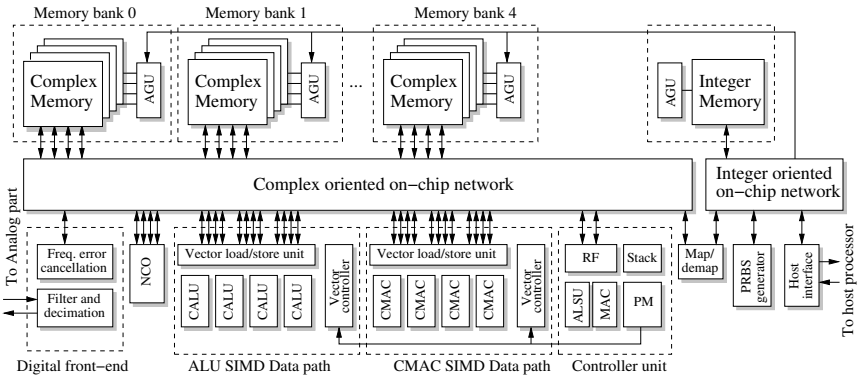

Figure 13.1: The BBP2 processor architecture.

- 22 Complex ALU instructions.
- 30 Complex MAC instructions.
- 14 Load / Store instructions.

All instructions are 24 bit wide. The branch instructions have up to two delay-slots in the RISC core. As the pipeline is exposed, data dependency checking has to be performed during compile time or during simulation of the code. Vector operations have 3 to 5 pipeline stages.

13.2.2 On-chip network

The on-chip network is implemented as two separate networks, where the complex valued crossbar switch has 15 ports (10×128 bit ports and 5×32 bit ports) and the integer network has 9×16 bit ports.

13.2.3 Execution units

As stated earlier, the BBP2 processor is equipped with a 4-Way CMAC and a 4-Way CALU SIMD unit.

Vector CMAC: The Complex MAC execution unit is capable of performing a number of different operations on its 4 Complex MAC lanes.

The unit can execute a radix-4 butterfly or two parallel radix-2 operations per clock cycle in addition to vector multiplication and similar operations. The execution unit also supports various other transforms such as Modified Walsh Transforms (MWT) and Discrete Cosine Transform (DCT). Each data-path within the unit uses 14×14 bit complex multipliers and the SIMD unit has eight 2×40 bit accumulator registers.

Vector ALU: The complex ALU unit is similar to the CMAC unit except for the multipliers that are replaced by a "short" complex multiplier capable of only multiplying by $\{0, \pm 1; 0, \pm i\}$. The short complex multipliers are further described in Section 8.4.3. Along with address and code generators, this unit can efficiently be used to perform Rake finger processing and de-spread in WCDMA and DSSS systems in addition to normal add/subtract operations. Each data-path has 2×24 bit accumulators.

13.2.4 Accelerators

The following functionality is accelerated using configurable accelerators:

- Digital front-end functionality such as filtering, decimation, sample-rate conversion and I/Q impairment compensation. The digital front-end also contains a packet detector to wake up the processor core when a packet arrives.
- A 4-way Numerically Controlled Oscillator (NCO) used to provide coefficients for the CMAC unit during phase error correction calculations.
- A map/de-map unit capable of automatically map/de-map vector data.
- A PRBS generator used to index pilot tones in OFDM symbols.

Accelerators were selected using to the methodology presented in Chapter 6.

Table 13.1: Kernel benchmarks on the BBP2 processor. The presented cycle cost includes all set-up costs for memory addressing

Operation (complex valued data)	Clock cycles
4k point FFT	6186
4k sample vector multiplication	1027
32 sample dot product	11
256 sample correlation	4288
64 sample square absolute value	22
8k sample absolute maximum search	2051
64 sample de-spread of four OVSF codes	70

13.3 Kernel benchmarking

To illustrate the effectiveness of the SIMT architecture kernel benchmarks of the BBP2 processor are presented in Table 13.1.

Since each vector operation only requires one 24 bit instruction word, the code density is very high. A complete 8192 sample FFT routine requires only 16 assembly instructions. A complex dot-product or vector multiplication can be performed using only one instruction word. Processing efficiency is further improved by the SIMT concept since multiple (up to two in BBP2) vector operations can be executed in parallel.

13.4 Implementation

The BBP2 processor was implemented in VHDL. Synthesis and back-end were performed by using the Synopsys Design Compiler and Synopsys Astro tools respectively. Mentor Graphics Modelsim was used for RTL simulations and post-synthesis simulations. The BBP2 processor was taped-out in an ST 0.12 µm dual Vt process early 2007. A layout snapshot is presented in Figure 13.2. The physical properties of the BBP2 processor are summarized in Table 13.2.

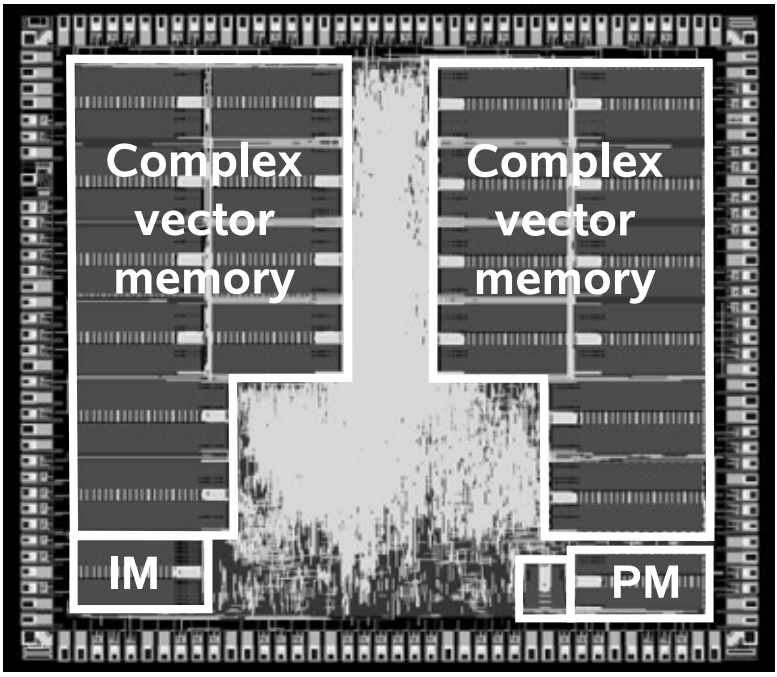

Figure 13.2: Layout snapshot of the BBP2 processor. IM denotes the integer memory. Note the size of the Program Memory (PM) which is enough for managing Wireless LAN and DVB at the same time. The core area including memories is 11 mm^2 in a 0.12 µm process.

13.4.1 Cell area of individual components

The cell area of each individual core component is presented in Table 13.3. Worth noticing is the low complexity of the controller core and the low percentage of the total cell area that is used for on-chip interconnect. In total, only 4.3 % of all gates are located in the crossbar switch.

13.4.2 Clock and power gating

The SIMT architecture provides excellent opportunities for both clock and power gating at the boundary of the on-chip network (as discussed in

Table 13.2: Features of the BBP2 test chip.

Feature	Value
Target frequency	240 MHz
Number of gates	201 k (NAND2)
Core area including memories	11 mm^2
Memory area	6 mm^2 (1.49 Mbit)
Total area	17 mm^2 (including pads)
Total number of pins	155
Number of IO pins	94
Package	160 pin TQFP

Section 11.3.2). Since the network and execution units are under strict program control, clock and power gating can easily be applied. However, due to lack of proper back-end tool support for clock and power gating, these techniques were not implemented in the BBP2 test chip.

13.5 System demonstrator

To aid the algorithm development and to provide a platform for system demonstration, a demonstration system comprising a number of exchangeable radio modules and interface modules connected to a commercial "off-the-shelf" FPGA board was developed. An overview of the system is illustrated in Figure 13.3. The modular design allows easy plug and play of radio modules. Currently radio modules for the following standards are available:

- WCDMA (receive only).
- Multi-mode entertainment: DVB-T/H, DMB, DAB, FM, AM
- WLAN - IEEE 802.11abg (2.5 GHzand 5 GHz)
- Analog baseband (I/Q).

Table 13.3: Gate count and relative area of BBP2 core components

Unit	Area [kGates]	Relative area [%]
Controller core (Including program flow controller)	26.84	13.1
Complex ALU SIMD unit	20.01	9.8
Complex MAC SIMD unit	61.29	29.8
Complex memory AGU	5×2.53	5×1.7
Integer memory AGU	0.22	0.1
Complex network	8.61	4.2
Integer network	1.02	0.1
NCO	3.74	1.8
Map/De-map	2.35	1.1
Front-end, filter	6.12	3.0
Front-end, farrow	20.05	9.8
Front-end, misc.	33.99	16.5
Host interface	1.64	0.8
PRBS Address generator	2.38	1.2
Total	200.91 kGates (NAND2)	100 %

A photo of the demonstrator system is shown in Figure 13.4.

13.6 Measurement results

Unfortunately, due to a large delay in the fabrication of the BBP2 chips, the chips have not arrived at the time of publication of this thesis. Hence, no measurement results are available.

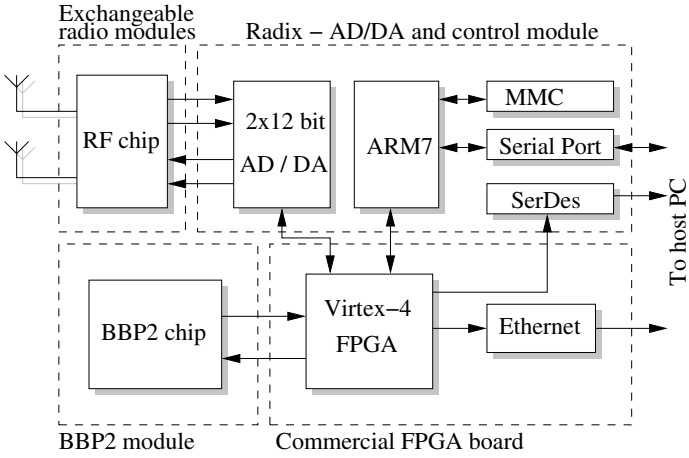

Figure 13.3: Overview of the demonstration and data-acquisition system.

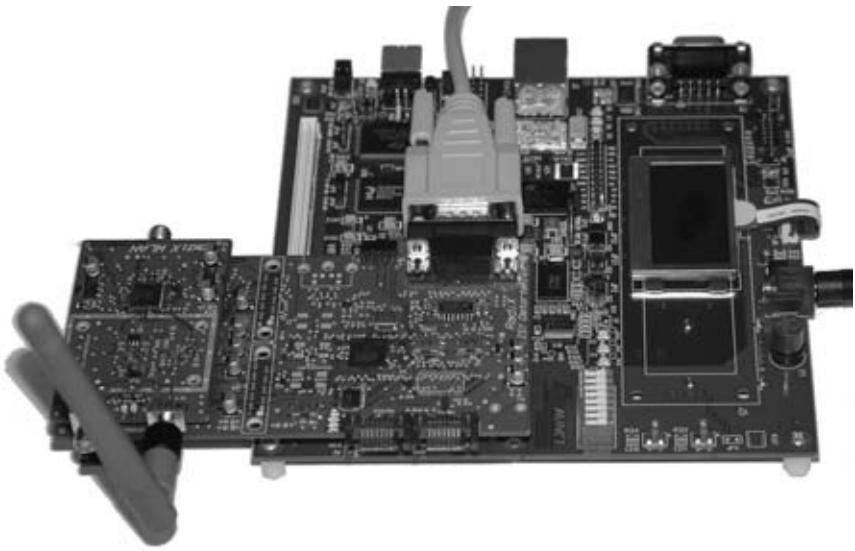

Figure 13.4: The demonstration and data-acquisition system.

13.7 Scaling

The SIMT architecture including the BBP2 processor will scale well with new technologies. Since the processor is fully synthesizeable and operates at moderate frequencies (most low-leakage cells) the computing capacity can be increased by increasing the operating frequency in a process with smaller geometries. The clock frequency can be traded against extra execution units to provide a good tradeoff between dynamic and leakage power consumption in modern technologies (65 nm and beyond).

13.8 References

[1] Anders Nilsson and Dake Liu; *Area efficient fully programmable baseband processors*; In proceedings of International Symposium on Systems, Architectures, Modeling and Simulation (SAMOS) VII Workshop; July 2007, Samos, Greece.

Chapter 14
Verification and Emulation

14.1 Introduction

In this chapter, a verification methodology for baseband processors is described. The methodology has successfully been used during the design and verification of the BBP2 processor.

14.2 Tool-chain for verification

The tool-chain used for verification of both design-tools, RTL-code and manufactured chips contains the following tools and models [1]:

- A high level executable model. This model is used as a "Golden Reference" model during verification of the other models, tools and devices.

- A bit and cycle true simulator. The simulator is used during processor and algorithm development as well as during verification of the RTL-code and the manufactured chip.

- An RTL simulator. The RTL simulator is used to simulate the RTL-code and interface the RTL-code with the rest of the tool-chain. By using a test bench that easily interfaces with the bit and cycle true

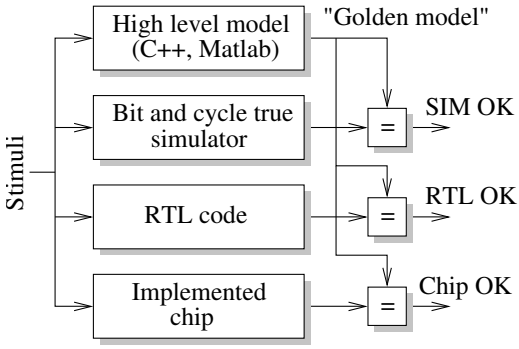

Figure 14.1: Hardware verification flow.

simulator and provides methods for batch tests, the verification speed is improved.

- A stimuli generator. The stimuli generator is used to create input stimuli to all simulators and test-beds.

14.3 Verification methodology

The verification methodology is illustrated in Figure 14.1. The presented methodology aims at verifying congruence between all models and the RTL-code, not between the RTL-code and the synthesized net-list as there are formal methods for this. The verification flow and tools can also be used to verify the functionality of the manufactured chip. The presented verification flow is dependent on good input stimuli to provide good verification coverage. The verification coverage is analyzed in the RTL simulator and used to perform targeted tests in order to ensure a full verification coverage (all lines of source code reached). Normally four different stimuli types are used:

Application data: The first type of stimuli is the application data, i.e. normal data that are representative for the data processed in the system.

This can for example be a complete OFDM frame or data representing the service channel in a cellular system.

Corner cases: Corner case tests use specially crafted stimuli to provoke a certain condition (corner case) such as overflow or saturation in the processor.

Random data: During random stimuli tests, all input signals are randomized. Randomized tests are useful to find behavioral differences between the simulator and the device under test when the system operates outside its normal mode.

Directed random data: In contrast to fully random testing, a directed random test utilizes random stimuli within a certain boundary, e.g. legal instruction codes or valid bus transactions.

The stimuli are created by the stimuli generator using either high level system models, system simulation tools or real data recorded from a live system. The usage of recorded data are most used during system level verification and validation since it can be used to find problems that have been overlooked during the creation of the system models.

14.3.1 Formal verification

In addition to traditional verification, formal verification techniques can also be used to verify the models and RTL code. The idea used in formal verification is to prove the congruence of the RTL code and the source code by formal means, i.e. mathematically and logically. Due to the limitations of formal verification, the method is mostly used to check the source RTL code with the synthesized net-list.

14.4 Lab installation

To verify the functionality of the manufactured processor, a pattern generator and a logic analyzer are used to provide stimuli to the test chip

and capture output signals from the chip at full clock rate. At Linköping University, a combined pattern generator and logic analyzer (Tektronix TLA721) are used to interface the device under test. The TLA721 has the possibility to be remotely controlled in order to automate test and verification. This lab-setup is used together with the demonstrator hardware developed for the BBP2 processor, as described in Section 13.5, to record real air data which can be used for verification and system validation.

14.5 Emulation

During the design of a baseband processor, there are many occasions where it is useful to test different parameters of a design under realistic system conditions. This is especially important when designing hardware and algorithms used for closed loop systems where hardware interaction is needed, such as AGC and similar tasks. These tasks are very hard to accurately simulate since the behavior of the entire system including the radio front-end, analog filters and ADC must be modeled.

As an alternative to a full system simulation, the functionality of the RTL-code can be implemented in a programmable logic device such as a FPGA. However, due to differences of the available logic functionality (ASIC standard cells versus FPGA lookup tables), the FPGA implementation will seldom reach the same operation frequency as the ASIC implementation. Nevertheless, a system operating at reduced frequency is still useful in most cases.

14.6 References

[1] *Writing Testbenches: Functional Verification of HDL Models*; Kluwer Academic Publishers, ISBN 9781402074011

Part IV

Extensions of the SIMT architecture

Chapter 15
MIMO and Multicore support

15.1 Introduction

MIMO support and multicore processor systems are definitely of interest in modern baseband processing systems. Although these topics are not the main focus of this dissertation some work has been performed to ensure applicability of the SIMT architecture to these research topics.

15.2 MIMO

Multiple In - Multiple Out (MIMO) technology will play an important role in future wireless standards since it can be used to improve the spectral efficiency (bits/Hz) and the reliability of a wireless channel. MIMO technology provides three advantages compared to conventional systems:

Spatial diversity. This is the most important advantage of a MIMO system. Spatial diversity utilizes spatially separated paths to transmit different data on different paths. This is used to improve the overall throughput of a wireless system.

Beam forming. By using beam-forming in the transmitter, the transmit-

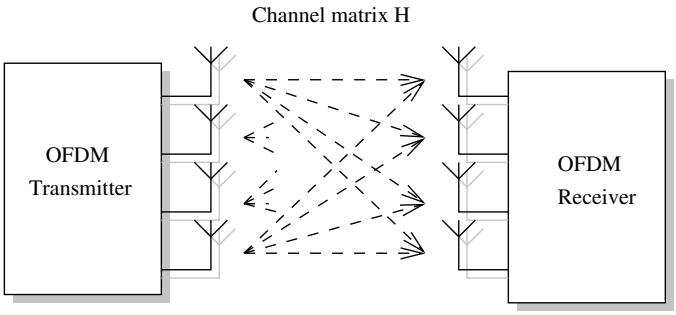

Figure 15.1: Example of a 4 by 4 MIMO-OFDM system.

ted energy is better utilized and interference is reduced in other directions.

Maximum ratio combining. By using MRC in the receiver, fading due to multi-path propagation can be mitigated more efficiently than in a single antenna system.

In Figure 15.1 a 4×4 MIMO system is illustrated. The transfer function from transmitter antenna $i = 0..N_t$ to the receiving antenna $j = 0..N_r$ is given by an $i \times j$ matrix. In order for the receiver to recover the transmitted data the $i \times j$ matrix must be inverted. In a MIMO-OFDM system, this is performed for each sub-carrier.

In order to investigate the implementation of MIMO-OFDM systems using SIMT baseband processors, high level mappings of a MIMO-OFDM system have been performed onto the SIMT architecture. The mapping results show that only a matrix-manipulation unit must be added to the SIMT architecture. This unit is however seamlessly integrated in the SIMT architecture since it can either be used as an accelerator or an execution unit connected to the on-chip network.

The processing flow in a MIMO-OFDM system [1] is illustrated in Figure 15.2.

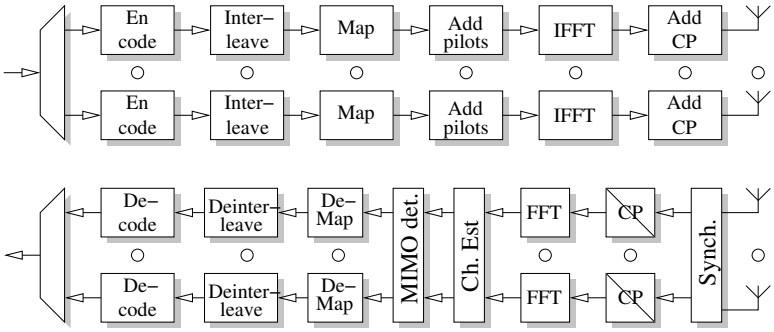

Figure 15.2: Processing flow in a MIMO-OFDM system.

15.2.1 Front-end processing

The most obvious changes that have to be made in a baseband processor in order to support MIMO functionality are the extra radio front-ends needed for each antenna. Since the same operation (e.g. filtering, decimation and frequency offset correction) will be applied to data from all antennas, resource sharing can be applied. Also, since the signal arriving to each radio front-end will have the same common carrier frequency and sample frequency error, estimation of these errors is only needed to be performed for one antenna.

15.2.2 Channel estimation

As the computing load is much in higher in the receiver than in the transmitter, only the receiver is considered here. The amount of raw data processed by the receiver is proportional to the number of receiver antennas (N_r). The channel estimator must estimate the channel matrix H, see Figure 15.1, which describes the transfer function from each transmitting antenna to each receiving antenna. This is performed by encoding orthogonal information in the preamble which is transmitted from each antenna.

15.2.3 Matrix inversion

The most computationally demanding step in a normal MIMO-OFDM system is the inversion of the channel matrix H. In a 4×4 MIMO system, the inversion of a 4 by 4 complex matrix for each sub-carrier in the OFDM symbol must be performed in the same time as a normal OFDM system performs channel compensation.

Matrix inversion is preferably performed in a programmable accelerator attached to the complex on-chip network in SIMT processors. As matrix inversion requires a large internal dynamic range, the execution unit should preferably be implemented using floating point arithmetic. However, the rest of the baseband processor should still be implemented using fixed point arithmetic for silicon efficiency.

15.2.4 Integration in the SIMT architecture

The usage of an on-chip network in the SIMT architecture allows multiple radio front-ends to be attached to the network without penalty. In the same way the matrix inversion function can be implemented either as an execution unit or as an accelerator, both attached to the complex on-chip-network. By attaching the matrix manipulation unit to the on-chip network, the unit can access main memory in the same way as normal SIMD execution units, thus working in parallel with the SIMD execution units of the processor.

15.3 Multicore support

Even though most baseband algorithms can be mapped to one SIMT enabled processor more parallelism is sometimes needed. Certain baseband problems are too complex to handle with only one or a small number of processor cores; they need the capacity of a large number of processors working in parallel. Examples of such systems are xDSL line-cards, radio base stations and radar systems.

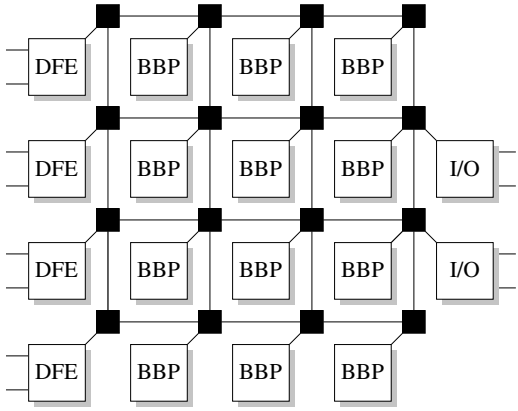

Figure 15.3: Example of a multicore system.

Network-on-Chip (NoC) is a hot research topic and several different approaches exist. In [3] the SoCBUS NoC is presented along with a mapping of a multi-core baseband system. From a designer point of view, the network on chip can be considered to be a standard I/O-port as the complexity is hidden in the network adapter and the core wrapper provided by the NOC.

An example of a system implementing a MIMO-OFDM receiver [1] with 12 baseband processor (BBP) cores, 4 digital front-end cores and two I/O cores is illustrated in Figure 15.3

15.3.1 Memory management

As baseband processing in the application mentioned above can be parallelized and split into individual tasks, there is no need to maintain a global addressing space in the multi-core system. Instead, data blocks are handed over between the different processor nodes in a *data stream* fashion, much like the internal processing flow within a single SIMT processor [3].

15.3.2 Integration in the SIMT architecture

A network adapter is used to interface the internal on-chip network of the SIMT processor with the global on-chip network. The network adapter is a master unit (like the SIMD units) on the internal on-chip network in the SIMT processor. This allows the network adapter to read and write to memory banks without intervention from the processor core.

15.4 References

[1] H. Jiao, A. Nilsson and D. Liu; *MIPS Cost Estimation for OFDM-VBLAST systems*; IEEE Wireless Communications and Networking Conference, Las Vegas, NV, USA, Apr 2006

[2] J. Eilert, D. Wu and D. Liu; *Efficient Complex Matrix Inversion for MIMO Software Defined Radio*; to be presented on IEEE ISCAS, USA, 2007

[3] D. Wiklund; *Development and Performance Evaluation of Networks on Chip*; Ph.D. thesis, Linköping Studies in Science and Technology, Dissertation No. 932, Linköping University, 2005

[4] A Hottinen, O. Tirkkonen,R. Wichman; *Multi-antenna transceiver Techniques for 3G and Beyond*; John Wiley & Sons; 2003; ISBN 0470 84542 2

Part V

Conclusions and future work

Chapter 16
Conclusions

To succeed, jump as quickly at opportunities as you do at conclusions.
– Benjamin Franklin

Programmability is essential for multi-standard baseband processors. To be able to efficiently support multiple standards within a processing device, new processor architectures are necessary. As a response to this, this dissertation presents the SIMT architecture, which is a SIMD based architecture utilizing the vector property of baseband processing tasks to reduce the control overhead in the processor while maintaining flexibility and computing throughput. The reduction of control overhead reduces both the silicon area of the processor and the power consumption of the system.

The SIMT architecture is a result of research focused on the following six topics, all aimed at a solid foundation for development of a flexible yet power efficient processor architecture:

- Algorithm selection and development.
- Design of execution units and memory systems for multi-standard signal processing.
- Instruction issue and memory architectures.

- Hardware and software considerations for simultaneous multi-standard execution.
- Selection, specification and design of accelerators.
- Specification and implementation of simulation environments.

16.1 Achievements

As stated earlier, the goal of this research project has been to create a new power- and area efficient programmable baseband processor architecture, suitable for multi-standard operation. All achievements presented below are results of research with this goal in mind.

16.1.1 Algorithm selection and development

Along with the other activities in the project, much research effort has been spent on gaining an understanding of how to select algorithms suitable both for a certain type of problem, but also suitable for a certain architecture. Here the result of this algorithm research has been incorporated in the design and implementation of all hardware and all architectures presented in this thesis. Algorithms have been studied, selected, designed and implemented for both OFDM and CDMA transmission, reception and channel compensation.

16.1.2 Models

To be able to create correct and realistic models of baseband processing systems and transmission channels, much effort has been spent on understanding both the properties of the radio front-end and the channel, but also the requirements imposed on the system by the MAC layer. Here I have created high-level models as well as bit- and cycle-true models, which embody all relevant impairments and constraints imposed by the system environment. These models have been used as a foundation for

the design of the processor architecture. Furthermore, the results of the algorithm related research have been incorporated in all models.

16.1.3 Accelerator selection

A large research effort has been spent on accelerator selection and design. In this area I have formulated a methodology for algorithm and accelerator selection and have specified, designed and implemented multi-standard accelerators for WiMAX, Wireless LAN and WCDMA systems. Special care has been taken to ensure multi-standard support and flexibility in the designed and implemented accelerators.

16.1.4 Instruction issue

A fundamental property of the SIMT baseband processor architecture is the single-issue concept that has been refined since the first BBP generation. The concept has been extended and transformed into a new architecture with many clustered SIMD execution units.

16.1.5 Simultaneous multi-standard execution

Analysis of scheduling techniques as well as design of hardware support for simultaneous multi-standard execution have also been performed in this research project.

16.1.6 SIMT components

A large part of the research effort has been spent on designing and dimensioning of execution units, memory system and controller core for a multi-standard baseband processor. A controller core with multiple context support has been designed to simplify the scheduling of operations in parallel with vector operations. Investigations of execution unit design have also been performed, for example Rake receiver functions have been mapped to a versatile short-complex-MAC unit, allowing a SIMT baseband processor to run the Rake receiver algorithms smoothly.

16.1.7 System demonstrator

To demonstrate the SIMT architecture, the BBP2 processor has been designed using SIMT technology. The demonstrator chip was manufactured early 2007 and implements a full scale system with multiple SIMD execution units and a controller core supporting multiple threads. The design of the BBP2 processor included all steps from algorithm selection of all considered standards to ASIC back-end integration. The BBP2 processor is mainly targeted at demonstrating symbol processing of DVB-H/T, WiMAX, IEEE 802.11a/b/g and WCDMA. The core area of the BBP2 processor is 11 mm^2 in a 0.12 $\mu\mathrm{m}$ technology.

Unfortunately, due to a large unexpected delay in the fabrication of the BBP2 chips, the chips have not arrived at the time of publication of this thesis. Hence, no measurement results are available.

16.2 The SIMT architecture

The presented SIMT architecture provides:

- High processing parallelism
- Low control-path overhead and high code density
- High hardware reuse

which enables low silicon area (low cost) and low clock rate implementations of programmable baseband processors. The low control overhead, high code density and the memory system (including the on-chip network) all contribute to the low silicon area and high memory efficiency.

Unlike VLIW-machines, the SIMT architecture will allow concurrent vector operations and control flow instructions, without the drawbacks and code memory usage of a pure VLIW machine, which further improves the memory efficiency. The use of vector instructions are enabled by the underlying architecture which consists of:

- Vector execution units (Complex MACs and Complex ALUs).

- Small memory blocks with de-centralized address generators.
- An on-chip network.
- A scalar data path executing RISC instructions.
- Accelerators for certain tasks.

To conclude, the SIMT architecture provides a flexible yet efficient platform for multi-standard baseband processing combining benefits from both VLIW- and SIMD-based DSPs. Relying on the large fraction of vector based processing in baseband processing, it achieves a high degree of parallelism combined with low control overhead and compact programs by use of the SIMT technology, which enables future efficient multi-standard wireless terminals.

Chapter 17
Future work

During the work with this project, a number of interesting topics for a continuation has been identified. Three different topics were of extra interest for the research area of computer engineering and especially in the area of design of application specific instruction set processors for baseband processing.

The most obvious continuation of this project is to investigate how to efficiently implement a fully programmable bit manipulation and forward error correction processor. Other areas of interest are to investigate how large scale MIMO systems could be mapped to one or several SIMT based processors and how to use multiple SIMT processor cores in a multicore system.

17.1 Multi-standard FEC processors

This research project has focused on creating flexible yet area and power efficient baseband processor architectures for symbol processing. To provide flexible baseband processors managing all processing steps between the antenna and the application processor, bit manipulation and forward error correction operations must also be addressed. This requires research and development of new processor architectures which are capable of handling these tasks in an efficient way.

17.2 MIMO

As MIMO technology most certainly will be used in modern wireless standards, efficient support of MIMO-related operations are necessary. Research of both algorithms and hardware architectures at the same time are necessary to be able to design MIMO systems that are efficient yet feasible for implementation in real systems.

17.3 Multi-core systems

Certain baseband problems are too complex to handle with only one or a small number of processor cores – they need the capacity of a large number of processors working in parallel. Two examples of systems requiring >100 cores are base-stations for future wireless systems and radar systems.

17.3.1 Wireless base-stations

By implementing more and more functionality in a base-station with fully programmable baseband processors, agile solutions can be implemented. This agility can be used to allow the base-station to support a number of different standards at the same time allowing by re-distribution of computing resources to match the instantaneous need of all connected users. A flexible implementation would also prolong the service life of a base-station since it can be software upgradeable in a larger scale than today. Early estimates of the processing needs in such base station, indicate the need for more than 100 baseband processor cores to manage symbol processing and forward error correction for a complete cell.

17.3.2 Radar systems

A completely different application area for multicore baseband processors is the area of radar signal processing. Radar processing is characterized

by the large amount of data needed to be processed. However, the program flow of radar processing programs is often fully deterministic which allows the signal processing to take advantage of vector processing.

17.4 Concluding remarks

Programmable baseband processors are important in future wireless systems. The SIMT architecture provides a solid foundation for continuing research within the area of programmable baseband processors. By extending the SIMT concept, many new and interesting applications can be targeted for future research.