Towards Software Defined Radios Using Coarse-Grained Reconfigurable Hardware

Gerard K. Rauwerda, Paul M. Heysters, and Gerard J. M. Smit

***Abstract*—Mobile wireless terminals tend to become multimode wireless communication devices. Furthermore, these devices become adaptive. Heterogeneous reconfigurable hardware provides the flexibility, performance, and efficiency to enable the implementation of these devices. The implementation of a wideband code division multiple access and an orthogonal frequency division multiplexing receiver using the same coarse-grained reconfigurable MONTIUM tile processor is discussed. Besides the baseband processing part of the receiver, the same reconfigurable processor has also been used to implement Viterbi and Turbo channel decoders.**

***Index Terms*—Heterogeneous reconfigurable hardware, orthogonal frequency division multiplexing (OFDM), software defined radio (SDR), system-on-chip (SoC), turbo decoding, viterbi, wideband code division multiple access (WCDMA).**

I. INTRODUCTION

FUTURE wireless communication systems tend to become multimode, multifunctional devices. Adaptivity becomes more important now then ever. These systems have to adapt to changing environmental conditions (e.g., more or less users in a cell or varying noise figures due to reflections or user movements) as well as to changing user demands [bandwidth, traffic patterns, and quality-of-service (QoS)]. When the system can adapt (at run-time) to the environment, significant savings in computational costs can be obtained [3], [4]. Furthermore, the hardware architectures have to be extremely efficient as these are used in battery-operated terminals and have to be cost effective as they are used in consumer products.

Heterogeneous reconfigurable hardware platforms offer the necessary flexibility for performing multiple wireless communication standards and can achieve the performance required by the wireless standards. Furthermore, the combination of mixed-grained reconfigurable solutions enables energy efficient implementations of the wireless standards. Much work has been done on software defined radio (SDR) in the SDR forum context.[1]

One of the main reasons for introducing reconfigurable hardware in a wireless terminal is to support multiple wireless communication standards. The support of multiple wireless communication standards introduces a first level of adaptivity in the wireless terminal because the terminal can switch between wireless communication standards. For example, when packet data transport is performed over Universal Mobile Telecommunications System (UMTS) and a wireless local area network (WLAN) hotspot becomes available the terminal can switch from UMTS to a WLAN standard. This is referred to as *standards level* adaptivity. Standards level adaptivity has an impact on the digital signal processing (DSP) in the wireless terminal because the wireless communication standard defines the DSP functions that have to be performed to implement the standard [5].

Although a wireless communication standard usually defines the DSP functionality, which has to be performed to implement the standard, it usually does not define the algorithms that have to be used to implement these functions. So, the communication system can, therefore, "adapt the algorithms" that are used to implement the DSP functionality. "Adapt the algorithms" means that the communication system selects an algorithm from a set of algorithms that implement the same DSP functionality. Therefore, this second level of adaptivity is referred to as *algorithm-selection level* adaptivity [5].

For a specific algorithm, there are also opportunities for adaptivity by changing parameters of the algorithm. This third level of adaptivity is called *algorithm-parameter level* adaptivity [5].

Dynamic reconfiguration of hardware is required in order to have real adaptive systems. The rate of reconfiguration depends on the levels of adaptivity that is addressed by the receiver. Hence, the *algorithm-parameter level* will be more frequently addressed than the *algorithm-selection level*. The reconfiguration rate is highly dependent on the operating environment. The *standards level* is due to interaction with the end-user. For instance, the standard selected by the user changes on a minute or hour rate, while the parameters of a standard can change on a second rate, influenced by the quality of, e.g., the wireless channel.

In this paper, we discuss the implementation of wireless communication systems on heterogeneous dynamically reconfigurable hardware. The implementation of a flexible RAKE receiver, used for UMTS communications, and the implementation of an orthogonal frequency division multiplexing (OFDM)

Manuscript received May 10, 2006; revised May 6, 2007. This work was supported in part by the EU-FP6 project 4S (Smart Chips for Smart Surroundings) (IST-001908) and the Freeband Knowledge Impulse Programme, a joint initiative of the Dutch Ministry of Economic Affairs, knowledge institutions and industry. A preliminary version of this paper was presented at the Proceedings of the International Conference on Engineering and Reconfigurable Systems Algorithms (ERSA), 2005 and 2006.

G. K. Rauwerda is with Recore Systems, 7500 AB Enschede, The Netherlands and also with University of Twente, Department of Electrical Engineering, Mathematics, and Computer Science, 7500 AB Enschede, The Netherlands (e-mail: gerard.rauwerda@recoresystems.com).

P. M. Heysters is with Recore Systems, 7500 AB Enschede, The Netherlands (e-mail: paul.heysters@recoresystems.com).

G. J. M. Smit is with Department of Electrical Engineering, Mathematics, and Computer Science, University of Twente, 7500 AB Enschede, The Netherlands (e-mail: g.j.m.smit@utwente.nl).

Digital Object Identifier 10.1109/TVLSI.2007.912075

[1]SDR forum. [Online]. Available: http://www.sdrforum.org

receiver, used in HiperLAN/2, is studied to show the feasibility of implementing multimode communication systems using dynamically reconfigurable hardware.

Besides baseband processing, the presented reconfigurable architecture is also used to implement channel decoding algorithms. The implementation of flexible Viterbi and Turbo decoders is discussed. These channel decoders have been applied in many wireless communication standards, using slightly different settings for each standard.

Section II introduces the heterogeneous reconfigurable system-on-chip (SoC) template. The coarse-grained reconfigurable processing elements in the SoC are implemented by MONTIUM processing tiles. The application domain of the proposed SoC template is explained in Section III. Examples of applications, which are intended to be mapped on reconfigurable hardware, are baseband functionality of wideband code division multiple access (WCDMA) and OFDM receivers, and channel decoders. The implementation results of DSP kernels in reconfigurable hardware are presented in Section IV. In Section V, conclusions are drawn on the presented work.

II. RECONFIGURABLE HETEROGENEOUS ARCHITECTURE

Implementation of SDR requires a flexible hardware architecture. Traditional SDR approaches are implemented on *homogeneous* flexible architectures, like general purpose processors (GPPs) or digital signal processors (DSPs) [6]. Since baseband processing in the wireless receiver is computationally intensive, the terminal's hardware architecture has to be quite powerful. Moreover, as wireless terminals are battery-powered, the importance of energy efficiency of the hardware architecture is emphasized.

A common drawback of the traditional homogeneous flexible architecture is its relative energy inefficiency. Whereas, *heterogeneous* reconfigurable hardware, consisting of processing elements of *different* granularities, is designed with these constraints—flexibility, performance, and energy efficiency—in mind.

A. Chameleon SoC Template

The idea of heterogeneous processing elements is that one can match the granularity of the algorithms with the granularity of the hardware. Four processor types can be distinguished: *general purpose, fine-grained* [e.g., field-programmable gate array (FPGA)], *coarse-grained* (e.g., MONTIUM [7], [8]), and *dedicated* [e.g., application-specific integrated circuit (ASIC)]. Fig. 1 depicts a heterogeneous reconfigurable hardware template, consisting of processing elements of different granularities. Matching the granularity of the reconfigurable hardware with the algorithm provides flexibility at the right level.

- *General Purpose.* The general purpose processor is the most flexible hardware architecture. It can be programmed to perform almost any algorithm. General purpose processors are well suited for control-oriented functions. Due to the large overhead in control, these processors are not very energy efficient.
- *Fine-Grained.* Fine-grained reconfigurable devices are bit-level programmable. Because of the configurability at bit-level, the configuration overhead is large. Fine-grained reconfigurable devices are perfectly suited for prototyping and implementing encryption algorithms.
- *Coarse-Grained.* Coarse-grained reconfigurable devices are flexible at word-level. Multipliers, adders, etc., are hardwired in these devices. Because only coarse functional blocks have to be configured, the configuration overhead is small. These architectures are more suited for data-oriented functions, like algorithms performed in the DSP domain.

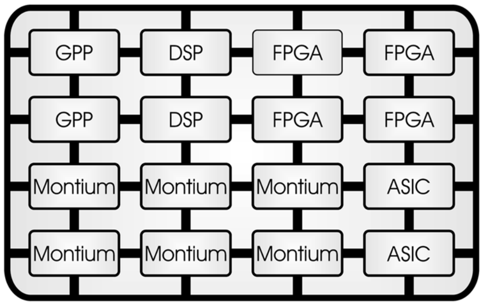


Fig. 1. Chameleon SoC template.

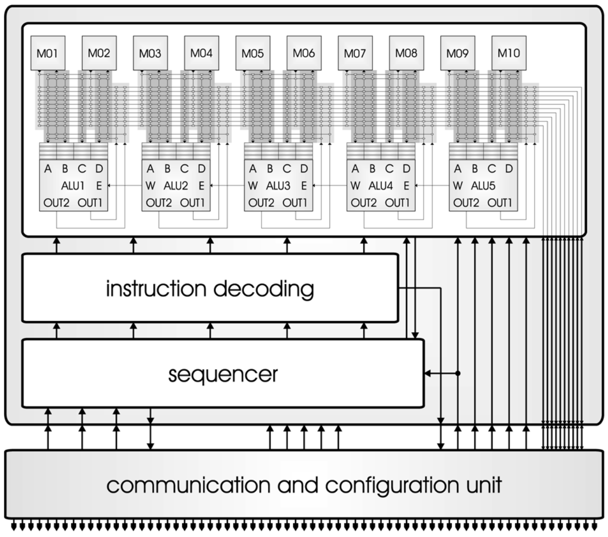


Fig. 2. MONTIUM processing tile.

The proposed tiled SoC template, Chameleon [8], is composed of the previously mentioned processor types (see Fig. 1). The tiles are interconnected by a network-on-chip (NoC). Both SoC and NoC are dynamically reconfigurable, which means that the programs (running on the reconfigurable tiles) as well as the communication channels are defined at run-time. The configuration of the processing tiles and the configuration of the NoC is coordinated by a special *coordination function*. This *coordination function* can be implemented in a GPP processing tile, which is programmed with a run-time operating system, which schedules the DSP tasks at run-time on the heterogeneous reconfigurable SoC. The coarse-grained reconfigurable tiles in the Chameleon SoC template are MONTIUM processing tiles [7], as depicted in Fig. 2.

B. Montium Processing Tile

The MONTIUM is an example of a coarse-grained reconfigurable processor. The MONTIUM [7], [8] targets the 16-bit DSP algorithm domain. A single MONTIUM processing tile is depicted in Fig. 2. At first glance the MONTIUM architecture bears a resemblance to a VLIW processor. However, the control structure of the MONTIUM is very different.

1) Communication and Configuration Unit: The lower part of Fig. 2 shows the communication and configuration unit (CCU) and the upper part shows the reconfigurable tile processor (MONTIUM TP). The CCU implements the interface for off-tile communication. The definition of the off-tile interface depends on the NoC technology that is used in the SoC. The CCU enables the MONTIUM to run in "streaming" as well as in "block" mode. In "streaming" mode the CCU and the MONTIUM TP run in parallel (communication and computation overlap in time). In "block" mode the CCU first reads a block of data, then starts the MONTIUM TP, and finally after completion of the MONTIUM TP the CCU sends the results to the next tile.

The CCU implements the network interface controller between the NoC and the MONTIUM TP. The CCU provides configuration and communications services to the MONTIUM TP, i.e., it follows:

- *configuration* of the Montium TP and parts of the CCU itself;
- *block-based communication* to move data into or from the Montium TP memories and registers (using direct memory access);
- *streaming communication* to stream data into and/or out of the Montium TP while computing.

2) Montium Tile Processor: The TP is the computing part of the MONTIUM that can be configured to implement a particular algorithm. Fig. 2 reveals that the hardware organization of the tile processor is very regular. The five identical arithmetic logic units (ALUs) (ALU1–ALU5) in a tile can exploit spatial concurrency to enhance performance. This parallelism demands a very high memory bandwidth, which is obtained by having ten local memories (M01–M10) in parallel. The small local memories are also motivated by the locality of reference principle. The data path has a width of 16-bits and the ALUs support both signed integer and signed fixed-point arithmetic. The ALU input registers provide an even more local level of storage. Locality of reference is one of the guiding principles applied to obtain energy efficiency in the MONTIUM. A vertical segment that contains one ALU together with its associated input register files, a part of the interconnect, and two local memories is called a processing part (PP). The five PPs together are called the processing part array (PPA). A relatively simple sequencer controls the entire PPA. The sequencer selects configurable PPA instructions that are stored in the decoders of Fig. 2. For (energy) efficiency, it is imperative to minimize the control overhead. This can be accomplished by statically scheduling instructions as much as possible at compile time.

Fig. 3 shows a block diagram of the ALU that is used in the MONTIUM. A single ALU has four 16-bit inputs. Each input has a private input register file that can store up to four operands. The input register file cannot be bypassed, i.e., an operand is always read from an input register. Input registers can be written by various sources via a flexible interconnect. An ALU has two 16-bit outputs, which are connected to the interconnect. The ALU is entirely combinational and consequentially there are no

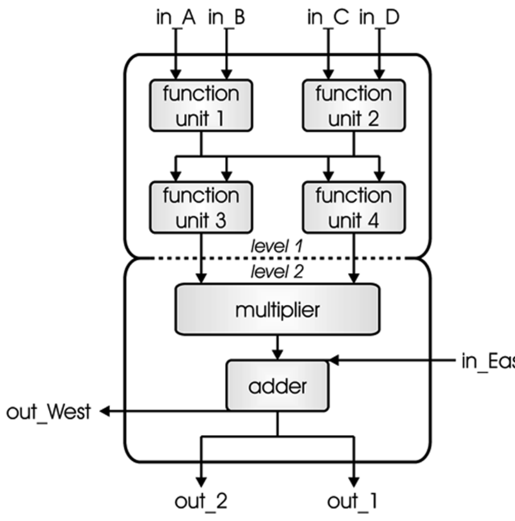


Fig. 3. MONTIUM ALU.

pipeline registers within the ALU. The function units in level 1 of the ALU can be configured to perform general arithmetic and logic operations that are available in languages like C (except multiplication and division). Neighboring ALUs can also communicate directly on level 2. The West-output of an ALU connects to the East-input of the ALU neighboring on the left.

3) Configuration: The MONTIUM TP has no fixed instruction set, but the instructions are configured at configuration-time. During configuration of the MONTIUM, the CCU loads the configuration data (i.e., instructions of the ALUs, memories, and interconnects; sequencer and decoder instructions) in the configuration memory of the MONTIUM. The total configuration memory size of the MONTIUM is about 2.8 kB. However, configuration sizes of DSP algorithms mapped on the MONTIUM are typically in the order of 1 kB. For example, a 64-point fast fourier transform (FFT) has a configuration size of 946 bytes.

By sending a configuration file containing configuration RAM addresses and data values to the CCU, the MONTIUM TP can be configured via the NoC. The configuration memory of the MONTIUM is implemented as a 16-bit wide RAM memory that can be written by the CCU. By updating certain configuration locations of the configuration memory, the MONTIUM is partially reconfigured.

4) Memory: In the considered MONTIUM implementation, each local SRAM is 16-bit wide and has a depth of 1024 positions, which adds up to a storage capacity of 16 kb/local memory. A reconfigurable address generation unit (AGU) accompanies each memory. It is also possible to use the memory as a lookup table for complicated functions that cannot be calculated using an ALU, such as sine or division (with one constant). A memory can be used for both integer and fixed-point lookups.

C. Multiprocessor SoC

The MONTIUM is typically used in a heterogeneous multiprocessor SoC. For instance, one or more MONTIUM cores can be used to offload digital signal processing tasks from a general

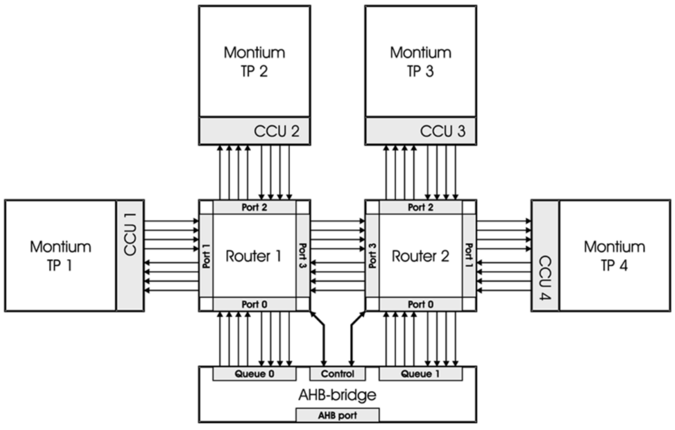


Fig. 4. Reconfigurable subsystem with four MONTIUM tiles.

purpose processor. Fig. 4 shows an example of a simple reconfigurable subsystem that is part of a more complex SoC. A prototype chip with four MONTIUM tiles is currently implemented and samples are expected end 2007. The chip is intended to be used for digital radio (e.g., DAB) and contains only 4 MONTIUM TPs, which is sufficient for the digital radio application. The chip is manufactured in 130-nm CMOS technology and the area of the reconfigurable subsystem is about 15 mm^2 [9].

In Fig. 4, four MONTIUM processing tiles are connected via the CCU to an NoC. Each processing tile is connected to a router of the circuit switched NoC. Both routers are connected to the advanced high performance bus (AHB) bridge, which connects the reconfigurable subsystem to embedded processors, high-performance peripherals, DMA controllers, on-chip memory, and input/output interfaces.

III. APPLICATION DOMAIN

A. SDR

SDR for wireless communication systems are characterized by an analog front-end followed by a programmable, digital baseband processing part. In the analog front-end, the radio signal is received, filtered, and amplified. The filtered, amplified radio signal is converted to digital samples, which are the input of the digital baseband processing part. A programmable, digital baseband processing part enables adaptation features as described in Section I.

A complete ASIC-based radio system has limited use since parameters for each of the functional modules are fixed. A radio system built using SDR technology extends the usability of the system to a range of applications using different link-layer protocols and modulation/demodulation techniques. SDR provides an efficient and relatively inexpensive solution to the design of multimode, multiband, multifunctional wireless devices that can be enhanced using software upgrades.

SDR-enabled devices (i.e., mobile terminals) can be dynamically programmed to reconfigure the characteristics of the device. So, the same hardware can be adapted to perform different functions at different times (time-multiplexed).

Another advantage of the SDR template is the fact that real adaptive systems can be implemented. Traditional algorithms in wireless communications are rather static. The recent emergence of new applications that require sophisticated adaptive, dynamical algorithms based on real-time signal and channel statistics to achieve optimum performance has drawn renewed attention to run-time reconfigurability [10].

However, this flexibility comes at a cost of extra area and configuration overhead. By choosing the right granularity of the reconfigurable hardware, the costs can be controlled. To get a better understanding of these costs, we will proceed as follows:
1) describe key building blocks of the wireless application domain (Sections III-B–III-E);
2) discuss the implementation results and costs of building blocks in reconfigurable hardware (Sections IV-A–IV-E).

TABLE I
DOWNLINK UMTS PROPERTIES

chip rate	3.84 Mega chips/s
scrambling code length	38400 chips
spreading factor (SF)	4 – 512
output symbol rate	7.5 – 960 kilo symbols/s
modulation	QPSK, 16-QAM

B. Wideband CDMA Receiver

The UMTS standard, defined by ETSI, is an example of a third generation (3G) mobile communication system. The communication system has an air interface that is based on CDMA. We only focus on the downlink of the UMTS receiver at the mobile terminal in the FDD mode, the most relevant UMTS properties are shown in Table I [11].

Fig. 5 shows a possible baseband processing function, performed in the WCDMA receiver. Since multipath fading is a common phenomenon in wireless communication systems, the receiver has to combat for the effects of multipath fading. In the UMTS communication system, the signals from the strongest multipaths are received individually. This means that the path searcher of the receiver searches for the strongest received paths and estimates the path-delays. Whenever the delay of an individual path is known, the receiver will perform the descrambling and despreading operations on the delayed signal. The operations of descrambling and despreading are also denoted as a RAKE finger. In the maximal ratio combiner (MRC) the received soft-values of the individual RAKE fingers are individually weighted and combined to provide optimal signal-to-noise ratio. The weighting factors of the individual RAKE fingers are determined by a channel estimator. The RAKE fingers in cooperation with the MRC are called RAKE receiver.

C. OFDM Receiver

High performance radio local area network (HiperLAN/2) is a WLAN access technology and is similar to the IEEE 802.11a WLAN standard. HiperLAN/2 operates in the 5 GHz frequency band and uses orthogonal frequency division multiplexing (OFDM) to transmit the analogue signals. The bit rate of HiperLAN/2 at the physical level depends on the modulation type and is either 12, 24, 48, or 72 Mb/s.

The basic idea of OFDM is to transmit high data rate information by dividing the data into several parallel bit streams, and let each one of these bit streams modulate a separate subcarrier. A

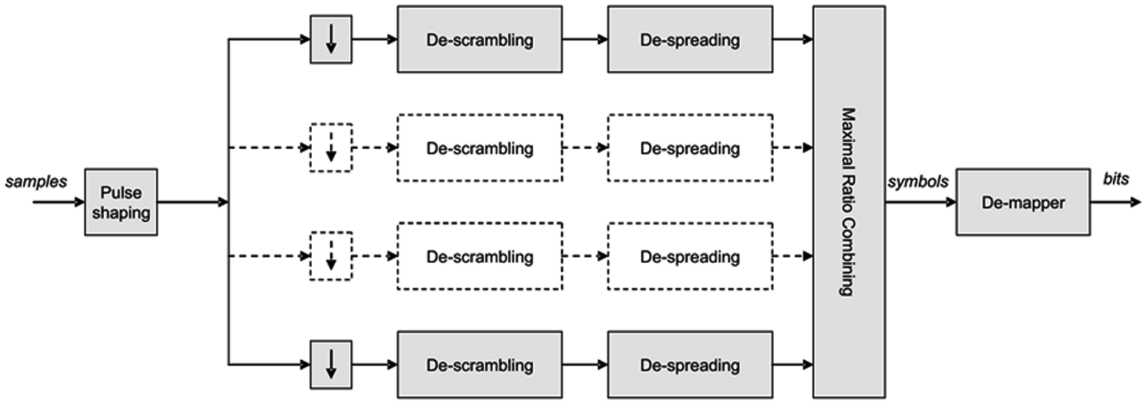


Fig. 5. WCDMA baseband functions in the receiver.

TABLE II
PROPERTIES OF DIFFERENT OFDM-BASED STANDARDS

	Hiper LAN/2	DAB I	DAB II	DAB III	DAB IV	DRM A	DRM B	DRM C	DRM D
Bandwidth [MHz]	20	1.54	1.54	1.54	1.54	0.012	0.012	0.012	0.012
Modulated subcarriers	52	1536	384	192	768	226	206	138	88
FFT size	64	2048	512	256	1024	288	256	176	112
Symbol time [μs]	4	1,246	312	156	623	26,667	26,667	20,000	16,667
Frame time [ms]	2	96	24	24	48	400	400	400	400
Subcarrier spacing [kHz]	312.5	1	4	8	2	0.0417	0.0469	0.0682	0.1071

	Hiper LAN/2	DAB	DRM
Modulation	BPSK / QPSK / 16-QAM / 64-QAM	D-QPSK	BPSK / QPSK / 16-QAM / 64-QAM
Max. user data rate [Mbit/s]	54	$\sim$ 1.8	$\sim$ 0.070

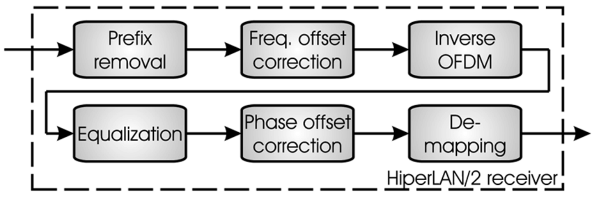


Fig. 6. HiperLAN/2 baseband functions in the receiver.

HiperLAN/2 channel contains 52 subcarriers and has a channel spacing of 20 MHz. 48 subcarriers carry actual data and 4 carry pilots.

The receiver not only performs the inverse operation of the transmitter, it also has to correct for all the distortions that are introduced in the wireless channel. Fig. 6 depicts a model of the HiperLAN/2 receiver. In general, the model can be used for any OFDM-like system. The different standards for OFDM-like systems, e.g., HiperLAN/2, WiMAX, digital audio broadcasting (DAB), digital radio mondiale (DRM), are generally different in the number of subcarriers and the transmission bandwidth. Table II summarizes the OFDM properties for different standards.

The synchronization of the receiver is performed in two steps. First, coarse-synchronization is performed in order to synchronize the receiver with the frame. During coarse-synchronization the received signal is correlated with known preambles, which indicate the start of a frame. Second, the prefix information of an OFDM symbol is used for fine-synchronization. After fine-synchronization, the prefix is removed from the OFDM symbol.

Differences between the oscillator frequencies of the transmitter and the receiver result in frequency offset and cause inter-subcarrier interference. The HiperLAN/2 receiver can compensate for frequency offset by multiplying the data samples of an OFDM symbol with the frequency offset correction coefficient. The frequency offset correction coefficient can be determined by using information from the received preamble sections.

The inverse OFDM part of the receiver converts the received signal into received subcarrier values. The received subcarrier values may still suffer from distortions that need to be corrected before demapping them to a bitstream.

The equalizer corrects the distortions caused by frequency selective fading. The coefficients for the equalizer can be determined by using information from the received preamble sections of the MAC frame. Since the coherence time of a HiperLAN/2 channel is about 20 ms and a burst of a MAC frame has a duration of 2 ms, the coefficients need to be determined only at the start of the MAC frame [12]. Based on the equalized pilot values, the phase distortion of the received signal is corrected.

The received complex-number samples will be translated into an useful received bitstream. The demap function assumes that the most likely symbol that was transmitted, was the symbol that maps to the value closest to the received value.

D. *Viterbi Decoder*

Shannon described in [13] that it is possible to reliably send information over a communication channel with a transmission rate, which is limited by the Shannon capacity (or Shannon

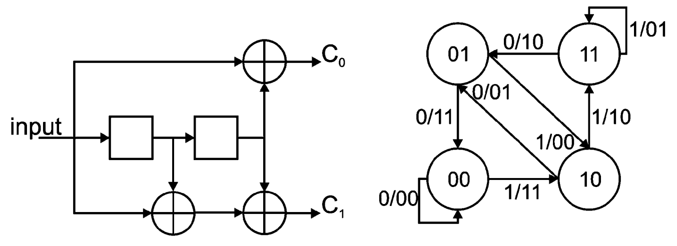


Fig. 7. Convolutional encoder (left) and its state machine (right).

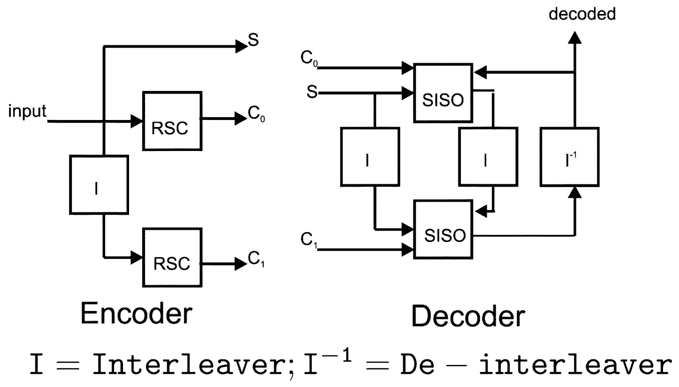


Fig. 8. Turbo encoder (left) and decoder (right).

limit). The Shannon limit is the absolute limit, where no improvement on the bit error rate (BER) can be made without increasing the energy of the bits. Shannon described his theorem, however, he did not give a solution to reliably send information. Many error-correction code schemes have been proposed until now. For example, convolutional codes are widely used in communication systems as error-correction codes. These error-correction codes enable reliable communication of information over a noisy, distorted communication channel by adding redundant information [14].

Convolutional code decoding algorithms are used to estimate the encoded input information, using a method that results in the minimum possible number of errors. Fig. 7 shows the functional diagram of a convolutional encoder as well as its state machine. In [15], Viterbi originally described his maximum-likelihood sequence estimation algorithm, commonly known as the Viterbi algorithm. The job of the decoder is to estimate the path through the trellis that was followed by the encoder. A trellis diagram simply shows the progression of the state of the encoder for different symbol times.

E. Turbo Decoder

Turbo codes, a new family of convolutional codes were proposed in [16] and [17]. These codes are built using concatenation of two recursive systematic convolutional (RSC) codes and their performance is close to the Shannon limit. The recursive codes have a feedback loop in the convolutional encoder, which causes the state of the encoder to be dependent on the state as well as the input. Fig. 8 depicts the basic building blocks of the turbo encoder and its decoder. The Turbo encoder consists of two RSC encoders and an interleaver.

The decoding of Turbo codes is performed in an iterative way. The decoder consists of a deinterleaver and two decoder blocks. These decoders are mostly referred to as soft-input–soft-output

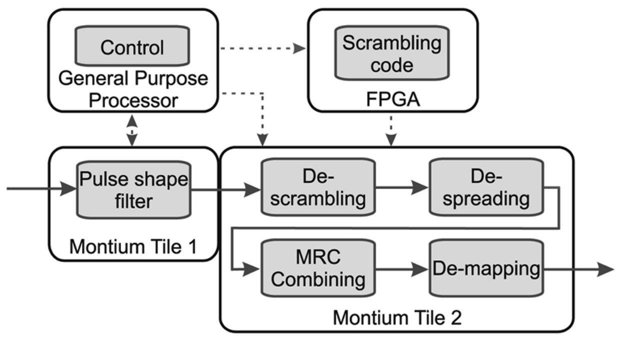


Fig. 9. RAKE receiver in heterogeneous processing tiles.

(SISO) decoders. Each SISO decoder estimates the log-likelihood ratio (LLR), which denotes the logarithm of the probability that a "1" is transmitted divided by the probability that a "0" is transmitted, based on its input signals. These input signals of the SISO decoder are the parity input and systematic input, which is also called the *intrinsic information*, and the feedback information derived by the previous SISO decoder, which is called the *extrinsic information*. Each iteration of Turbo decoding will add extra information to make a better decision on the decoded bit stream.

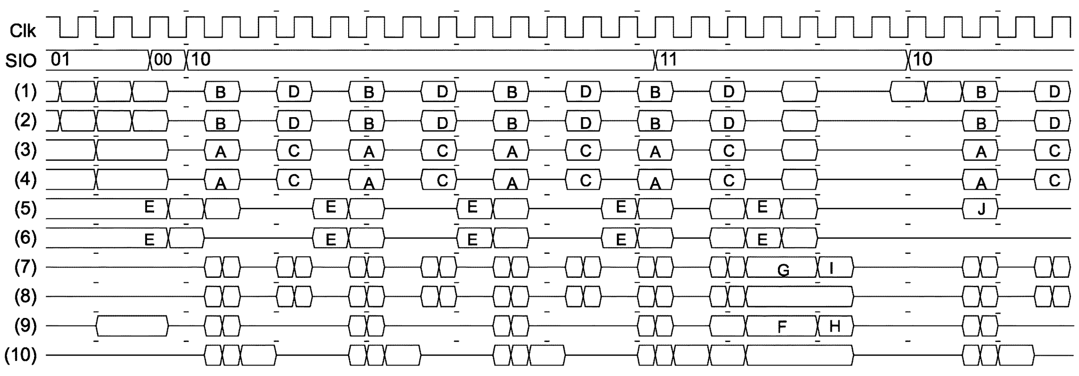
The Turbo and convolutional code schemes have been adopted by many wireless communication standards. In the 3G UMTS system both coding schemes are employed. Turbo coding has been used for data channels and convolutional coding for voice channels [18]. Convolutional code schemes are employed in many OFDM-based standards, like HiperLAN/2 and DAB.

IV. Implementation Results

The previously mentioned DSP building blocks have been implemented in heterogeneous reconfigurable hardware. The target architecture for mapping the DSP algorithms was the coarse-grained Montium architecture. Mapping an application efficiently to, e.g., the Montium TP requires knowledge of both the hardware architecture and the application. Details on the mapping of the applications are described in [1] and [2]. This section emphasizes on implementing multistandard multimode receivers using the Montium architecture in a heterogeneous reconfigurable SoC.

A. Wideband CDMA Receiver

The baseband processing of the WCDMA receiver has been implemented in heterogeneous reconfigurable hardware. Since most baseband processing consists of multiply-accumulate (MAC) operations, the baseband processing of the receiver was implemented in coarse-grained reconfigurable hardware, in our case, the Montium. The scrambling code in the receiver can be generated with simple combinational logic, consisting of shift-registers and XOR gates. These are typical operations that can be performed well in fine-grained reconfigurable hardware, like an FPGA. We assume that the control-oriented functionality is performed in the GPP and provides the right information to the baseband processing part of the WCDMA receiver. Fig. 9 shows the functional blocks of the WCDMA



A - data finger 1
B - data finger 2
C - data finger 3
D - data finger 4
E - scrambling code
F - MRC coefficient finger 1
G - MRC coefficient finger 2
H - MRC coefficient finger 3
I - MRC coefficient finger 4
J - output bitstream

Fig. 10. Signal activity inside the MONTIUM on the global buses (1)–(10).

receiver that are implemented in each processing tile of a heterogeneous reconfigurable SoC.

The WCDMA receiver runs in "streaming" mode. The receiver can process four individual paths of the received signal. Consequently, the receiver requires four complex-number data streams for the four fingers. All fingers require the same scrambling code. The receiver takes the complex-number scrambling code stream as an input. The spreading code is stored in local memory, because the code is relatively small with a maximum length of 512 samples. Furthermore, the spreading code is assigned to a particular user in the UMTS communication system and, therefore, the spreading code will not change frequently. The received symbols of the individual signal paths—fingers—are combined, where each symbol is scaled with a complex-number coefficient. These coefficients are provided by the channel estimator, which is performed on the GPP. The receiver outputs a bit stream with the received data.

Fig. 2 shows that the CCU is directly connected to the global buses inside the MONTIUM. The CCU implements the interface for off-tile communication and so it guarantees that during "streaming" mode the correct signals are available for the MONTIUM tile. Fig. 10 depicts typical signal activity on the global buses inside the MONTIUM during RAKE processing. The different signal streams, which are streamed from outside the MONTIUM, are indicated with characters ("A" till "J") in Fig. 10. The MONTIUM is able to process two RAKE fingers in parallel. The chips of two RAKE fingers can be descrambled and despread in two clock cycles. The typical signal activity reveals the regular organization of the implemented receiver. First, one chip of finger 1 and one of finger 2 are descrambled and despread, in the next two clock cycles one chip of finger 3 and one of finger 4 are descrambled and despread. This typical sequence of signal processing repeats till a complete symbol (consisting of *SF* chips) is descrambled and despread. The next five clock cycles are used for combining the results of the four fingers and demapping the symbols to a bit stream. So, in total $4 \times SF + 5$ clock cycles are needed to process one output symbol, with *SF* denoting the *spreading factor*.

1) Configuration: The configuration size of the flexible RAKE receiver in the MONTIUM is only 858 bytes. One tile can be configured for RAKE receiving in 429 clock cycles. For a configuration clock frequency of 100 MHz this means that a RAKE receiver with four fingers can be configured in 4.29 μs.[2]

In case the spreading factor changes, and so the spreading code, the new spreading code only has to be loaded in the local memory of the MONTIUM and a constant in the MONTIUM configuration has to be changed. Loading a particular spreading code and reconfiguring the constant takes $SF + 1$ clock cycles (partial reconfiguration).

The signal streams for the different fingers are buffered in local memories inside the MONTIUM. When the delay of one of the paths changes, then the buffering strategy in the local memories has to be changed. The buffering strategy of the memories is configured with 24 bytes. These 24 bytes can be reconfigured into 12 clock cycles. Consequently, the RAKE receiver can update its complete path delay profile in 120 ns.

The signal activity in Fig. 10 shows that the signal processing of four RAKE fingers is very regular. The idea behind the regular structure of the four-RAKE receiver is that it can be easily adapted to another configuration with less fingers. Suppose we want to change the receiver to a two-finger equivalent, this means that finger 3 and finger 4 are no longer needed. The CCU will, therefore, stall the streaming of stream "C" and "D" onto global buses 1,…,4 (see Fig. 10). So, the descrambling and despreading phase of finger 3 and finger 4 (data streams "C" and "D") can be bypassed and the number of operations in the combining phase can also be reduced. In total, for reconfiguring the number of fingers from 4 to 2, only 24 bytes have to be reconfigured in the configuration memory of the MONTIUM. The RAKE receiver can be reconfigured in 120 ns, which corresponds to 12 clock cycles.

2) Frequency Scaling: From Fig. 10 it can be seen that the clock frequency of the MONTIUM during RAKE processing of four fingers is about four times the chip rate. Moreover, when the RAKE receiver is reconfigured to two-finger processing, then

[2]In the rest of this paper, we assume that the clock frequency of (re)configuration is 100 MHz.

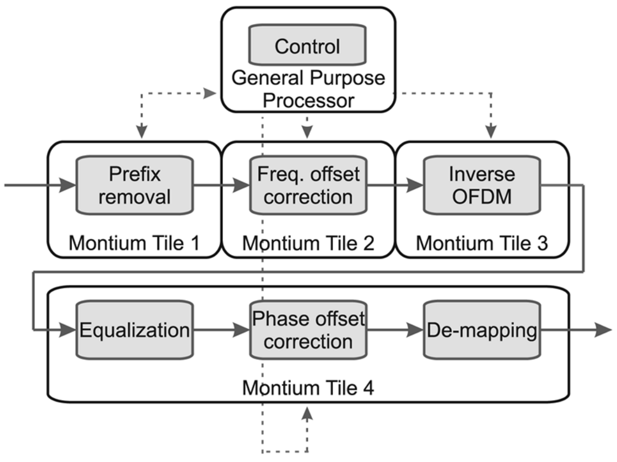


Fig. 11. HiperLAN/2 receiver in heterogeneous processing tiles.

the clock frequency of the MONTIUM can be reduced to about two times the chip rate.

Using power estimation tooling, we estimated the dynamic power consumption of a typical multiply-accumulate operation in the MONTIUM to be about 0.5 mW/MHz, realized in 0.13-μm CMOS technology. Consequently, the power consumption of the implemented RAKE receiver will be 5 mW in two-finger mode and 10 mW in a four-finger mode.

An efficient ASIC implementation of a WCDMA RAKE receiver was described in [19]. The receiver was implemented in 0.13-μm CMOS technology. According to [19], the power dissipation of the ASIC implementation is about 1.5 mW, regardless whether two or four RAKE fingers are implemented. When we compare the power consumption of the ASIC implementation with the MONTIUM implementation, we can conclude that the power consumption of the MONTIUM is about three to seven times larger. As expected, the ASIC implementation is more energy efficient than an implementation in reconfigurable hardware, however, the ASIC implementation is fixed and the functionality of the ASIC cannot be changed, whereas the MONTIUM can be reconfigured for another function.

B. OFDM Receiver

The baseband processing part of the HiperLAN/2 receiver has been implemented in the same reconfigurable hardware. Fig. 11 shows the functional blocks of the receiver that are implemented in each processing tile of a heterogeneous reconfigurable SoC.

Irregular tasks, which are outside the algorithm domain of the MONTIUM, are performed in software (i.e., on the GPP). The irregular processes in the HiperLAN/2 receiver are the estimation of frequency offset and estimation of equalization coefficients. These channel estimations have to be determined only once per MAC frame, i.e., once per 2 ms.

During frequency offset correction, which is performed in the MONTIUM tile, every complex-number sample is multiplied with the frequency offset correction factor. One OFDM symbol, containing 64 complex-number samples, can be corrected in 67 clock cycles.

An FFT on a vector of 64 complex-number time samples can perform the inverse OFDM function. Using the MONTIUM, the 64-FFT can be performed in 204 clock cycles for one OFDM symbol.

TABLE III
PROPERTIES OF THE HIPERLAN/2 IMPLEMENTATION

		Frequency offset correction	Inverse OFDM	Equalizer, Phase offset, De-mapper
Execution time	[cycles]	67	204	110
Communication time	[cycles]	128	116	< 100
Minimum system clock with streaming communication	[MHz]	17	51	28
Configuration size	[bytes]	274	946	576
Configuration time	[cycles]	137	473	288

The equalizer, phase offset correction and demapping functionality are implemented in one MONTIUM tile in a pipelined fashion. The coefficients for equalization are determined once every 2 ms in software by the GPP. During equalization, the received subcarriers are multiplied with the equalization coefficients. After equalization the pilot values are used to determine the phase offset correction factor. The phase offset correction factor is determined in the MONTIUM, since the phase offset can vary for every OFDM symbol and the correction factor has to be determined on an OFDM symbol basis (i.e., once every 4 μs). Hence, determining the phase offset correction factor in software (i.e., on the GPP) would create a large communication overhead between the GPP and the MONTIUM tile. Phase offset correction invokes also a complex multiplication, like equalization. As a consequence, the equalizer and phase offset corrector use the same functionality of the MONTIUM. In a pipelined manner, the corrected complex-number samples are translated into a bitstream. Hard-decision demapping is implemented with the LUT functionality. A parametrizable demapper has been implemented, which can be used for QPSK, 16-QAM, and 64-QAM modulated signals by only changing the LUT table in the local memory of the MONTIUM.

1) Configuration: The total configuration sizes of the MONTIUM are small for the different functions (see Table III). The FFT implementation in the MONTIUM requires the largest configuration size, which is less than 1 kB of data. The configuration data of the FFT algorithm can be written into the configuration memory of the MONTIUM in 473 clock cycles, since 2 bytes are written in one clock cycle. This MONTIUM tile can be (re)configured in 4.73 μs. Notice that the maximum radio turn-around time of the HiperLAN/2 communication system is 6 μs [20] and, therefore, the implemented HiperLAN/2 receiver can be considered as a real-time dynamically reconfigurable receiver.

2) Frequency Scaling: All operations in the physical layer of the HiperLAN/2 system are performed on OFDM symbols. So, one should assure that each 4 μs a new OFDM symbol can be processed. When a streaming on-chip network between the processors is assumed, the communication time is not a bottleneck and one only has to guarantee that, for example, the data processing for frequency offset correction is performed within 4 μs. Hence, the minimum clock frequency of the assigned MONTIUM processing tile is 17 MHz, when a streaming on-chip network between the tiles is assumed. Table III summarizes requirements on the clock frequency for the MONTIUM tile processors.

C. Viterbi Decoder

A fully flexible Viterbi decoder has been implemented in the MONTIUM, based on a hybrid register exchange/traceback approach [21]. The *rate* R as well as the *constraint length* k and the *decision depth* d of the decoder can be adapted within certain boundaries. These boundaries depend on the size of the local memories inside the MONTIUM. Implementation properties of the Viterbi decoder on the MONTIUM are discussed based on the settings of the DAB communication system, $R = 1/4$ and $k = 7$ with a *decision depth* $d = 50$.

1) Configuration: The total configuration size of the MONTIUM Viterbi implementation is 1356 bytes. The configuration of the Viterbi decoder can be loaded in the MONTIUM's configuration memory in 6.78 μs.

Once the MONTIUM is configured as Viterbi decoder, only partial reconfiguration has to be performed in order to adjust the *constraint length, decision depth*, or *rate*. Especially the *decision depth* depends heavily on the conditions of the wireless channel. Thus, adjusting the *decision depth* can be typically performed at run-time via dynamic reconfiguration.

2) Throughput: In the implemented DAB Viterbi decoder, always 10 bits are generated during the survivor decision phase. On average 47 clock cycles are required to decode one output bit. The output rate of the Viterbi decoder in the MONTIUM is 2.1 Mb/s running at 100 MHz. This is sufficient for DAB, which requires an output rate of 1.8 Mb/s.

D. Turbo Decoder

The SISO decoders in the Turbo decoder can be implemented using several algorithms [22]. We implemented the max-log-map (MLM) algorithm in the MONTIUM. This algorithm has a regular optimized structure and achieves near-optimal BER.

The MLM algorithm consists of three processing phases: *forward recursion, backward recursion*, and *LLR calculation*. The information from the forward and backward recursion are used to estimate the LLR information. Because the LLR calculation can be done while the backward recursion is performed, the backward estimations do not need to be stored in memory. However, all the forward estimations have to be stored in memory. Hence, in order to be compliant with the 3G UMTS standard, at most 5114×8 forward estimates have to be stored for full block length. The required memory to store the forward estimates can be reduced by applying the *sliding window* approach [23]. This approach divides the full block into smaller blocks, *windows*, on which the algorithm is applied. The number of forward estimates that needs to be stored is now equal to the window length.

1) Configuration: The total configuration size of the MONTIUM MLM implementation is 1262 bytes. This configuration can be loaded in the MONTIUM's configuration memory in 6.36 μs.

2) Throughput: The Turbo codes used in the UMTS communication system have constraint length $k = 4$, which means that eight states exist in the trellis of the Turbo code. So, for each time instant of the trellis, eight forward *state metric* estimations have to be performed during the *forward recursion*, and eight backward *state metric* estimations have to be made for the *backward recursion*. The parallelism of ALUs and memories in the MONTIUM provides resources to calculate the forward and backward recursion in four clock cycles for one time instant of the trellis.

The intermediate forward state metrics are stored in the local memories of the MONTIUM. Immediately after the calculation of the backward state metrics, the LLR is calculated. The LLR calculation in the MONTIUM is performed in four clock cycles per time instant of the trellis. Consequently, eight clock cycles are required to apply the MLM algorithm for one time instant of the trellis.

The maximum channel data rate of the UMTS communication system is 1.92 Mb/s, which means the maximum Turbo frame of 5114 bits has to be processed in 2.66 ms. In order to perform Turbo decoding with ten iterations, the MONTIUM should run at a speed of 110 MHz in this case. The inner and outer decoder are applied during one Turbo decoding iteration without considering the interleaving process.

E. Discussion

In [24], a channel decoder chip was proposed that is compliant with the 3G wireless standard. However, this chip is a dedicated solution for the 3G UMTS system, which cannot be used in other wireless communication standards. We implemented both the Turbo and the Viterbi decoder in the coarse-grained reconfigurable MONTIUM architecture, which can also be used to implement the baseband processing. The flexible coarse-grained reconfigurable MONTIUM enables the implementation of flexible baseband processing and flexible channel decoding in multimode communication systems.

The unified channel decoder chip in [24] has been implemented in older CMOS technology, therefore, we cannot fairly compare the implementation with the MONTIUM implementation. In [25], another reconfigurable architecture for Viterbi and Turbo decoding was reported. That architecture, *Viturbo*, can be configured to decode convolutionally coded data and Turbo coded data. The *Viturbo* decoder is only aimed for channel decoding, whereas the MONTIUM architecture is more flexible and suitable for baseband processing as well. The area of the MONTIUM is slightly larger than the *Viturbo* decoder ($\sim$250 kGates versus $\sim$200 kGates).

To enable a multistandard multimode SDR receiver that is capable of both baseband processing and channel decoding, the heterogeneous reconfigurable SoC needs to be equiped with a *coordination function*, as introduced in Section II. This *coordination function* can be implemented in a GPP processing tile, which is controlled by a run-time operating system. The operating system schedules the DSP tasks at run-time on processing tiles in the heterogeneous reconfigurable SoC. The configurations for the MONTIUM TPs of the different SDR applications, as described in Section IV, can be compiled at design time. The *coordination function* selects the right configuration when an application is started and handles the reconfiguration of the processing tiles in the SoC.

V. CONCLUSION

Because, in our opinion, heterogeneous reconfigurable systems will become the future of mobile hardware, we proposed a heterogeneous SoC containing reconfigurable processing elements of different grain sizes. The processing elements in the SoC are dynamically interconnected by an NoC.

The MONTIUM architecture showed to have sufficient flexibility and processing capabilities for implementing key building blocks of wireless communication systems. The feasibility of using heterogeneous hardware is demonstrated by implementing a RAKE receiver and a HiperLAN/2 receiver on the same SoC.

The flexible RAKE receiver implements the baseband processing for receiving WCDMA signals. It is flexible because the number of RAKE fingers can be adjusted in real-time. In less than 5 μs a MONTIUM can be configured for RAKE processing. One MONTIUM can be partially reconfigured to change the number of fingers in the RAKE receiver. Adjusting the number of fingers from four to two only takes 120 ns; short enough to classify as dynamic reconfiguration.

The same reconfigurable hardware can be configured as a HiperLAN/2 receiver. The HiperLAN/2 receiver can be implemented in four MONTIUM tiles. The performance requirements of the receiver can be met at fairly low clock frequencies, with low configuration overhead. The MONTIUM tiles can be configured for HiperLAN/2 baseband processing in less than 5 μs.

Moreover, we showed that the coarse-grained reconfigurable MONTIUM is suitable for implementing channel decoding algorithms. We presented the implementation results of the Viterbi algorithm as well as the MLM algorithm, used in Turbo decoding, in the same MONTIUM processing tile.

The configuration overhead of the decoders is relatively small, as the configuration files of both decoders are small. Hence, changing the functionality of the channel decoder from Turbo to Viterbi, or vice versa, can be done dynamically, because of the short reconfiguration times. Depending on the desired communication standard, one can configure the hardware in the mobile terminal to implement the right channel decoder. The reconfiguration time of the Viterbi or Turbo decoder implementation is less than 7 μs.

REFERENCES

[1] G. J. M. Smit and G. K. Rauwerda, "Reconfigurable architectures for adaptable mobile systems," in *Proc. Int. Conf. Eng. Reconfigurable Syst. Algorithms (ERSA)*, 2005, pp. 17–25.

[2] G. K. Rauwerda *et al.*, "Reconfigurable turbo/viterbi channel decoder in the coarse-grained montium architecture," in *Proc. Int. Conf. Eng. Reconfigurable Syst. Algorithms (ERSA)*, 2006, pp. 110–116.

[3] L. T. Smit, G. J. M. Smit, and J. L. Hurink, "Energy-efficient wireless communication for mobile multimedia terminals," in *Proc. Int. Conf. Adv. Mobile Multimedia*, 2003, pp. 115–124.

[4] L. T. Smit, "Energy-efficient wireless communication," Ph.D. dissertation, Dept. Comput. Sci., Univ. Twente, Enschede, The Netherlands, 2004.

[5] G. Rauwerda *et al.*, "Adaptive wireless networking," in *Proc. 4th PROGRESS Symp. Embedded Syst.*, 2003, pp. 205–211.

[6] R. Schiphorst, "Software-defined radio for wireless local-area networks," Ph.D. dissertation, Dept. Elect. Eng., Univ. Twente, Enschede, The Netherlands, 2004.

[7] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," *J. Supercomput.*, vol. 26, no. 3, pp. 283–308, Nov. 2003.

[8] P. M. Heysters, "Coarse-grained reconfigurable processors—Flexibility meets efficiency," Ph.D. dissertation, Dept. Comput. Sci., Univ. Twente, Enschede, The Netherlands, 2004.

[9] "Smart chips for smart surroundings," [Online]. Available: http://www.smart-chips.net

[10] J. Potman, F. Hoeksema, and K. Slump, "Tradeoffs between spreading factor, symbol constellation size and rake fingers in UMTS," in *Proc. ProRISC*, 2003, pp. 543–548.

[11] H. Holma and A. Toskala*, WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*. New York: Wiley, 2001.

[12] A. Berno, "Time and frequency synchronization algorithms for HIPERLAN/2," M.S. thesis, Dept. of Electron. Comput. Sci., Univ. Padova, Padova, Italy, 2001.

[13] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423–623–656, 1948.

[14] S. Lin and D. J. Costello*, Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[15] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[16] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE ICC*, 1993, pp. 1064–1070.

[17] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[18] 3rd Generation Partnership Project, "Technical specification group radio access network; multiplexing and channel coding (FDD)," , 3GPP TS 25.212 v4.3.0 (2001-12), Jan. 2002.

[19] M. Nilsson, "Efficient ASIC implementation of a WCDMA rake receiver," M.S. thesis, Dept. Comput. Sci. Elect. Eng., Div. Comput. Eng., Luleå Univ. Technol., Stockholm, Sweden, 2002.

[20] European Telecommunications Standards Institute (ETSI), "Broadband radio access networks (BRAN); HiperLAN Type 2; Data link control (DLC) layer part 1: Basic data transport functions," ETSI TS 101 761-1 v1.1.1 (2000-04), 2000.

[21] C. M. Rader, "Memory management in a viterbi decoder," *IEEE Trans. Commun.*, vol. 29, no. 9, pp. 1399–1401, Sep. 1981.

[22] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE ICC*, 1995, pp. 1009–1013.

[23] J. Dielissen *et al.*, "Power-efficient layered turbo decoder processor," in *Proc. Conf. Des., Autom. Test Europe (DATE)*, 2001, pp. 246–251.

[24] M. A. Bickerstaff *et al.*, "A unified turbo/viterbi channel decoder for 3GPP mobile wireless in 0.18-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, Nov. 2002.

[25] J. R. Cavallaro and M. Vaya, "VITURBO: A reconfigurable architecture for viterbi and turbo decoding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2003, pp. 497–500.

Gerard Rauwerda received the M.Sc. degree in wireless communications from the University of Twente, Enschede, The Netherlands, in 2002, where he is currently pursuing the Ph.D. degree with an interest in mapping software defined radio algorithms on reconfigurable hardware. His thesis is entitled "Multi-standard adaptive wireless communication receivers."

He is currently Executive Director with Recore Systems, Enschede, The Netherlands, of which he is also a cofounder. He was a Visiting Researcher with Atmel Germany, Ulm, Germany, where he investigated opportunities for reconfigurable computing in digital radio broadcasting receivers.

Paul Heysters received the M.Sc. degree in computer science and the Ph.D. degree from the University of Twente, Enschede, The Netherlands, in 1998 and 2004, respectively, with the Ph.D. thesis entitled "Coarse-grained reconfigurable processors—Flexibility meets efficiency."

He has been CEO of Recore Systems, Enschede, The Netherlands, since September 2005. He has more than seven years experience working in the field of reconfigurable computing. In his career, he worked for high-technology companies in both Europe and the USA, including Ericsson, Philips, and Chameleon Systems. Prior to cofounding Recore Systems, he was leading research on coarse-grained reconfigurable computing for the CHAMELEON Project with the University of Twente, Enschede, The Netherlands, and worked collaboratively with industry organizations.

Gerard Smit received the M.Sc. degree in electrical engineering from the University of Twente, Enschede, The Netherlands. He finished his Ph.D. thesis entitled "the design of Central Switch communication systems for Multimedia Applications" in 1994.

He has been a Full Professor with the faculty of EEMCS, University of Twente since 2007, where he is responsible for a number of research projects sponsored by the EC, industry, and Dutch government in the field of multimedia and reconfigurable systems. After receiving the M.Sc. degree, he worked for four years at the Research Laboratory of Océ, Venlo, The Netherlands. In 1994, he was a Visiting Researcher with the Computer Laboratory, Cambridge University, Cambridge, MA, and, in 1998, he was a Visiting Researcher with Lucent Technologies Bell Labs Innovations, Murray Hill, NJ. Since 1999, he has been with the CHAMELEON Project, which investigates new hardware and software architectures for battery-powered hand-held computers. Currently, his research interests include low-power communication, wireless multimedia communication, and reconfigurable architectures for energy reduction.