Reconfigurable Architectures for Adaptable Mobile Systems

(Invited Paper)

Gerard J.M. Smit, Gerard K. Rauwerda
University of Twente, department EEMCS
P.O. Box 217, 7500 AE Enschede, the Netherlands
g.j.m.smit@utwente.nl

***Abstract*— Mobile wireless terminals tend to become multi-mode 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 WCDMA and an OFDM receiver in the same coarse-grained reconfigurable MONTIUM processor is discussed.**

***Index Terms*— Heterogeneous reconfigurable hardware, Software defined radio, SoC, MONTIUM, Wideband CDMA, OFDM**

I. INTRODUCTION

Future wireless communications systems tend to become multi-mode, multi-functional devices. Adaptivity becomes ever more important. 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 QoS). When the system can adapt – at run-time – to the environment, significant savings in computational costs can be obtained [1], [2]. Furthermore, the hardware architectures have to be extremely efficient as these are used in battery-operated terminals and cost effective as they are used in consumer products.

Heterogeneous reconfigurable hardware offers 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 hardware enables energy-efficient implementations of the wireless standards. Much work has been done on Software Defined Radio (SDR) in the SDR forum context [3]. However this forum mainly focuses on general-purpose processors and they do not concentrate on reconfigurable platforms and energy-efficiency.

As already stated in the introduction 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 UMTS and a 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 in the wireless terminal because the wireless communication standard defines the DSP functions that have to be performed to implement the standard.

Although a wireless communication standard usually defines the DSP functions which have 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 a DSP function. ‘Adapt the algorithms’ means that the communication system selects an algorithm from a set of algorithms that implement the same DSP function. Therefore this second level of adaptivity is referred to as *algorithm-selection level* adaptivity.

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 [4].

An adaptive wireless terminal only makes sense when it better suits the needs of a user at an acceptable complexity and efficiency compared to a traditional non-adaptive terminal. The standards level of adaptivity allows the terminal to adapt the communication standard that is used to the Quality of Service (QoS) requirements and the communication environment. Here the

term *communication environment* is used to indicate the available wireless communication standards and wireless channel conditions at a certain location. The algorithm-selection level of adaptivity allows the terminal to select the algorithms that satisfy the QoS requirements in the given communication environment in the most efficient manner. The algorithm-parameter level of adaptivity allows the terminal to do the same with the parameters of a specific algorithm. So a reconfigurable and adaptive terminal is able to track the QoS requirements and communication environment on a finer grained scale than a traditional non-adaptive terminal.

The complexity of an adaptive terminal is determined by the complexity of the standards that it has to support, the complexity of the algorithms that are used to implement the DSP functions of the standard and the complexity of the measurement and control overhead that is required to make the terminal adaptive. The complexity of the algorithms in the set of algorithms that is used for algorithm-selection level adaptivity should therefore be centered around the complexity of the algorithm that is used in a traditional non-adaptive terminal. To minimize the overhead of measuring QoS and channel conditions, measurements that are already required by the wireless communication standard should be (re)used wherever possible. The complexity of any additional measurement algorithms should be kept low. The same is true for control algorithms.

In this paper we discuss the implementation of wireless communication systems in heterogeneous reconfigurable hardware. The implementation of a flexible RAKE receiver, used for UMTS communications, and the implementation of an OFDM receiver, used in HiperLAN/2, is studied to show the feasibility of implementing multi-mode communication systems using reconfigurable hardware.

II. Related work

So far most algorithmic level research on reconfigurability in UMTS, as for example in the *MuMoR* [5] project, has focussed on multi-mode reconfigurability to enable Software Defined Radios (SDRs) supporting multiple communication standards. The *EASY* project [6] aims at developing a power/cost efficient System-on-Chip (SoC) implementation of the HiperLAN/2 standard. In the *Adaptive Wireless Networking (AWGN)* project [4], however, reconfigurability will be used to allow the communication system to adapt to changing environmental conditions.

Both academy and industry show interest in coarse-grained reconfigurable architectures. The Pleiades project at UC Berkeley focuses on an architectural template for ultra low-power high-performance multimedia computing [7]. In the Pleiades architecture template a general-purpose microprocessor is surrounded by a heterogeneous array of autonomous, special-purpose satellite processors. The Pleiades SoC design methodology assumes a (very) specific algorithm domain. The extreme processor platform (XPP) of PACT is based on clusters of coarse-grained processing array elements (PAEs) [8]. Actual PAEs are tailored to the algorithm domain of a particular XPP processor. Silicon Hive [9] offers coarse-grained reconfigurable block accelerators (e.g. Avispa and Moustique) and stream accelerators (e.g. Bresca) for high performance and low power applications. The architecture consists of VLIW-like datapath elements.

III. Reconfigurable heterogeneous architecture

Heterogeneous reconfigurable systems might become the future of mobile hardware. The basic idea behind the use of heterogeneous reconfigurable hardware is that one can match the granularity of the DSP algorithms with the granularity of the hardware. For instance some algorithms perform operations best on bit-level while other perform best on word-level. Four types of processing elements can be distinguished: *general-purpose* processor, *fine-grained reconfigurable* hardware, *coarse-grained reconfigurable* hardware and *dedicated* hardware.

A. The Chameleon System-on-Chip

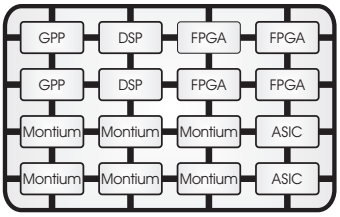

Fig. 1. The Chameleon SoC.

We propose a System-on-Chip (SoC), which consists of the above mentioned processing elements. Figure 1 shows the SoC, which we call the Chameleon SoC. The SoC contains a tiled architecture, where tiles can be processing elements of different granularities. The tiles in the SoC are interconnected by a Network-on-Chip (NoC). Both the SoC and NoC are dynamically reconfigurable, which means that the programs (running

on the reconfigurable processing elements) as well as the communication links between the processing elements are defined at run-time [10].

The Chameleon SoC contains several general-purpose tiles, i.e. GPP and DSP. The FPGA tiles in the SoC represent fine-grained processing elements. The coarse-grained processing elements in the Chameleon SoC are implemented by MONTIUM tile processors. Some tiles in the SoC are implemented as ASIC, which have dedicated functionality.

B. The Montium Tile Processor

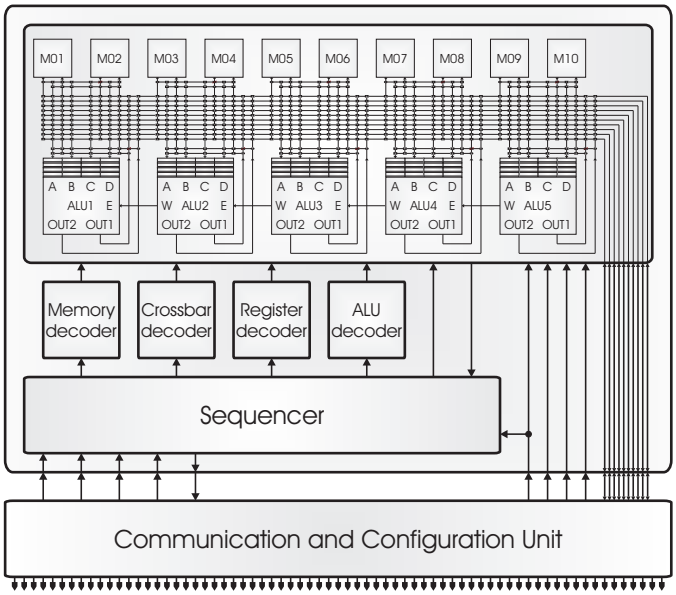

Fig. 2. The MONTIUM Tile Processor.

The MONTIUM is an example of a coarse-grained reconfigurable processor. The MONTIUM [11], [10] targets the 16-bit digital signal processing (DSP) algorithm domain. A single MONTIUM processing tile is depicted in Figure 2. At first glance the MONTIUM architecture bears a resemblance to a VLIW processor. However, the control structure of the MONTIUM is very different. 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.

The lower part of Figure 2 shows the Communication and Configuration Unit (CCU) and the upper part shows the reconfigurable Tile Processor (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 run in parallel (communication and computation overlap in time). In 'block' mode the CCU first reads a block of data, then starts the MONTIUM, and finally after completion of the MONTIUM the CCU sends the results to the next tile.

The TP is the computing part that can be configured to implement a particular algorithm. Figure 2 reveals that the hardware organization of the tile processor is very regular. The five identical ALUs (ALU1 $\cdots$ ALU5) in a tile can exploit spatial concurrency to enhance performance. This parallelism demands a very high memory bandwidth, which is obtained by having 10 local memories (M01 $\cdots$ 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 Processing Parts 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 Figure 2.

Each local SRAM is 16-bit wide and has a depth of 512 positions, which adds up to a storage capacity of 8 Kbit per 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.

Figure 3 shows 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 pipeline registers within the ALU. Neighbouring ALUs can also communicate directly on level 2. The West-output of an ALU connects to the East-input of the ALU neighbouring on the left. The East-West connection does

not introduce a delay or pipeline, as it is not registered.

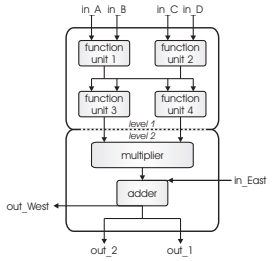

Fig. 3. The MONTIUM ALU.

IV. APPLICATION DOMAIN

A. Software Defined Radio

Software Defined Radio (SDR) denotes wireless communication systems that 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 reprogramming of the functional modules that have to be performed, like modulation/demodulation techniques.

A complete hardware based radio system (e.g. an ASIC solution) has limited utility since parameters for each of the functional modules are fixed. A radio system built using SDR technology extends the utility of the system to a wide range of applications using different link-layer protocols and modulation/demodulation techniques. SDR provides an efficient and relatively inexpensive solution to the design of multi-mode, multi-band, multi-functional wireless devices that can be enhanced using software upgrades only.

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.

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 signal and channel statistics to achieve optimum performance has drawn renewed attention to run-time reconfigurability [12].

B. Wideband CDMA receiver

The Universal Mobile Telecommunications System (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 Code Division Multiple Access (CDMA). We will investigate the possibilities of implementing the DSP functionality of a UMTS receiver in reconfigurable hardware. 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 [13].

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

Figure 4 shows the baseband processing, performed in the W-CDMA receiver. Since multi-path fading is a common phenomenon in wireless communication systems, the receiver has to combat for the effects of multi-path fading. In the UMTS communication system the signals from the strongest multi-paths 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 de-scrambling and de-spreading operations on the delayed signal. The operations of de-scrambling and de-spreading are also denoted as a RAKE finger. In the Maximal Ratio Combiner (MRC) the received soft-values of the individual RAKE fingers are combined and individually weighted to provide optimal Signal-to-Noise Ratio (SNR). The weighting factors of the individual RAKE fingers are determined by a channel estimator. The RAKE fingers in co-operation with the MRC are called RAKE receiver.

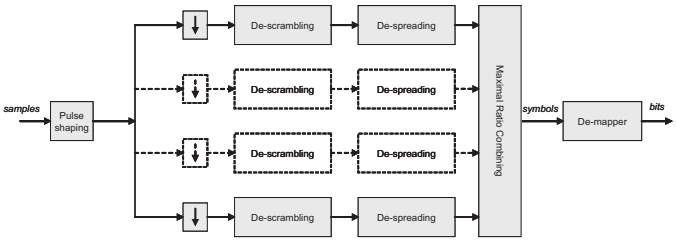

Fig. 4. W-CDMA baseband functions in the receiver.

C. OFDM receiver

HiperLAN/2 is a wireless local area network (WLAN) access technology and is similar to the IEEE 802.11a WLAN standard. HiperLAN/2 operates in the 5 GHz frequency band and makes use of 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 Mbit/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 HiperLAN/2 channel contains 52 subcarriers and has a channel spacing of 20 MHz. 48 subcarriers carry actual data and 4 carry pilots.

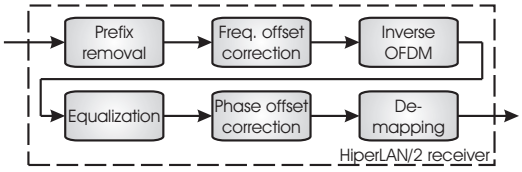

Fig. 5. HiperLAN/2 baseband functions in the receiver.

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. Figure 5 depicts a model of the HiperLAN/2 receiver. In general, the model can be used for any OFDM-like system. The diffent standards for OFDM-like systems, e.g. HiperLAN/2, DAB, DRM, are generally different in the number of carriers and the transmission bandwidth. Table II summarizes the OFDM properties for different standards.

TABLE II
PROPERTIES OF THE DIFFERENT OFDM STANDARDS.

	Hiper LAN/2	DAB				DRM	
		I	II	III	IV	A	B
Bandwidth [MHz]	20	1.54	1.54	1.54	1.54	0.012	0.012
# carriers	52	1536	384	192	768	203	181
Symbol time [μs]	4	1,246	312	156	623	26,667	26,667
Frame time [ms]	2	96	24	24	48	400	400

The synchronization of the receiver is performed in two steps. Firstly, 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. Secondly, 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 of the MAC frame.

The inverse OFDM part of the receiver converts the received signal into received subcarrier values. The received sub-carrier values may still suffer from distortions that need to be corrected before de-mapping 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 [14].

Based on the equalized pilot values, the phase distortion of the received signal is corrected. The phase correction coefficient is determined using pilots.

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

V. IMPLEMENTATION

Good development tools are essential for quick implementation of applications in reconfigurable architectures. The intention of the MONTIUM development tools is to start with a high-level description of an application (in C/C++ or Matlab) and translate this description to a MONTIUM configuration. The development tools comprise, among other things, a stand-alone MONTIUM simulator, a MONTIUM assembler and a MONTIUM configuration editor [10]. Until now, majority of the applications are implemented in the MONTIUM using the assembler and the configuration editor.

For functional simulations we use a co-simulation environment with Matlab and ModelSim, as depicted in Figure 6. Using this environment, one can simulate the (baseband) functionality of a complete communication

system in software (i.e. using Matlab) and partly in hardware (i.e. using ModelSim).

In Matlab code all functions of the system under simulation are defined. The software part of the simulator is furthermore responsible for the control mechanisms in the system under simulation. VHDL code describes the functionality implemented in hardware. ModelSim uses this code to perform simulations of the functionality in hardware. The software and hardware counterparts cooperate in one simulation with each other via file transfers.

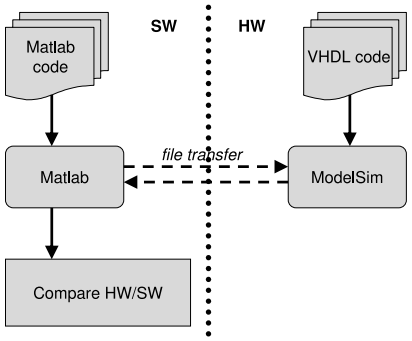

Fig. 6. The co-simulation environment.

A. Wideband CDMA receiver

The W-CDMA 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 will be generated with simple combinational logic, consisting of shift-registers and XOR gates. These are typical operations that can be performed 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 W-CDMA receiver.

In our design the pulse shape filter, which can be implemented as a FIR filter, is implemented in one MONTIUM tile. The output streams of the pulse shape filter are the input for the RAKE receiver, which is implemented in a second MONTIUM tile. Figure 7 shows the functional blocks in the W-CDMA receiver that are implemented in each processing tile.

The W-CDMA 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

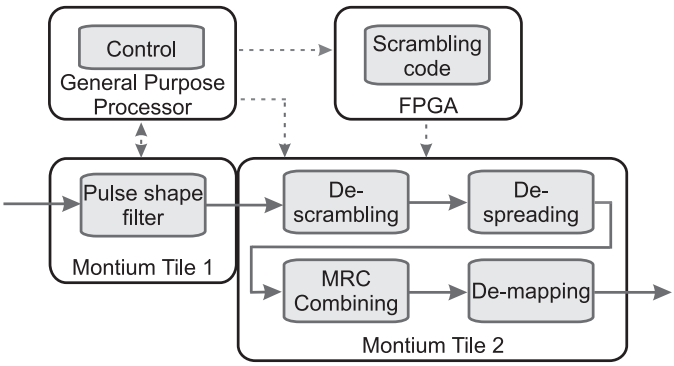

Fig. 7. The RAKE receiver in heterogeneous processing tiles.

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, while 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.

Figure 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. Figure 8 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 Figure 8. The MONTIUM is able to process two RAKE fingers in parallel. The chips of two RAKE fingers can be de-scrambled and de-spread 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 de-scrambled and de-spread, in the next 2 clock cycles one chip of finger 3 and one of finger 4 are de-scrambled and de-spread. This typical sequence of signal processing repeats till a complete symbol (consisting of SF chips) is de-scrambled and de-spread. The next 5 clock cycles are used for combining the results of the 4 fingers and de-mapping the symbols to a bit stream. So, in total $4 \times SF + 5$ clock cycles are needed to process one output symbol.

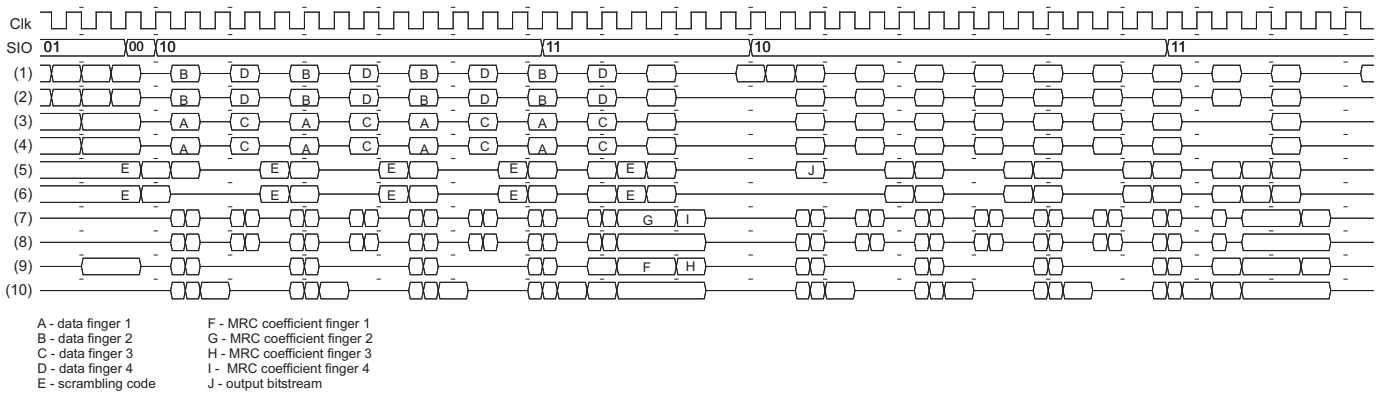

Fig. 8. Signal activity inside the MONTIUM on the global buses (1) $\cdots$ (10).

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 4 fingers can be configured in 4.29 μs.

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 costs $SF + 1$ clock cycles.

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 in 12 clock cycles. Consequently, the RAKE receiver can update its complete path delay profile in 120 ns, assuming that the configuration clock frequency is 100 MHz.

The signal activity in Figure 8 shows that the signal processing of 4 RAKE fingers is very regular. The idea behind the regular structure of the 4-RAKE receiver is that it can be easily adapted to another configuration with for instance less fingers. Suppose we want to change the receiver to a 2 finger one, 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 till 4 (Figure 8). So, the de-scrambling and de-spreading 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. Assuming that the clock frequency of the processor tile during reconfiguration is 100 MHz, the RAKE receiver can be reconfigured in 120 ns, which corresponds to 12 clock cycles.

2) Dynamic Voltage and Frequency Scaling: Voltage and frequency scaling is an important measure to control the power consumption of embedded systems. In CMOS design, the power consumption depends quadratically on the supply voltage and linearly on frequency. The main idea of dynamic voltage and frequency scaling is that the supply voltage should be kept as low as possible. Besides, the maximum operating frequency is tightly coupled to the supply voltage level. This means that by scaling the clock frequency of hardware, the supply voltage can be scaled as well, resulting in a quadratic decrease of the power consumption.

From Figure 8 can be seen that the clock frequency of the MONTIUM during RAKE processing of 4 fingers is about 4 times the chip rate. Moreover, when the RAKE receiver is reconfigured to 2 finger processing, then the clock frequency of the MONTIUM can be reduced to about 2 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 2-finger mode and 10 mW in 4-finger mode.

An efficient ASIC implementation of a W-CDMA RAKE receiver was described in [15]. The receiver was implemented in 0.13 CMOS technology. According to [15], the power dissipation of the ASIC implementation is about 1.5 mW, regardless whether 2 or 4 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 3 to 7 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 can not be changed.

B. HiperLAN/2 receiver

The HiperLAN/2 receiver has been implemented in the same reconfigurable hardware. Figure 9 shows the functional blocks in the receiver that are implemented in each processing tile. The synchronization part (prefix-removal) has still not been implemented. Nevertheless the function, which contains many correlation operations, can easily be implemented in the MONTIUM.

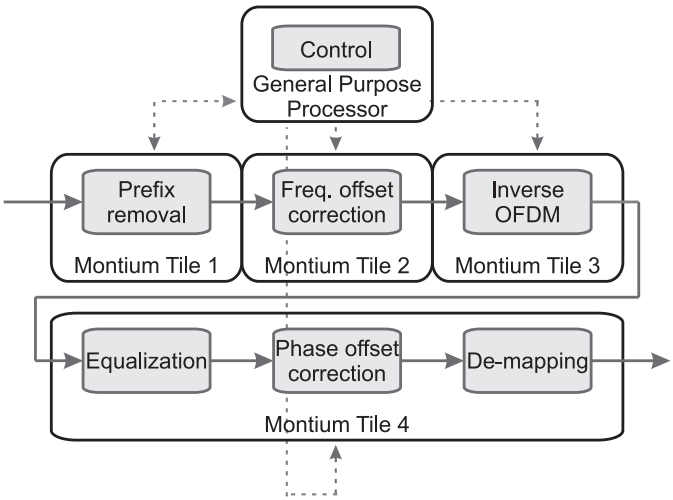

Fig. 9. The HiperLAN/2 receiver in heterogeneous processing tiles.

Irregular tasks, which are outside the algorithm domain of the MONTIUM, are performed in software (i.e. on a GPP). The irregular processes in the HiperLAN/2 receiver are the estimation of frequency offset and computation of equalization coefficients. These coefficients 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. The frequency offset is estimated in software by the GPP once per MAC frame. One OFDM symbol, containing 64 complex-number samples, can be corrected in 67 clock cycles.

A Fast Fourier Transform (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.

The equalizer, phase offset correction and de-mapping 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 carriers 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 (once every 4 μs). Hence, determining the phase offset correction factor in software (i.e. GPP) would create 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, parallel manner the corrected complex-number samples are translated into a bitstream. Hard-decision de-mapping is implemented with LUT functionality. A parametrizable de-mapper has been implemented, which can be used for QPSK, 16-QAM and 64-QAM modulated signals by only changing the LUT table in the memory of the MONTIUM.

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
Minimum processor clock with block communication (@ 100 MHz) [MHz]	25	72	37
Configuration size [bytes]	274	946	576
Configuration time [cycles]	137	473	288

1) Configuration: The total configuration sizes of the MONTIUM are small for the different functions (Table III). The FFT implementation in the MONTIUM requires the largest configuration size, which is less than 1 Kbyte of data. The configuration data can be written into the configuration memory of the MONTIUM in about 500 clock cycles, since 2 bytes are written in one clock cycle. Suppose that the MONTIUM is running at a clock frequency of 100 MHz, then this MONTIUM tile can be (re-)configured in 4.73 μs. Notice that the maximum radio turn-around time of the HiperLAN/2 system is 6 μs [16], so the implemented HiperLAN/2 receiver can

be considered as a real-time dynamically reconfigurable receiver.

2) Frequency scaling: All operations in the physical layer 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 during 67 clock cycles in 4 μs. Hence, the minimum clock frequency of the MONTIUM is 17 MHz, when a streaming on-chip network between the tiles is assumed.

Typically, the clock frequency of the NoC will be fixed and only the clock frequency of the processing tiles can be varied. When we assume the clock frequency of the NoC to be fixed at 100 MHz, then the clock frequency of the MONTIUM for frequency offset correction has to be at least 25 MHz (Table III).

VI. CONCLUSIONS

Because heterogeneous reconfigurable systems might become the future of mobile hardware, we proposed a heterogeneous System-on-Chip (SoC) containing reconfigurable processing elements of different grain sizes. The processing elements in the SoC are dynamically interconnected by a Network-on-Chip (NoC).

The MONTIUM architecture showed to have sufficient flexibility and processing capabilities for implementing next generation wireless communication systems. The feasibility of using heterogeneous hardware is demonstrated by implementing a RAKE receiver and a HiperLAN/2 receiver.

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 procesing. One MONTIUM only has to be partially reconfigured to change the number of fingers in the RAKE receiver. Adjusting the number of fingers from 4 to 2 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.

ACKNOWLEDGEMENT

This research is supported 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.

REFERENCES

[1] Lodewijk T. Smit, Gerard J. M. Smit, and Johann L. Hurink. Energy-efficient Wireless Communication for Mobile Multimedia Terminals. In *Proceedings of The International Conference On Advances in Mobile Multimedia*, pages 115–124, Jakarta, Indonesia, September 2003.

[2] Lodewijk T. Smit. *Energy-Efficient Wireless Communication*. PhD thesis, University of Twente, Enschede, the Netherlands, January 2004.

[3] SDR Forum. `http://www.sdrforum.org`.

[4] Gerard Rauwerda, Jordy Potman, Fokke Hoeksema, and Gerard Smit. Adaptive Wireless Networking. In *Proceedings of the 4th PROGRESS Symposium on Embedded System*, pages 205–211, Nieuwegein, the Netherlands, October 2003.

[5] MuMoR project. `http://www.mumor.org`.

[6] EASY project. `http://easy.intranet.gr`.

[7] A. Abnous. *Low-Power Domain-Specific Processors for Digital Signal Processing*. PhD thesis, University of California, Berkeley, USA, 2001.

[8] V. Baumgarte, F. May, A. Nückel, M. Vorbach, and M. Weinhardt. PACT XPP – A Self-Reconfigurable Data Processing Architecture. In *Proceedings Engineering of Reconfigurable Systems and Algorithms*, pages 64–70, Las Vegas, Nevada, USA, June 2001.

[9] Silicon Hive. `http://www.siliconhive.com`.

[10] Paul M. Heysters. *Coarse-Grained Reconfigurable Processors – Flexibility meets Efficiency*. PhD thesis, University of Twente, Enschede, the Netherlands, September 2004.

[11] Paul M. Heysters, Gerard J. M. Smit, and Egbert Molenkamp. A Flexible and Energy-Efficient Coarse-Grained Reconfigurable Architecture for Mobile Systems. *Journal of Supercomputing*, 26(3):283–308, November 2003.

[12] Jordy Potman, Fokke Hoeksema, and Kees Slump. Tradeoffs between Spreading Factor, Symbol Constellation Size and Rake Fingers in UMTS. In *Proceedings of PRORISC 2003*, pages 543–548, Veldhoven, the Netherlands, November 2003.

[13] H. Holma and A. Toskala. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*. John Wiley & Sons, 2001.

[14] Anna Berno. Time and Frequency Synchronization Algorithms for HIPERLAN/2. Master's thesis, University of Padova, Italy, October 2001.

[15] Max Nilsson. Efficient ASIC implementation of a WCDMA Rake Receiver. Master's thesis, Luleå University of Technology, Sweden, April 2002.

[16] 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), April 2000.