The Journal of Supercomputing, 30, 263–282, 2004

Mapping Wireless Communication Algorithms onto a Reconfigurable Architecture

GERARD K. RAUWERDA g.k.rauwerda@utwente.nl
PAUL M. HEYSTERS p.m.heysters@utwente.nl
GERARD J. M. SMIT g.j.m.smit@utwente.nl
Department of Electrical Engineering, Mathematics & Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands

Abstract. Future mobile communication systems have to be flexible while adapting to environmental conditions and user demands. These systems also have to be energy-efficient as they are used in battery-operated terminals. We expect that heterogeneous reconfigurable hardware can overcome the contradicting requirements in flexibility, energy-efficiency and performance. A coarse-grain reconfigurable processor, called MONTIUM, is presented. An overview of a wireless LAN communication system, HiperLAN/2, and a Bluetooth communication system will be given. Possible implementations of these systems in heterogeneous reconfigurable hardware are discussed. Performance figures of the implemented HiperLAN/2 baseband processing in the MONTIUM architecture are presented. The required performance can be obtained at low clock frequencies with small configuration overhead. The flexibility of the MONTIUM is shown, as the baseband processing of both HiperLAN/2 and Bluetooth is implemented on the same architecture.

Keywords: heterogeneous reconfigurable architecture, System-on-Chip (SoC), MONTIUM, wireless communication algorithms, Software Defined Radio (SDR), adaptivity

1. Introduction

Traditionally, computations are either implemented in hardware or in software running on processors. More recently, new alternatives are introduced, which mix properties of the traditional hardware and software alternatives. The class of these new alternatives is denoted as *reconfigurable computing*. This class of architectures is important because it allows the computational capacity of the architecture to be highly customized to the instantaneous needs of an application, while allowing the computational capacity to be reused in time.

Systems that have to support future mobile communication systems have to be adaptive. 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 (QoS). Furthermore, these architectures have to be extremely efficient as these are used in battery-operated terminals and cost effective as they are used in consumer products. Although energy-efficiency is a major issue in terminals as they draw their energy from small batteries, energy consumption is also an issue in base stations from both a technical (costly cooling of chips and power supplies) and an environmental point of view.

The *Adaptive Wireless Networking* (*AWGN*) project [19] aims at implementation of (adaptive) multi-standard communication systems in reconfigurable embedded systems.

One of the tasks in the project is to map the algorithms found in these communication systems on a platform of heterogeneous reconfigurable architectures. The platform is heterogeneous in the sense that processing is performed in general purpose processors (GPPs), bit-level reconfigurable hardware or in word-level reconfigurable hardware. It is expected that performance and power gains will be achieved by applying dynamically reconfigurable heterogeneous architectures [18].

In this article an introduction to the *Adaptive Wireless Networking* project is given. Research, related to our project, is briefly described in Section 2. An overview of a particular reconfigurable heterogeneous architecture is given in Section 3. In Section 4, some basics of the HiperLAN/2 and Bluetooth communication system are discussed. These systems are currently under investigation in our project. Possible implementations of these systems in reconfigurable hardware are given in Section 5. In Section 6, experiments are done to verify the performance of the implemented HiperLAN/2 receiver. The obtained results of these implementations are shown in Section 7. Further directions of research are given in Section 8 and finally some conclusions are presented in Section 9.

2. Related work

Recently, there have been several announcements of heterogeneous reconfigurable architectures on a single chip [1, 26]. For example, Xilinx delivers a combination of Virtex II FPGA technology and one or more PowerPC(s) 405 on a single chip: the Virtex-II Pro. But conventional reconfigurable processors are bit-level reconfigurable and are far from energy-efficient. Both academy and industry show interest in coarse-grained reconfigurable architectures. The Pleiades project at UC Berkeley [2, 17] focuses on an architectural template for ultra low-power high-performance multimedia computing. 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. A processor for speech coding applications, called Maia, was designed using the Pleiades architecture template. Quicksilver's adaptive computing machine (ACM) [11] technology is intended for low-power mobile devices. Their key observation is that algorithms are heterogeneous by nature. The architecture of the ACM comprises heterogeneous nodes of different granularities. The extreme processor platform (XPP) of PACT [3] is based on clusters of coarse-grained processing array elements (PAEs), which is either an ALU or a memory. Actual PAEs are tailored to the algorithm domain of a particular XPP processor. Silicon Hive [24] offers course-grained reconfigurable block accelerators (e.g. Avispa and Moustique) and stream accelerators (e.g. Bresca) for high performance and low power applications. The architecture is comprised of VLIW-like datapath elements called process storage elements (PSEs).

Much work has been done on Software Defined Radio in the SDR forum context [22]. However this forum mainly focuses on general-purpose processors and they do not concentrate on reconfigurable platforms and energy-efficiency. The Software Defined Radio project [20, 23] at the University of Twente in the Netherlands aims to develop a receiver front-end capable of receiving all WLAN standards. In the Information Society Technologies (IST) EASY project [8] the objective is to develop a cost/power efficient heterogeneous

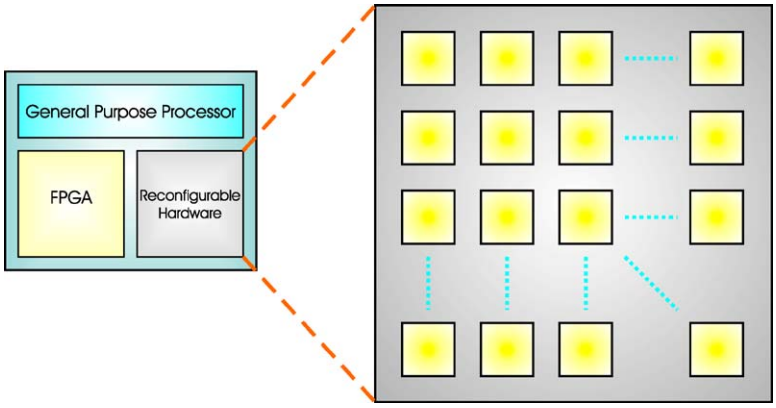

Figure 1. The CHAMELEON heterogeneous SoC architecture.

SoC processor, which handles the baseband processing of 5 GHz WLAN systems. Their heterogeneous SoC consists of embedded FPGA and an ARM processor.

3. Reconfigurable heterogeneous architecture

In the *Chameleon* project [6] at the University of Twente a dynamically reconfigurable heterogeneous System-on-Chip (SoC) is being defined. The SoC contains a general purpose processor, a fine-grained reconfigurable part (consisting of FPGA tiles) and a coarse-grained reconfigurable part. The latter comprises several MONTIUM processor tiles. The proposed SoC is shown in Figure 1. The algorithm domain of the MONTIUM comprises 16-bit digital signal processing (DSP) algorithms that contain multiply accumulate (MAC) operations. A MONTIUM tile is proposed to execute highly regular computational intensive DSP kernels. The irregular parts of the algorithms run on the general purpose processor. The SoC yields a combination of performance, flexibility and energy-efficiency.

3.1. Target architecture: MONTIUM

In this section we give a brief overview of the MONTIUM architecture, because in this article the architecture is used for mapping DSP algorithms of wireless communication systems. More details can be found in [6, 12–14]. Figure 2 depicts a single MONTIUM processor tile. The hardware organisation within a tile is very regular and resembles a very long instruction word (VLIW) architecture. The five identical arithmetic and logic units (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 10 local memories (M01$\cdots$M10) in parallel. The small local memories are also motivated by the locality of reference principle. The ALU input registers provide an even more local

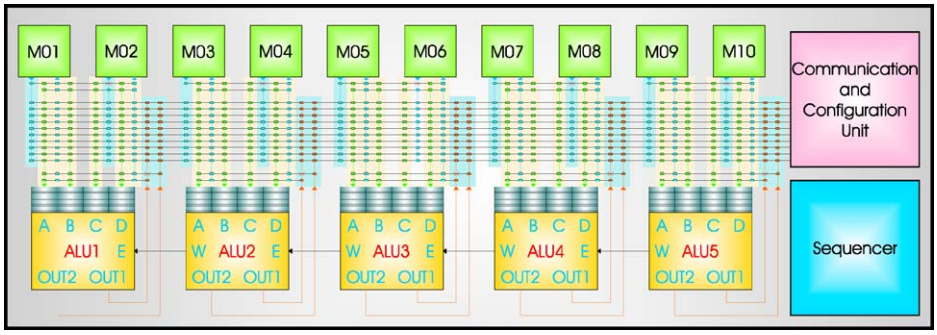

Figure 2. The MONTIUM processor tile.

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 communication and configuration unit (CCU) implements the interface with the world outside the tile. The MONTIUM has a datapath width of 16-bits and supports both integer and fixed-point arithmetic. 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 memory has only a single address port that is used for both reading and writing. 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.

The interconnect provides flexible routing within a tile. The configuration of the interconnect can change every clock cycle. There are ten busses that are used for inter-PPA communication. Note that the span of these busses is only the PPA within a single tile. The CCU is also connected to the global busses. The CCU uses the global busses to access the local memories and to handle data in streaming algorithms. Communication within a PP uses the more energy-efficient local busses. 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 combinatorial and consequentially there are no pipeline registers within the ALU. The diagram of the MONTIUM ALU in Figure 3 identifies two different levels in the ALU. Level 1 contains four function units. A function unit implements the general arithmetic and logic operations that are available in languages like C (except multiplication and division). Level 2 contains the MAC unit and is optimized for algorithms such as FFT and FIR. Levels can be bypassed (in software) when they are not needed.

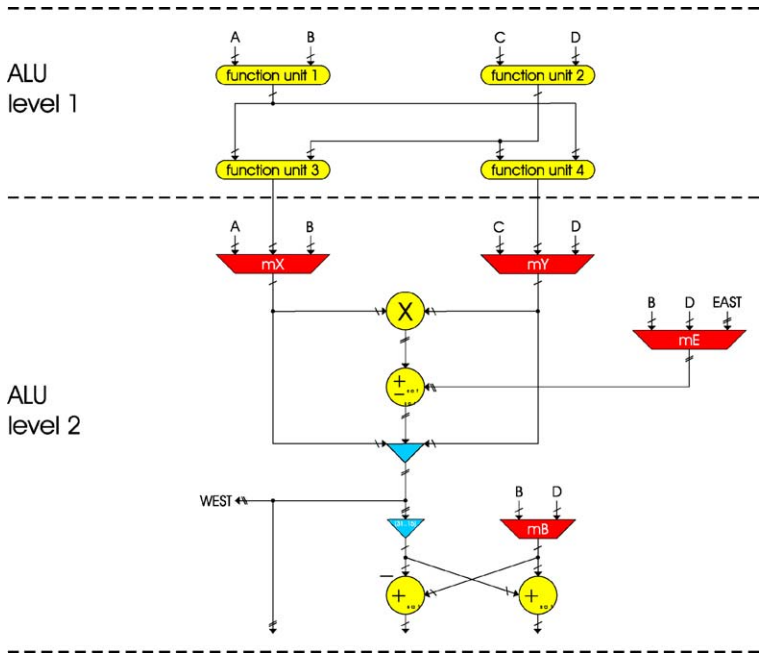

Figure 3. The MONTIUM ALU.

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 32-bit wide East-West connection makes it possible to accumulate the MAC result of the right neighbour to the multiplier result. This is particularly useful when performing a complex-number multiplication, or when adding up a large amount of numbers (up to 20 additions in one clock cycle). The East-West connection does not introduce a clock delay, as it is not registered.

4. Communication systems

The research presented in this article focuses on multi-standard communication systems. By studying the state-of-the-art in Software Defined Radio (SDR) for Bluetooth, HiperLAN/2 and UMTS, we will define a set of typical DSP algorithms used in wireless communications. We present the initial results of mapping a set of characteristic algorithms on our dynamically reconfigurable heterogeneous platform. Currently, the physical layers of two communication systems are subject of our research: a rather complex wireless LAN system, the HiperLAN/2 standard, and a less complex communication system, the Bluetooth standard. These two communication standards have been selected, because they are already part of ongoing research [20, 23] and they are different enough to give an indication whether our approach is feasible or not.

4.1. HiperLAN/2 receiver

HiperLAN/2 [9] is a wireless local area network (WLAN) access technology and is similar to the IEEE 802.11a WLAN standard at the physical level. HiperLAN/2 operates in the 5 GHz frequency band. The air interface is based on time division multiple access (TDMA) and time division duplex (TDD). In a TDMA system a medium access control (MAC) frame is divided in multiple time slots. Multiple users can make use of the same MAC frame by utilizing different time slots. TDD means that time slots for both downlink and uplink are available within the same MAC frame. The task of the physical layer in HiperLAN/2 is to modulate bits that originate from the data link control layer at the transmitter side and to demodulate them at the receiver side. The transmission format of the physical layer is a burst, which consists of a preamble and a data part. Orthogonal frequency division multiplexing (OFDM), which is a special kind of multicarrier modulation, has been used in HiperLAN/2. This modulation technique divides the high data rate information in several parallel bit streams and each of these bit streams modulates a separate subcarrier. 52 subcarriers are being transmitted in parallel per radio channel, which occupies a bandwidth of 20 MHz. However, 4 of these subcarriers are used to transmit pilot tones. 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.

One OFDM symbol is represented by a block of 80 complex-number input samples (in the time domain), so once per 80 samples the receiver has enough information to demodulate the received samples and output the resulting bits. The sample rate of the input signal is 20 MHz, so the duration of an OFDM symbol is 4 μs. For every MAC frame, the physical layer transmits a preamble—a sequence of known OFDM symbols—before the OFDM symbols containing the actual data are transmitted.

The receiver not only has to convert the received signal to data bits by performing the inverse of the transmitter, but it also has to inverse distortions caused by the radio channel. The receiver can roughly be divided into two parts, a time domain part and a frequency domain part.

The location of the functions in the receiver architecture shown in Figure 4, is based upon a trade-off between the necessary resolution that must be reached for a certain correction and the solution with the minimum number of operations. Furthermore, one tries to keep

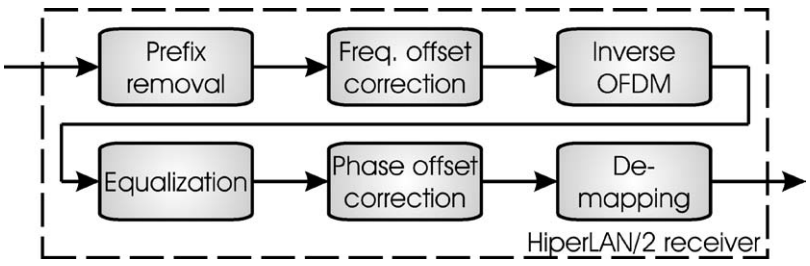

Figure 4. Block diagram of the baseband functions in the HiperLAN/2 receiver.

the corrections independent of each other by deciding the execution order of the functions [15].

We will describe the most important functions (i.e. the most computational intensive) in order to map the functionality on the MONTIUM architecture. We only consider the receiver part of the HiperLAN/2 physical layer, although the transmitter part can be described in a similar way.

4.1.1. Prefix removal. The receiver first needs to synchronize with the transmitter. It does this by detecting the start of a transmission and then detecting the preamble sections by means of correlating the received stream of samples with the known preamble.

A MAC frame always starts with a preamble. A preamble is a sequence of predefined OFDM symbols. Which preamble is used depends on the burst type. In the HiperLAN/2 standard five burst types are defined: broadcast, downlink, uplink with short preamble, uplink with long preamble and direct link. All burst types use one or more preamble sections to precede the data burst. The following preamble sections are defined: A, B (short), B (long) and C. Figure 5 depicts how the preamble sections are used for each burst type and the figure shows that a preamble section consists of repetitions of identical parts.

Because the sampling clock of the receiver hardware is not synchronized with the clock in the transmitter, the OFDM symbol window in the receiver may slowly wander away from the ideal OFDM symbol window. Therefore, the start position of the data of each received OFDM symbol containing data has to be determined. The prefix information of an OFDM symbol is used for this purpose. The maximum of correlations between 16 samples at the beginning of an OFDM symbol and 16 samples received 64 samples later determines the location of the prefix. The prefix is then removed from the OFDM symbol.

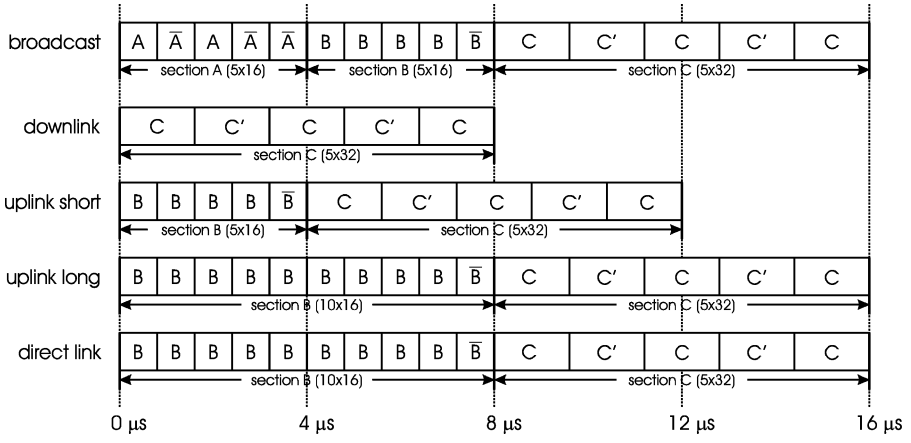

Figure 5. Preambles for all HiperLAN/2 burst types.

4.1.2. Frequency offset correction. There exists a difference in mix frequencies between transmitter and receiver, which is called frequency offset and causes inter-subcarrier interference. The frequency offset is determined in the time domain. In the time domain a frequency offset causes a phase-shift. Hence, the frequency offset can be determined by taking the phase-shift between a known and a received signal. Using information that can be retrieved from the received preamble sections, the receiver can compensate for frequency offsets.

4.1.3. Inverse OFDM. 64 time domain samples represent the useful data part of the OFDM symbol that has to be demodulated. Before demodulation can take place, the subcarrier values must be retrieved from the useful data part. This can be done by applying a fast Fourier transform (FFT) to the vector containing the 64 samples. The FFT efficiently implements a discrete Fourier transform (DFT), given in Equation 1:

$$\hat{f}_n[x] = \sum_{m=0}^{N-1} \tilde{s}_n[m] e^{-j2\pi \frac{xm}{N}}, \tag{1}$$

with $x = 0, \ldots, 63$ and $N = 64$. $\tilde{s}_n$ denotes the vector of input samples in the time domain. $\hat{f}_n$ denotes the vector of samples in the frequency domain. From this vector 52 subcarrier values—48 complex-number data values and 4 pilot values—can be extracted.

4.1.4. Equalization. The received complex-number values may still suffer from distortions that need to be corrected before de-mapping them to a bitstream. The reflections of the transmitted radio signals have different propagation times before they reach the receiver. Slow movement of the receiver introduces a Doppler shift in the received signal. These two effects together result in frequency selective fading. The equalizer can compensate these effects.

4.1.5. Phase offset correction. The various distortions of the transmitted signal caused by a (realistic) wireless channel make it very difficult to estimate the frequency offset exactly at the receiver side. Therefore, the signal will still experience a small frequency offset. This small frequency offset causes a small, linearly changing phase difference between the transmitted and received signals. This phase offset is small and is assumed to be constant during one OFDM symbol. A phase offset corrector can detect this small phase offset by comparing the received pilot values with their predefined value and subsequently compensate for it.

4.1.6. De-mapping. In HiperLAN/2 four mapping techniques are available: BPSK, QPSK, 16-QAM and 64-QAM. Each of these techniques has a different number of bits per complex-number symbol. The de-map function assumes that the most likely symbol to be transmitted was the symbol that maps (given the modulation type) to the complex-number value closest to the received complex-number value. This method of de-mapping is called hard decision de-mapping.

4.2. Bluetooth receiver

The Bluetooth radio technology [10] provides a universal radio interface for electronic devices to communicate via short-range ad hoc radio connections. The task of the physical layer in Bluetooth is to modulate bits that origin from the data layer at the transmitter side and to demodulate them at the receiver side, and vice versa. The Bluetooth system uses packet-based transmission: the information stream is fragmented into packets. In each slot, only a single packet can be sent. All packets have the same format, starting with an access code, followed by a packet header, and ending with the user payload. Single slot, 3-slot and 5-slot packets have been defined.

The channel is a hopping channel with a nominal hop dwell time of 625 μs that corresponds to a single slot. To simplify the implementation, full-duplex communication is achieved by applying time-division duplexing (TDD).

In the Industrial, Scientific and Medical (ISM) band, the signal bandwidth of the Bluetooth system is limited to 1 MHz. For robustness, a binary modulation scheme was chosen. With the mentioned bandwidth restriction, the data rates are limited to about 1 Mbps. Bluetooth uses Gaussian-shaped frequency shift keying (GFSK) modulation with a nominal modulation index of $k = 0.32$. Logical ones are sent as positive frequency deviations, logical zeros as negative frequency deviations [5].

In the receiver part, the transmitted radio signal is converted back into a binary NRZ signal. The radio signal, which is received, conveys all its information in the frequency deviation of the signal.

4.2.1. FM discriminator. One possible way to demodulate a FM signal is by means of FM-to-AM conversion, which is also called a FM-discriminator. The FM-discriminator allows the implementation of low-cost radio units, which is essential for Bluetooth systems. In the FM-discriminator, which is shown in Figure 6, the received signal is multiplied with its delayed version. The signal has to be delayed $\tau = \frac{1}{4f_c}$ seconds [21]. When the center frequency, f_c, of the intermediate Bluetooth signal is 2.5 MHz, one has to set the delay, τ, to 100 ns. Suppose the sample rate of the Bluetooth signal is equivalent to 10 Mega samples per second (MSPS). Consequently, the signal has to be delayed by 1 sample time.

4.2.2. Low Pass Filter. After FM-to-AM conversion, the signal is passed through a Low Pass Filter. The FIR filter is applied in order to pass the slowly varying AM signal with a frequency of 1 MHz and to block all high frequencies that occur due to multiplication. Actually the FIR filter, as depicted in Figure 7, operates at a sampling rate of 10 MSPS.

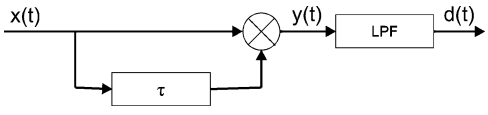

Figure 6. Block diagram of the FM-discriminator.

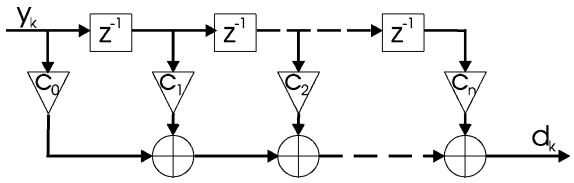

Figure 7. Finite Impulse Response filter.

4.2.3. Threshold detector. Finally, the consecutive bits are detected by a threshold detector, which is not shown in Figure 6. In the threshold detector decisions about the received bit value are taken. For the threshold detector it is sufficient to operate at a sample rate of 1 MSPS, since the bit rate of the Bluetooth communication system is 1 Mbps.

5. Implementation

For both communication systems we will analyze the feasibility of implementing the systems in the target architecture.

5.1. HiperLAN/2 receiver

The HiperLAN/2 receiver has been implemented in the target architecture. The computational intensive kernels are implemented in the MONTIUM. Irregular tasks are performed in software (i.e. GPP). The receiver's implementation block diagram is given in Figure 8.

5.1.1. Prefix removal. In the prefix removal function (and the synchronization function) complex-number samples are detected with matched filters in order to synchronize the receiver. In Table 1 the sizes of the matched filters that are applied in the different functions are shown.

After the prefix is detected, the useful part of the OFDM symbol is copied to a cyclic buffer of length 64.

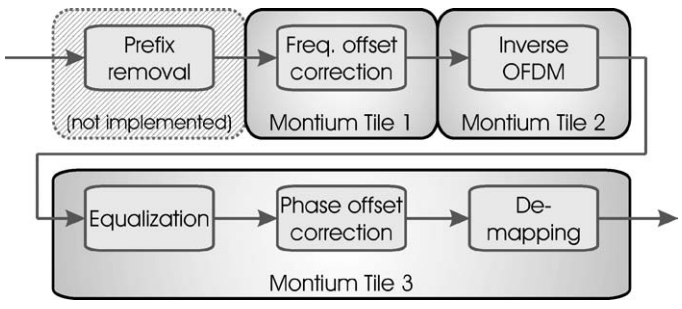

Figure 8. Block diagram of the implemented HiperLAN/2 receiver.

Table 1. Sizes of the matched filters that are applied for synchronization

Detection of	Size of matched filter
Preamble A	16 complex-number samples
Preamble B	16 complex-number samples
Preamble C	32 complex-number samples
Prefix	16 complex-number samples

In Equation 2 the cross-correlation between two complex-number vectors is defined. The output from a matched filter with N samples and two complex-number inputs x and y is:

$$output[l] = \frac{1}{N} \sum_{i=1}^{N} x[i]y[l+i]^*, \tag{2}$$

where y^* denotes the complex conjugate of y.

During matched-filtering the cross-correlation is determined in order to detect preambles and prefixes. One possible way to implement the cross-correlation function in the MONTIUM tile is a separate approach of calculating the real-part and the imaginary-part of the correlation.

With 4 ALUs one can determine the real and imaginary part of the complex-number cross-correlation. In order to result in one valuable output, one has to determine the absolute magnitude of the complex-number cross-correlation and also a scaling factor $1/N$ has to be applied. Since square root calculations are computationally complex, the squared absolute value of the cross-correlation will be calculated instead of the absolute magnitude. Hence the square root calculation is avoided. Scaling with $1/N$ is also neglected, since it only scales the output of the cross-correlation calculation. However, the shape of the correlation function will remain the same, so peaks in the output can still be detected.

Considering the complex-number cross-correlation one can conclude that 4 ALUs are needed in order to perform a cross-correlation calculation of one complex-number sample. When the complex-number cross-correlation of N samples has to be calculated, this calculation will consume $N + 1$ clock cycles. The real part and the imaginary part of the sum can be calculated in parallel using 4 ALUs. In total 1 ALU configuration is used to perform the complex-number correlation. A summary of the requirements for complex-number cross-correlation is given in Table 2.

Table 2. Computational requirements for a correlation of N complex-number samples on the MONTIUM

# Clock cycles	$N + 1$
# ALU configurations	1

Table 3. Computational requirements for frequency offset correction on the MONTIUM for 1 OFDM symbol

# Clock cycles	67
# Crossbar configurations	1
# Memory configurations	4
# ALU configurations	2
# Register configurations	4

5.1.2. Frequency offset correction. The frequency offset correction coefficient is estimated by computing the cross-correlation of preamble C [4]. Figure 5 shows that preamble C consists of a prefix of 32 samples and 2 equal sequences of 64 samples. The sequence consists of 2 parts (C and C′), which are mirrored. The first 16 samples of the two sequences are cross-correlated. The angle of the result of the complex-number correlation determines the total phase offset over 64 samples. The phase offset is linearly proportional with the frequency offset. The phase offset, due to frequency offset, is determined in software (i.e. on the GPP) because it has to be determined infrequently (once every MAC frame).

The calculated phase offset between two samples is stored in the MONTIUM's memory. Based on this phase offset and the index of the data sample in the OFDM symbol, the MONTIUM determines the complex-number offset correction coefficient with its look-up table (LUT) functionality. After the LUT operation the complex-number offset correction coefficient is multiplied with the corresponding complex-number data sample.

The frequency offset corrector is implemented in one MONTIUM TP. One complete OFDM symbol can be corrected for frequency offset in 67 clock cycles. In one OFDM symbol 64 complex-number time samples are multiplied with a complex-number coefficient. A summary of the requirements for frequency offset correction is given in Table 3. The amount of configurations in the MONTIUM is given in terms of defined instruction context. For example if all ALUs are configured with instruction *'a'* and ALU5 is reconfigured with instruction *'b'*, then two configurations (*'aaaaa'* and *'aaaab'*) are active in the MONTIUM. Even when all ALUs are configured with instruction *'a'* and ALU1 is reconfigured with instruction *'c'* while ALU3 is reconfigured with instruction *'d'*, still two configurations (*'aaaaa'* and *'cadaa'*) are active in the MONTIUM.

5.1.3. Inverse OFDM. The inverse orthogonal frequency division multiplexing in HiperLAN/2 is a 64-FFT operation. This operation is performed by radix-2 butterflies and consumes $32 \log_2(64)$ complex multiplications. On the MONTIUM architecture the 64-FFT can be performed in $(\frac{64}{2} + 2) \log_2(64)$ clock cycles [14].

The FFT algorithm can be performed with 4 ALUs and 10 memory banks, which corresponds to one MONTIUM tile, as given in Table 4.

5.1.4. Equalization, phase offset and De-mapping. The channel impulse response is assumed to be time-invariant during one complete MAC frame, because the coherence time of the wireless channel (20 ms) is much longer than the time of a MAC frame (2 ms) [4]. Therefore, the equalizer is trained only once per MAC frame. The equalizer coefficients are

Table 4. Computational requirements for 64-FFT on the MONTIUM

# Clock cycles	204
# Crossbar configurations	17
# Memory configurations	21
# ALU configurations	1
# Register configurations	2

determined by dividing the predefined preamble C values by the received complex-number values. The computation of the equalizer coefficients themselves is done on a GPP. This function is implemented in software because it occurs infrequently (once every 2 ms) and because the MONTIUM is not optimized for complex-number division operations. Once the GPP processor has computed the complex-number equalizer coefficients, they are stored in the memory space of the MONTIUM TP.

During equalization each of the 52 complex-number subcarrier values is multiplied with its corresponding complex-number coefficient. The MONTIUM TP can compute a complex-number multiplication in a single clock cycle using four ALUs.

The four received equalized pilot values are used to determine the phase offset coefficient. Because the phase offset coefficient needs to be determined for every OFDM symbol, it is implemented in hardware (i.e. on the MONTIUM). Computing the phase offset coefficient in software would be expensive due to the frequent communication between GPP and MONTIUM tile.

The phase offset correction itself is a complex-number multiplication of each equalized data value—which is already stored in a local memory of the MONTIUM TP—with the phase offset coefficient. The phase offset correction is implemented in the MONTIUM TP.

After phase offset correction, the de-mapping is performed as hard decision de-mapping. In HiperLAN/2 four modulation schemes are available: BPSK, QPSK, 16-QAM and 64-QAM. A parameterizable quadrature amplitude modulation (QAM) de-mapper was implemented. In 16-QAM de-mapping, both the real and the imaginary part of the complex-number value contain two bits of information. The de-mapping itself is implemented by means of a single look-up table (LUT) containing 64 entries for 16-QAM. In Table 5 the size of the look-up table for the different modulation schemes is given. The implemented de-mapping function is parameterizable, which means that the function can switch between the different modulation types without reconfiguration.

Table 5. Size of LUT for different modulation types

Modulation	Entries in LUT
BPSK	4
QPSK	16
16-QAM	64
64-QAM	256

Table 6. Computational requirements for equalization, phase offset correction & de-mapping on the MONTIUM for 1 OFDM symbol

# Clock cycles	110
# Crossbar configurations	2
# Memory configurations	5
# ALU configurations	3
# Register configurations	13

All equalization, phase offset correction and de-mapping functionality has been implemented in one MONTIUM TP. The computational requirements for implementing these functions in one MONTIUM TP are summarized in Table 6.

5.2. Bluetooth receiver

5.2.1. FM-discriminator. In the FM-discriminator (Figure 6) the incoming FM signal is multiplied with its delayed version. Since the incoming signal is sampled at a frequency of 10 MHz, the signal only has to be delayed with one sample time under the assumption that the intermediate frequency of the incoming signal is 2.5 MHz (Section 4.2.1).

In one MONTIUM tile one can perform 5 multiplications in parallel. An advantage of the Bluetooth FM signals is that all sample values represent real numbers. Consequently, 5 real multiplications can be calculated in one clock-cycle.

5.2.2. FIR filter. The processing delay of the FM-discriminator depends on the amount of samples that have to be processed at the initialization phase; the FIR filter has to be initialized with an amount of samples that is equal to the number of taps.

Suppose that a FIR filter with 10 taps is used, then 10 sample values have to be generated by the FM-discriminator, before the FIR filter is initialized. Since the incoming FM modulated signal has to be delayed with one sample time, $10 + 1$ samples have to be loaded in the local memory before the signal processing can start.

The length of the FIR filter used in the FM-discriminator is not variable. However, in a MONTIUM tile one can perform a FIR filter with a variable amount of taps up to a maximum of 2560 taps. In all ALUs multiply and accumulate operations can be performed. Utilizing the EAST-WEST interconnect between the ALUs gains 1 clock cycle while performing FIR filtering. When there are no EAST-WEST interconnects applied, each ALU has to store its temporary result. The temporary results of all ALUs have to be collected in one extra clock cycle. Depending on the implementation without or with EAST-WEST interconnect, FIR filtering with N coefficients can be performed in $\lceil \frac{N}{5} \rceil + 2$ or $\lceil \frac{N}{5} \rceil + 1$ clock cycles, respectively. Not using the EAST-WEST connection has a positive influence on the maximum clock frequency of the MONTIUM tile.

5.2.3. Threshold detector. In the threshold detector one has to decide whether a '0' or '1' is received. The output of the FIR filter depends on the frequency deviation in the received

Bluetooth signal. The signal at the output of the FIR filter varies between -1 and $+1$. This signal has to be translated into 'received' bits, by applying these rules:

$$\text{if } x > 0 \quad \text{then bit:=1}$$
$$\text{if } x \leq 0 \quad \text{then bit:=0}$$

A decision of a bit value is performed using one ALU and takes only one clock cycle. The actual output bit is determined by the status information bit of the function unit.

6. Experiments

The implemented HiperLAN/2 receiver (without the prefix removal function according to Figure 8) has been tested by means of hardware simulations. A reference model of the HiperLAN/2 receiver was implemented in Matlab (using 64-bit floating-point computations) in order to verify the MONTIUM implementation. Simulations were performed with both the MONTIUM implementation of the receiver and the reference model. In each simulation a complete MAC frame, containing 500 OFDM symbols, was received. The OFDM symbols were 16-QAM modulated in the experiments. We assumed that the receiver was situated in an indoor office environment and indoor hall with non line-of-sight (NLoS) between the transmitter and receiver ('A' channel and 'E' channel, respectively) [16].

Figures 9 and 10 show the results of different simulations. For an 'A' channel and an 'E' channel the curves of both the MONTIUM implementation (marked as 'sim.') and the reference model (marked as 'ref.') are shown. The results for an Additive White Gaussian Noise (AWGN) channel are also included.

The curves of the MONTIUM receiver and the reference model in Figure 9 show hardly any difference. The BER gap between reference and MONTIUM implementation increases when the SNR becomes better. Possibly, for bad channel conditions (i.e. low SNR) the BER of the receiver is mostly determined by the channel. Whereas, for good channel conditions (i.e. high SNR) the limited (16-bit) precision of the hardware has a significant influence on the BER.

The results in Figure 10 show situations wherein the phase offset varies. The results show again that there is only a small difference between the MONTIUM implementation and the reference model.

7. Implementation results

7.1. HiperLAN/2

Mapping the HiperLAN/2 receiver algorithms on the MONTIUM architecture requires knowledge of the typical HiperLAN/2 symbol durations. In Section 5 we have discovered the typical computational requirements of these algorithms. Furthermore we discovered that all algorithms can be mapped on the MONTIUM. In Table 7 we summarize the time consumption

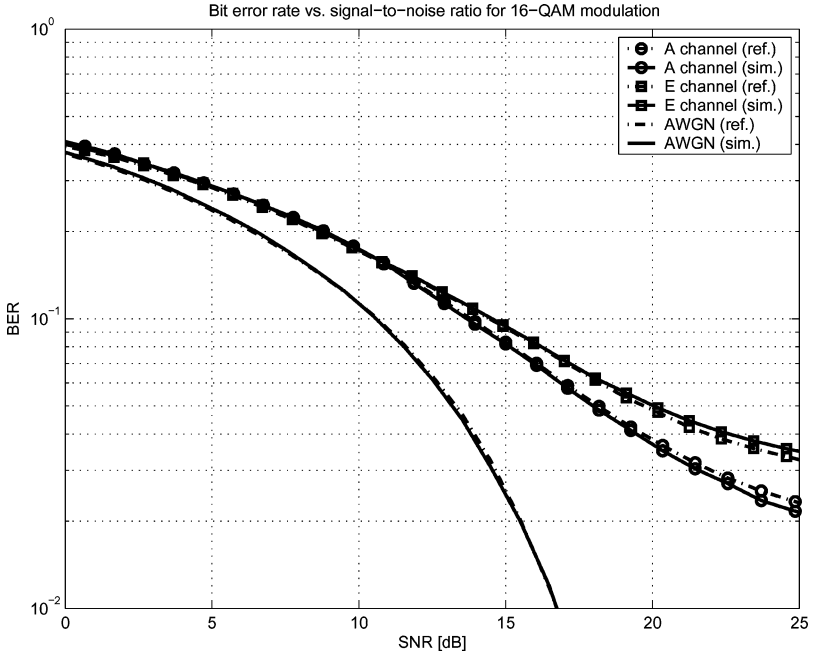

Figure 9. The simulation results of the implemented HiperLAN/2 receiver.

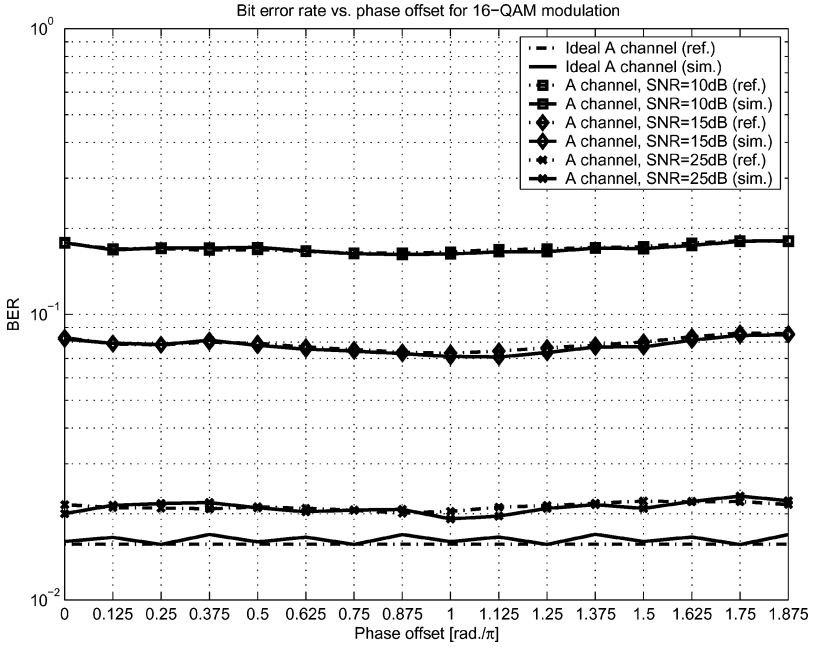

Figure 10. The influence of phase offset on the simulation results.

Table 7. Computational requirements of the HiperLAN/2 receiver on the MONTIUM while processing 1 OFDM symbol

Functional block	Exec. time (clock cycles)	Config. size (bytes)	Config. time (clock cycles)
Frequency offset correction	67	274	137
Inverse OFDM	204	946	473
Equalization, Phase offset correction & De-mapping	110	576	288

and the size of the configuration of the receiver algorithms, while they are performed on a single tile.

One major property of signals in HiperLAN/2 physical layer is the timing aspect of OFDM symbols. All operations in the physical layer are performed on these OFDM symbols. An OFDM symbol has a fixed length of 80 samples. Hence, at a sample rate of 20 MHz the duration corresponds to 4 μs. One should assure that each 4 μs a new OFDM symbol can be processed.

The MONTIUM TP that implements the FFT for the inverse OFDM function has the worst execution time of the three tiles. The FFT has to be performed for every OFDM symbol, once per 4 μs. To obtain the performance this tile has to run at least at 51 MHz. A MONTIUM realized in a 0.12 μm CMOS technology can easily achieve this performance for the FFT. The maximum clock frequency of a MONTIUM depends on the critical path through the ALUs and ranges from 45 to 150 MHz.

The area of a single MONTIUM TP (excluding CCU) is about 2 mm^2 in 0.12 μm CMOS technology. To save area the frequency offset correction and the inverse OFDM function can be combined in the same tile. However, the execution time for this combined functionality would increase to about 271 clock cycles. The required clock frequency would be at least 68 MHz.

The total configuration size of all HiperLAN/2 receiver functionality is only 1,796 bytes for 3 MONTIUM tiles. At 100 MHz for example, the configuration time for the inverse OFDM function is only 4.73 μs.

7.2. Bluetooth

Mapping the Bluetooth receiver algorithms on the MONTIUM architecture requires knowledge about the typical Bluetooth symbol timing. We have seen that there are single slot, 3-slot and 5-slot packets in the standard, which have a duration of 625 μs, 1.875 ms and 3.125 ms, respectively. Actually the slots are not completely occupied by the packets. A single slot packet has a length of 366 bits at maximum, and therefore has a duration of 366 μs. The length of a 3-slot packet is at most 1626 bits, which corresponds to 1.626 ms, and for a 5-slot packet the length is 2870 bits at maximum, which is equal to a duration of 2.87 ms.

From these values of the packet lengths, one can conclude that it should be useful to perform the receiver functions in 'streaming' mode. This means that the processing of the

Table 8. Timing requirements of the Bluetooth receiver on the MONTIUM while processing 1 bit

Functional block	Execution time (clock cycles)	Execution time @ 100 MHz (ns)
FM-discriminator	1	10
N-taps FIR filter	$\lceil \frac{N}{5} \rceil + 2$	$\lceil \frac{N}{5} \rceil \times 10 + 20$
	$\lceil \frac{N}{5} \rceil + 1$	$\lceil \frac{N}{5} \rceil \times 10 + 10$
Threshold detector	1	10

receiver functions starts immediately, and does not wait till a complete packet has been loaded into the local memory. In Bluetooth the data rates are limited to about 1 Mbps, so at most every 1 μs a bit has to be processed by the Bluetooth receiver.

In Table 8 we summarize the time consumption of all the algorithms, while they are performed on a single tile. The processing delays in the table are given while processing is done for a single sample. During initialization of the receiver one has to load $N + 1$ samples in the local memory before the processing can start. This initialization phase will consume $\lceil \frac{N}{5} \rceil + 1$ clock cycles extra. The ceiling value function, $\lceil x \rceil$, computes the smallest integer not less than x.

8. Future work

The presented set of multi-standard algorithms is rather incomplete. In further research, more adaptive algorithms should be considered. An interesting case will be systems that adapt to changing environmental conditions as well as to changing user demands (QoS). A control system that is able to adapt the WCDMA receiver in order to minimize the energy consumption, while satisfying the quality constraints at run-time is presented in [25]. These adaptations should be done at run-time to deal with the continuously changing external environment.

Power consumption is another important issue, since energy-efficiency is a major issue in terminals. At the moment, the normalized (dynamic) energy consumption of the MONTIUM architecture is estimated at 0.5 mW/MHz per tile in .12 technology with a size of 2 mm^2 per tile. Comparison of the algorithm mapping as compared to other types of architectures (DSP, FPGA, etc.) should be done in terms of performance and power consumption.

Finally, scheduling of tasks on the reconfigurable heterogeneous architecture can be implemented in different ways: Multiple tiles can be performing different functions instantaneously, while the data in the tile is changing dynamically or the data stays in one tile, while the tile is reconfigured dynamically. Simulations have to show the advantages and disadvantages of these scheduling strategies.

9. Conclusion

We have analyzed the feasibility of implementing wireless communication systems in coarse-grain reconfigurable hardware. In the *Adaptive Wireless Networking* (*AWGN*) project

we will utilize a dynamically reconfigurable heterogeneous architecture for implementation of multi-standard communication systems. The architecture is heterogeneous in the sense that digital signal processing is performed in general purpose processors, bit-level reconfigurable parts or word-level reconfigurable parts.

Characteristics for mapping the HiperLAN/2 baseband receiver onto reconfigurable hardware are described. We have used our MONTIUM architecture to map the receiver functionality. In order to perform all receiver processing (excluding the synchronization and control part), 3 MONTIUM tiles are required.

The MONTIUM implementation of the HiperLAN/2 receiver was compared against a reference model. A standardized wireless channel model was used to enable a fair comparison. Simulations show that the difference in performance between the MONTIUM implementation and the reference model is negligible.

The functionality of the Bluetooth receiver has a fairly simple signal processing part, which can be implemented in the MONTIUM architecture quite well. The computation delays of the different receiver parts are a few clock cycles.

We showed the flexibility of the MONTIUM architecture. An HiperLAN/2 receiver as well as a Bluetooth receiver was implemented on the same architecture.

Acknowledgments

We thank our colleagues of the Signals & Systems group, Roel Schiphorst and Fokke Hoeksema, for their contribution to our research.

This research is supported by the Freeband Knowledge Impulse programme, a joint initiative of the Dutch Ministry of Economic Affairs, knowledge institutions and industry.

References

1. Altera. *Excalibur Embedded Processor Solutions*, http://www.altera.com/products/devices/excalibur/exc-index.html.
2. A. Abnous. Low-power domain-specific processors for digital signal processing. Ph.D. Dissertation, University of California, Berkeley, USA, 2001.
3. V. Baumgarte, F. May, A. Nckel, M. Vorbach, and M. Weinhardt. PACT XPP—A self-reconfigurable data processing architecture. In *Proceedings Engineering of Reconfigurable Systems and Algorithms*, Las Vegas, USA, pp. 64–70, June 2001.
4. A. Berno. Time and frequency synchronization algorithms for HIPERLAN/2. MSc. Thesis, University of Padova, Italy, October 2001.
5. Bluetooth SIG. Specification of the bluetooth system—Core. Technical Specification Version 1.1, 22 February 2001.
6. Chameleon project. *Reconfigurable Computing in Hand-Held Multimedia Computers*. http://chameleon.ctit.utwente.nl.
7. Chameleon Systems. *Reconfigurable Communications Processor*, http://www.chameleonsystems.com.
8. EASY project, http://easy.intranet.gr.
9. ETSI. Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer, ETSI TS 101 475 V1.2.2 (2001-02), 2001.
10. J. C. Haartsen. The bluetooth radio system. In *IEEE Personal Communications*, 7(1):28–36, 2000.
11. G. Heidari and K. Lane. Introducing a paradigm shift in the design and implementation of wireless devices. In *Proceedings Wireless Personal Multimedia Communications*, Aalborg, Denmark, vol. 1, pp. 225–230, September 2001.

12. P. M. Heysters, G. J. M. Smit, and E. Molenkamp. A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems. *Journal of Supercomputing*, Kluwer Academic Publishers, 26(3): 283–308, November 2003.
13. P. M. Heysters, G. J. M. Smit, and E. Molenkamp. Montium—balancing between energy-efficiency, flexibility and performance. In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'03)*, Las Vegas, USA, 235–241, June 2003.
14. P. M. Heysters, H. Bouma, J. Smit, G. J. M. Smit, and P. J. M. Havinga. A reconfigurable function array architecture for 3G and 4G wireless terminals. In *Proceedings of 2002 World Wireless Congress*, San Francisco, U.S.A, pp. 399–404, May 2002.
15. L. F. W. van Hoesel. Design and implementation of a software defined HiperLAN/2 physical layer model for simulation purposes. MSc. Thesis, University of Twente, Enschede, the Netherlands, August 2002.
16. J. Medbo and P. Schramm. *Channel Models for HIPERLAN/2*, ETSI/BRAN 3ERI085B, March 1998.
17. Pleiades project—*Ultra-Low-Power Reconfigurable Computing*, http://bwrc.eecs.berkeley.edu/Research/Configurable_Architectures/.
18. J. Rabaey. Silicon architectures for wireless systems. Tutorial Hotchips symposium, August 2001.
19. G. Rauwerda, J. Potman, F. Hoeksema, and G. Smit. Adaptive wireless networking. In *Proceedings of the 4th PROGRESS Symposium on Embedded System, Nieuwegein*, the Netherlands, October 2003.
20. R. Schiphorst, F. W. Hoeksema, and C. H. Slump. *A Bluetooth-Enabled HiperLAN/2 Receiver*. IEEE. VTC 2003 - Fall, Orlando, Florida, USA, October 2003.
21. R. Schiphorst, F. Hoeksema, and K. Slump. *Bluetooth Demodulation Algorithms and their Performance*, 2nd Karlsruhe Workshop on Software Radios, pp. 99–106, March 2002.
22. SDR Forum, http://www.sdrforum.org.
23. Software-Defined-Radio project—*A Bluetooth-HiperLAN/2 SDR receiver*, http://www.sas.el.utwente.nl/home/SDR/.
24. Silicon Hive, http://www.siliconhive.com.
25. L. T. Smit, G. J. M. Smit, and J. L. Hurink. Energy-Efficient Wireless Communication for Mobile Multimedia Terminals. In *Proceedings of the International Conference on Advances in Mobile Multimedia (MOMM 2003)*, Jakarta, Indonesia, pp. 115–124, September 2003.
26. Xilinx. *Virtex-II Pro Platform FPGAs*, http://www.xilinx.com/virtex2pro/.