An OFDM receiver implemented on the coarse-grain reconfigurable Montium processor

Gerard K. Rauwerda, Paul M. Heysters, Gerard J.M. Smit
University of Twente, Department of EEMCS
P.O. Box 217, 7500 AE Enschede, the Netherlands
g.k.rauwerda@utwente.nl

Abstract—Future mobile terminals become multimode communication systems. In order to handle different standards, we propose to perform baseband processing in heterogeneous reconfigurable hardware. OFDM is one of the techniques that exist in multimode communication systems. As an example, we present the results of implementing an HiperLAN/2 receiver in reconfigurable hardware. The receiver can be implemented with small configuration overhead, and the required performance can be obtained at low clock frequencies.

I. INTRODUCTION

Future mobile communication systems tend to become flexible devices capable of handling multiple wireless communication standards. Furthermore, these flexible systems will be aware of their environment and adapt to this environment. Since mobile devices are battery-powered energy-efficiency is an important issue. In the Adaptive Wireless Networking (AWGN) project [8] we aim at the implementation of adaptive wireless communication systems in heterogeneous reconfigurable hardware.

Orthogonal Frequency Division Multiplexing (OFDM) is a promising candidate for mobile communication systems. It is a multiple carrier modulation technique that eliminates the need for complex equalizers and utilizes the bandwidth efficiently.

This paper addresses various aspects on implementing an OFDM receiver in heterogeneous reconfigurable hardware. In section II a hardware platform for future mobile devices is presented. The HiperLAN/2 standard is used as an example. The main properties of HiperLAN/2 are presented in section III. The baseband processing part of the HiperLAN/2 receiver is mapped on reconfigurable hardware in section IV. In section V simulations are performed to show the performance of the implemented receiver.

Finally, we conclude our approach in section VI and present directions for future work.

II. RECONFIGURABLE HARDWARE

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 algorithms with the granularity of the hardware. Four processor types are distinguished: general-purpose processor, fine-grained reconfigurable hardware, coarse-grained reconfigurable hardware and dedicated hardware.

We propose a System-on-Chip (SoC), which consists of the above mentioned processors types (Figure 1). The different processors are interconnected to each other by a Network-on-Chip (NoC). Both the SoC and NoC are dynamically reconfigurable, which means that the programs (running on the reconfigurable processors) as well as the communication links between the processors are defined at run-time. It is expected that performance and power gains are achieved by applying dynamically reconfigurable heterogeneous architectures [7].

A. The Montium reconfigurable architecture

The Montium is an example of a coarse-grain reconfigurable processor. The Montium [4] targets the 16-bit digital signal processing (DSP) algorithm domain. A single Montium processor 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.

The TP is the computing part that can be configured to implement a particular algorithm. The hardware organization of the tile processor is very regular. The five identical ALUs 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 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 relatively simple sequencer controls the entire TP. The sequencer selects configurable 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. The memory can also be used as a lookup table for complicated functions that cannot be calculated using an ALU, such as sine or division (with one constant).

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 consequently there are no pipeline registers within the ALU. Neighbouring ALUs can also communicate directly: 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.

III. HIPERLAN/2 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.

The receiver not only performs the inverse of the transmitter, it also has to correct for all the distortions that are introduced in the wireless channel. Figure 3 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, DAB, DRM, are generally different in the number of carries and the transmission bandwidth. Table I summarizes the OFDM properties for different standards.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>PROPERTIES OF THE DIFFERENT OFDM STANDARDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiperLAN</td>
<td>DAB</td>
</tr>
<tr>
<td>Bandwidth [MHz]</td>
<td>20</td>
</tr>
<tr>
<td># carriers</td>
<td>52</td>
</tr>
<tr>
<td>Symbol time [µs]</td>
<td>4</td>
</tr>
<tr>
<td>Frame time [ms]</td>
<td>2</td>
</tr>
</tbody>
</table>

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 [1].

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.

IV. IMPLEMENTATION

We have implemented a HiperLAN/2 receiver in heterogeneous reconfigurable hardware. The implementation of the baseband processing part was implemented in a combination of general purpose processor and coarse-grained reconfigurable hardware. The physical layer of the HiperLAN/2 receiver [3] has been implemented in three MONTIUM tiles. Figure 4 shows the functional blocks in the receiver that are implemented in each MONTIUM tile. The synchronization part (prefix-removal) has still not been implemented. Nevertheless the function, which consists of correlation operations, can easily be implemented in the MONTIUM architecture.

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. Table II shows the results of partitioning the receiver’s functionality over the MONTIUM and the general-purpose processor.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>PARTITIONING OF THE HIPERLAN/2 FUNCTIONALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implemented in</td>
<td>Block size</td>
</tr>
<tr>
<td>Determine frequency offset</td>
<td>software</td>
</tr>
<tr>
<td>Determine equalizer coefficients</td>
<td>software</td>
</tr>
<tr>
<td>Prefix removal</td>
<td>-</td>
</tr>
<tr>
<td>Frequency offset correction</td>
<td>MONTIUM</td>
</tr>
<tr>
<td>Inverse OFDM</td>
<td>MONTIUM</td>
</tr>
<tr>
<td>Equalizer, Phase offset correction</td>
<td>MONTIUM</td>
</tr>
</tbody>
</table>

The frequency offset correction is implemented in one MONTIUM tile. During correction every complex-number sample is multiplied with the frequency offset correction factor. The correction factor is determined with a Lookup table (LUT) based on
the estimated frequency offset. 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. 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 for QPSK, 16-QAM and 64-QAM modulated signals by only changing the LUT table in the memory of the MONTIUM.

<table>
<thead>
<tr>
<th>Configuration time [cycles]</th>
<th>Frequency offset correction</th>
<th>Inverse OFDM</th>
<th>Equalizer</th>
<th>Phase offset, De-mapper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time [cycles]</td>
<td>67</td>
<td>204</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>Communication time [cycles]</td>
<td>128</td>
<td>116</td>
<td>&lt; 100</td>
<td></td>
</tr>
<tr>
<td>Minimum system clock with streaming communication [MHz]</td>
<td>17</td>
<td>51</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Minimum processor clock with block communication (at 100 MHz) [MHz]</td>
<td>25</td>
<td>72</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Configuration size [bytes]</td>
<td>274</td>
<td>904</td>
<td>356</td>
<td></td>
</tr>
<tr>
<td>Configuration time [cycles]</td>
<td>137</td>
<td>473</td>
<td>288</td>
<td></td>
</tr>
</tbody>
</table>

### A. Configuration

The total configuration sizes of the MONTIUM are small for the different functions (as seen in Table III). MONTIUM tile 2, on which the inverse OFDM is performed, requires the largest configuration size. The configuration of tile 2 contains 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 tile 2 can be (re-)configured in 4.73 µs. Notice that the maximum radio turn-around time of the HiperLAN/2 system is 6 µs [2], so the implemented HiperLAN/2 receiver can be considered as a real-time dynamically reconfigurable receiver.

### B. Flexible clock frequency

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.

The clock frequency of the reconfigurable processors is important from an energy-efficiency point-of-view. The dynamic power consumption of the heterogeneous tile processor depends on the clock frequency. Hence, the lower the clock frequency, the lower the supply voltage can be and the lower the dynamic power consumption will be. Typically, the clock frequency of the NoC, connecting the reconfigurable processors, will be fixed and only the clock frequency of the reconfigurable processor 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).

### V. Simulation

For functional simulation we have used a co-simulation environment in which the VHDL simulation results of the implemented HiperLAN/2 receiver are compared with the results of a reference model, specified in Matlab. During each simulation a downlink burst, containing 500 OFDM symbols, was received. The first two OFDM symbols in each frame, the so-called Preamble C [3], were used for frequency offset and channel estimation. The received information was 16-QAM modulated, so in each simulation...
95,616 bits were received. The receiver was simulated for different realistic channel settings [6].

The curves in Figure 5 show the performance of the implemented HiperLAN/2 receiver in the MONTIUM ('Montium') and the reference model in Matlab ('Reference'). Although, the error correction in HiperLAN/2 was not implemented explicitly, we can calculate the upper bounds of the BER after error correction [9] with the obtained results. Figure 6 depicts these upper-bounds.

![Graph showing BER vs. SNR for different channel settings](image)

**Fig. 5.** The BER before error correction for different settings.

![Graph showing Montium BER for coded 16-QAM modulation](image)

**Fig. 6.** The BER after error correction for different settings.

The HiperLAN/2 receiver needs a BER of $2.4 \times 10^{-3}$ in order to reach the minimum defined sensitivity of 10% packet error rate [3]. The upper bounds show that the minimum sensitivity can be met with the MONTIUM implementation. The BER performance of the MONTIUM implementation compares well with the simulation results in [5].

VI. CONCLUSION AND FUTURE WORK

We proposed heterogeneous reconfigurable hardware as a future technology for adaptive multi-mode communication systems. The feasibility of using heterogeneous hardware is demonstrated by implementing an HiperLAN/2 receiver. Our experiment showed that the performance obtained by the HiperLAN/2 implementation satisfied to the performance requirements. The required performance can be obtained at low clock speeds, and with low configuration overhead.

Currently, we aim at the implementation of the baseband processing part of an UMTS receiver. Moreover, we studied the baseband processing in Bluetooth. Based on the inventarisation in Table I we will study the requirements for implementing different OFDM systems in the same reconfigurable hardware. These investigations will result in requirements for a multi-mode receiver in reconfigurable hardware.

ACKNOWLEDGEMENT

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


