Rapid Prototyping Design of a 4×4 BLAST-over-UMTS System

Maxime Guillaud[‡], Suman Das[†], Andreas Burg[§], Markus Rupp[*], Eric Beck[†]
[‡]Institute EURECOM, France, maxime.guillaud@eurecom.fr
[†]Wireless Research Laboratory, Lucent Technologies,NJ, {sumand,ericbeck}@lucent.com
[§]ETHZ, Switzerland, apburg@iis.ee.ethz.ch
[*]Inst. of Comm. & Radio Frequency Engin., TU Wien, Vienna, Austria, mrupp@nt.tuwien.ac.at

Abstract

BLAST techniques to increase the utilized bandwidth in commercial systems are currently feasible. This paper describes the design of a UMTS prototype, supporting four transmit and four receive antennas, achieving almost four times the capacity of a conventional system. Various parts of the transmitter and receiver are mapped on FPGAs and fast DSPs communicating via a specially designed communication link. The system was entirely designed using C code, embedded in SIMULINK S-functions for simulation, and, after validation, automatically mapped onto the hardware platform.

1 Introduction

Recent improvements in understanding rich scattering environments give rise to hope that the upcoming UMTS system can support more users and higher bandwidths by applying BLAST techniques [1, 2, 3].

After developing a methodology for mapping C-code simulations automatically onto real-time hardware platforms [4, 5], the method has been improved and challenged on a larger system design. A UMTS transmission with four transmit and four receive antennas has been designed, enabling four times the data rate on a single CDMA code.

The entire simulation setup is shown in Figure 1. The transmitter supports four data streams through a 4-by-4 channel. The baseband signal is four times oversampled and Gaussian noise is added. The receiver, consisting of automatic gain control(AGC), synchronization unit (SYNC), square-root raised cosine filter (SRRC), receiver front end(RFE), and Multiple-Input-Multiple-Output Decoder(MIMO-DEC), processes the received signal. This setup matches the actual hardware implementation, with the RFE implemented in an FPGA and the MIMO-DEC running on a DSP. The various blocks are discussed in detail in the following sections. Finally, the simulation environment and the implemented system on a SUNDANCE experimental board is described.

2 Transmitter

The transmitter block from [4, 5] was copied four times and mapped onto a single V1000 FPGA from XILINX. Only 50% of the chip area was utilized supporting primary and secondary synchronization channels, a unique pilot channel for each transmit antenna, and up to 14 users with individual spreading factors ranging between 4 and 256 and gains between 0 and 255. 3L-Diamond's streaming software (WinServer32) is being used to connect the transmitter core to a simple RLP manager on a host-PC receiving and forwarding UDP packets from an ethernet port.

3 Channel Models

Currently, several MIMO channel models are under investigation. A straightforward 4-by-4 channel model defining 16 channels (one from each transmit to each receive antenna) was derived from the power profile and delay spreads of one-by-one UMTS channel models defined in the standard. The channel model can be defined as:

$$\mathbf{y}(k) = \mathbf{RC}(k)\mathbf{Tx}(k) \quad (1)$$

$$\mathbf{C}(k) = \mathbf{H}_0(k)\delta(k) + \mathbf{H}_1(k)\delta(k-\tau_1) + ... + \mathbf{H}_3(k)\delta(k-\tau_3) \quad (2)$$

$$[\mathbf{H}_i(k)]_{lm} = z_{lm} J_{lm}(kT_c \mathbf{f}_D 2\pi) g_i. \quad (3)$$

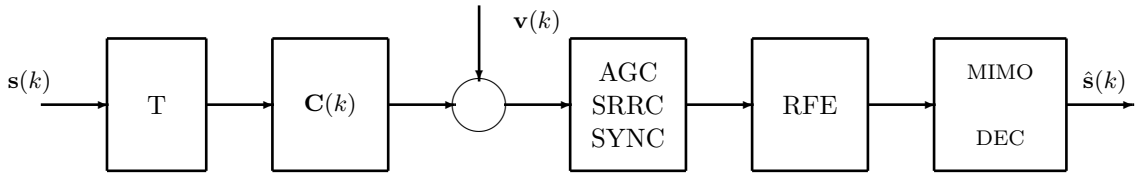

Figure 1: *BLAST over UMTS prototype setup.*

The transmitted symbols $\mathbf{x}(k)$ are filtered by the correlation matrix of the transmitter antenna array $\mathbf{T}$, followed by the time-variant channel $\mathbf{C}(k)$ and finally by the correlation matrix of the receiver antenna array $\mathbf{R}$. Multipath rays arrive at time $0, \tau_1 ... \tau_3$. The instantaneous values of each ray from transmit antenna l to receive antenna m is defined by a power profile z_{lm}, a Jakes fading component and a normalizing term g_i. Note that this model assumes that all rays in a MIMO channel are aligned in time, an assumption that has been validated by recent outdoor measurements. It should be noted that the channel model is only for simulation purposes and does not require real-time implementation.

4 Preprocessing

The first block of the receiver includes a digital AGC, the required eight squared-root-raised-cosine (SRRC) filters (two per receive antenna) and the synchronization unit (SYNC). A digital automatic gain control stage has been implemented as a first stage to effectively reduce the 14 bit dynamic range from the A/D converters down to 8 bit for the SRRC filters. The AGC together with the SRRC filter blocks for one channel is depicted in Figure 2. The magnitude of the incoming I and Q signal is approximated by

$$|I + jQ| \approx \frac{3}{8}(|I| + |Q|) + \frac{5}{8}\max(|I|, |Q|)\,. \qquad (4)$$

This estimate is first block-averaged over P samples and then used to update a recursive smoothing filter. It is further necessary to synchronize the updates with the channel estimation block to ensure that each channel estimate is not corrupted by the switching of the AGC (not shown here). The so obtained smoothed estimate of the magnitude needs to be inverted. A precise division is costly in timing and complexity, however in this case does not require a high precision. It was therefore decided to approximate a floating point division, by shifting the values into the range [1,2] and further inverting the number by a linear approximation in the range [0.5,1]. This estimation yields the 6-bit mantissa of the required gain and a gain exponent that is determined by the position of the highest used bit in the amplitude average. It is applied to the incoming I and Q signals through a 14×6 bit multiplication and a shift operation. The result is then truncated to 8 bit. Independent square root raised cosine filters are applied to the inphase and quadrature components of all four receive antennas. As their 25 coefficients are symmetrical, only 13 multipliers are required for each one of the filters. The AGC together with the SRRC filters requires about 40% of a V1000 XILINX FPGA.

The synchronization detector reliably finds the beginning of a frame and sends this information to the gain update of the AGC as well as to the receiver front end.

5 Receiver Front End

The detailed schematic of the receiver front end is shown in Figure 3. It consists of a timing reference that generates the scrambling and OVSF codes for pilots (PCG) and user codes (UCG), a frequency offset compensation mechanism (FOFF), channel estimation, a rake receiver, and a finger assignment subsystem. The RFE is clocked at 16 times the chip frequency and uses a four times oversampled input signal at each of the four receive antennas. This allows four processing cycles per sample which can be used to interleave the samples of the four receive antennas to process them sequentially. The scheme inherently infers a high degree of pipelining and heavily uses time multiplex resources sharing. This type of architecture is very suitable for FPGA implementations and is highly recommended by XILINX and other vendors in various application notes.

5.1 Channel Estimation

The channel estimation algorithm is by far the block with the highest complexity. It needs to handle 16 sub-channels with 64 taps each (4 times over-sampling, spanning 4μs). A straight forward matched filter implementation would require up to 16 FIR filters with the length of the training sequence (1024 chips at $4\times$

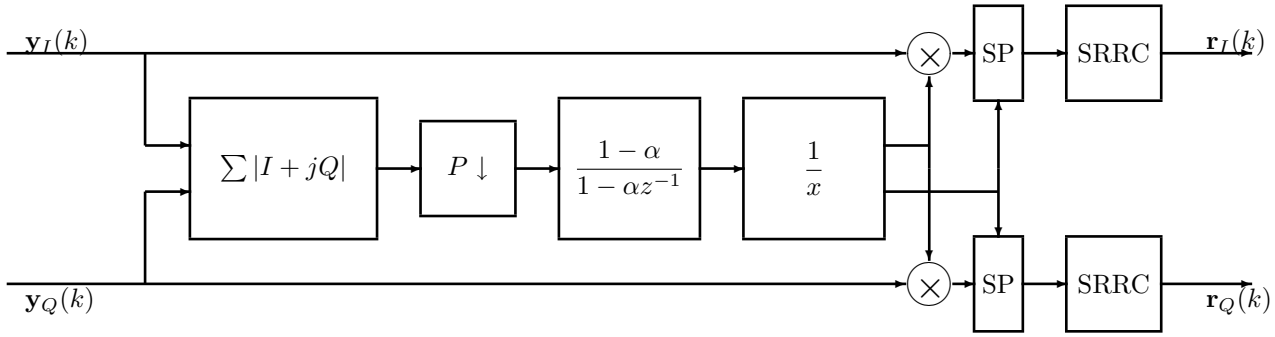

Figure 2: *AGC and SRRC per channel.*

oversampling) or alternatively a bank of 16×64 correlators. Both implementations would exceed the resources of a XILINX Virtex-1000 FPGA by far and would be rather inefficient in terms of area utilization. Instead a more advanced architecture was chosen combining FIR and correlator solution to implement a 4×1 channel estimation, by imposing minor restrictions on the choice of the pilot codes at the transmit antennas. This technique also allows to replace a great number of registers with memory as it avoids the necessity to access them all in parallel. A further area reduction was achieved through time multiplexing and sequentially processing of the receive antennas. The area requirement for this block was reduced to about 15% of a Virtex-1000 device. For details please refer to [13].

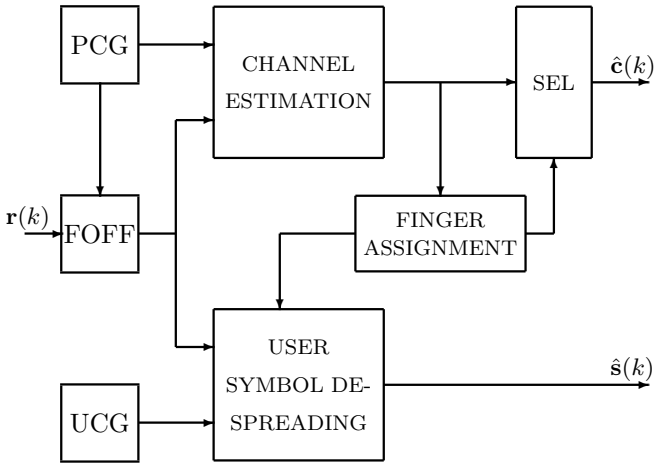

Figure 3: *Receiver front end.*

5.2 Finger Assignment & RAKE Receiver

The development of the channel tracking and finger assignment mechanism also required a particularly carefull design to suit the peculiar channel structure provided by the multiple antenna configuration. To identify the most suitable positions for the RAKE fingers a common channel power profile is approximated by summing up the absolute values of the real and imaginary part of the 16 channel profiles at each tap. This profile is then searched to determine the best positions for the RAKE fingers considering their power and avoiding correlation between them. The entire process is done on the fly as the channel estimate is being generated, so that there is no need to store the channel information for the finger search process, which would require a large amount of additional memory. The same holds for the determination of the finger coefficients.

The RAKE receiver is implemented as a set of four delay lines, one for each receive antenna. They are utilizing the FPGA's internal dual ported memories and are tapped by 16 parallel correlators allowing four independent fingers per receive antenna. Alternatively to the delay line implementation, an architecture was considered allowing the correlators to start the integration process independent of each other at their respective delays. While this avoids the necessity to implement a relatively long delay line it complicates the finger management process, as correlators might skip chips or even entire symbols as their position changes. The additional complexity inferred to control and minimize these effects exceeds the complexity required by the delay line and therefore the first implementation alternative was chosen.

5.3 MIMO-Decoder Interface

The interface between the RFE and the the MIMO decoder relies on a feed-forward structure, enabling the use of a simple one-way communication link through two FIFOs. One of them transfers the soft symbols, the other the channel estimate. Each channel update is marked with a two bit identifier. The symbols that were received during the same time period carry the same identifiers to align them with the corresponding channel information for decoding.

6 MIMO Decoding algorithms

Many different MIMO decoding algorithms have been implemented. Their details are presented in a companion paper [11]. The entire algorithms are mapped onto TI-C67 DSPs since they allow flexible programming for the irregular algorithms. An FPGA design of such a structure would have taken a much longer time. Worth mentioning here is our approach that allows separating the matrix computations and the actual symbol decoding. No matter what algorithm is applied, the required matrix inverses only depend on the channel information and do not require further updates as long as no new channel information arrives. Since the channel estimation is averaged over several pilot symbols, the matrix update is not required for many data symbols (in particular for low spreading factors) and thus the complexity of the algorithms is only moderate. Depending on the complexity of the algorithm one DSP can support one user with spreading factor 32 to 64, or, alternatively, more users with correspondingly higher spreading factors.

7 Simulation Platform

Several simulation environments exist to provide the developers with necessary tools to design and validate their algorithms. Among those SIMULINK from MATHWORKS is wide spread at universities as well as companies. It is a graphical environment that allows to partition a larger simulation problem easily into smaller functional blocks. Due to its graphical representation, all in- and output ports of such blocks are fully described and the representation serves as a good tutorial presented immediately to everybody in the design team. Such a tool helps structuring the code as well as keeping responsabilities for certain team members restricted to specific blocks. SIMULINK provides a large library with predefined functional blocks and can be combined with MATLAB, thus allowing to quickly write and test code in the MATLAB language while the entire toolset of MATLAB remains available.

In our design, particularly S-functions were used to implement the various functional blocks. The advantage here is that these operations can also be written in ANSI-C and compiled, thus allowing faster simulation runs as compared to using pure Matlab code. A particular hurdle seems to be bringing a C-code into the specific form of an S-function. The newest SIMULINK revision 12 provides a tools for this while older revisions only provide a template to fill out which is a very time consuming process. Our goal was to reuse the entire simulation code directly on the hardware platform, avoiding the necessity to rewrite/retarget the code manually with all the necessary additional debugging. To achieve this goal a mapping software was written that translates one *golden code* into any description necessary; i.e., into S-functions for simulation purposes, into code to run on a TI processor or into a C-code that can be mapped via automatic tools into VHDL and subsequently into FPGAs. The *golden code* is written in form of a GENERIC-C code as it is common in commercial system design tools such as COSSAP from SYNOPSYS or SPW from CADENCE. Such a GENERIC code is basically plain C code with a header that defines all necessary in- and output ports and their corresponding data rate. For the simulation for example the automatic mapping tool figures out how to map this information correctly into a valid S-Function that can be plugged into a SIMULINK simulation.

8 Implementation Platform

Since a UMTS receiver requires a huge amount of bit level manipulations at a fairly high rate, general purpose DSPs are very inefficient to implement those. FPGAs offer the right amount of flexibility at the cost of higher turnaround times for algorithmic modifications. On the other hand DSPs are very well suited for mult/add operations on irregular code as it is common especially for decoder algorithms that rely on matrix inversions. It was thus decided to implement the receiver front-end together with AGC, SRRC and synchronization unit on FPGAs, while the actual MIMO decoding algorithms were implement on DSPs. Not many experimental platforms provide the co-existence and even more, the co-operation of DSPs and FPGAs. The experimental boards from SUNDANCE (www.sundance.com), offer a large range of TI-DSPs and Xilinx FPGAs together with A/D and D/A converter modules.

The design as described in the previous sections allows to implement the entire receiver for a 4×4 BLAST system on two FPGAs of the V1000 XILINX series and one TI C-67. As mentioned before the GENERIC-C code was mapped directly to run on DSPs and FPGAs. For the TI-DSPs, the code is enriched by the real-time system 3L Diamond and by interface drivers to support the in- and output ports. More specifically, SUNDANCE provides two 16bit wide busses with FIFO buffers on their DSP boards that can sustain up to 100Mwords per sec. data transfer. The same 16bit wide bus also exists on the FPGA modules allowing different modules to communicate with each other.

Mapping algortithms into the FPGA world requires additional steps. The ANSI-C code is enriched by fix-point data types from FRONTIER DESIGN (www.frontierd.com). They come with a library that can easily be combined with the standard C compilers that support SIMULINK. Once the code runs entirely in fixed-point C data types, the algorithm can automatically be converted into VHDL by applying Frontier's A|RT-BUILDER tool. From there VHDL synthesizer continue to map the code into the required format for FPGAs. Note that this design path requires the developer to write the C code specifically to support mulirate designs. In particular, finite-state-machines need to be hand coded in C. Modern EDA tools like A|RT-DESIGNER (see [12]) allow even to add the control logic to a given C code description.

9 Conclusion

This paper outlined the architecture of a UMTS-based MIMO receiver frontend prototype, the rapid prototyping methodology and required tools to realize it. With the described C-based design methodology it was possible to realize an inital system, to run cycle accurate simulations and to assess the performance and the hardware complexity of the individual blocks at an early stage of the design. This allowed us recognizing critical parts such as the channel estimation early on and looking for optimized architectures with a reasonably low complexity.

Acknowledgments

The authors would like to thank Sue Walker and David Haessig for their excellent XILINX implementation work.

References

[1] G.J.Foschini, M.J.Gans, "On Limits of Wireless Communications in a Fading Environment when Using Multiple Antennas", Wireless Personal Communications, No. 6, 1998, pp. 315-335.

[2] P.W.Wolniansky, G.J.Foschini, G.D.Golden, R.A.Valenzuela, "V-BLAST: An architecture for achieving very high data rates over rich-scattering wireless channels", in Proc. ISSSE-98, Pisa, Italy.

[3] G.D.Golden, G.J.Foschini, R.A.Valenzuela, P.W.Wolniansky, "Detection Algorithm and Initial Laboratory Results Using V-BLAST Space-Time Communication Architecture", Electronics Letters, Vol. 35, No. 1, pp. 11-14, Jan. 1999.

[4] M.Guillaud, A.Burg, L.Mailaender, B.Haller, M.Rupp, E.Beck, "From Basic Concept to Real-Time Implementation: Prototyping WCDMA Downlink Receiver Algorithms - A Case Study", 34^{th} Asilomar Conference, Monterey, California, Oct. 2000.

[5] A.Burg, B.Haller, M.Guillaud, M.Rupp, E.Beck, L.Mailaender, "A Rapid Prototyping Methodology for Algorithm Development in Wireless Communications", In Proc. Design, Automation and Test in Europe DATE'01, Munich, 13-16 March, 2001.

[6] T.Kailath, A.H.Sayed, B.Hassibi, Linear Estimation, Prentice Hall, 1999.

[7] G.J.Foschini, G.D.Golden, R.A.Valenzuela, P.W.Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element antennas," IEEE JSAC, vol. 17, no. 11, Nov. 1999.

[8] R.Van Nee, A.van Zelst, G.Awater, "Maximum Likelihood decoding in a space division multiplexing system," VTC Japan, May 15-18, 2000.

[9] G. Awater, A. van Zelst and R. van Nee, "Reduced Complexity Space Division Multiplexing Receivers," VTC Japan, May 15-18, 2000.

[10] J.G. Proakis, Dimitris G. Manolakis, Introduction to Digital Signal Processing. Macmillan, 1988.

[11] M.Rupp, S.Das, M.Guillaud, "On MIMO Decoding Algorithms for UMTS", 35^{th} Asilomar Conference, Monterey, California, Nov. 2001.

[12] M.Rupp, "A 64-point FFT Design Example Using A|RT-Designer," 34^{th} Asilomar Conference, Monterey, California, Oct. 2000.

[13] A.Burg, M.Rupp, M.Guillaud, E.Beck, D.Perels, N.Felber, W.Fichtner, "FPGA Implementation of a MIMO Receiver Front-end for UMTS", submitted to the IZS-Conference 2002, Zurich, Switzerland