PROTOTYPING FOR MIMO SYSTEMS - AN OVERVIEW

T. Kaiser[1], *A. Wilzeck*[1], *M. Berentsen*[1], *M. Rupp*[2]

1: Duisburg-Essen University
Faculty of Engineering
Department of Communication Systems
Bismarckstraße 81, 47048 Duisburg, Germany

2: Vienna University of Technology
Institute for Communications and RF Engineering
Gusshausstr. 25/389, 1040 Wien, Austria

ABSTRACT

The past decade has shown distinct advances in the theory of MIMO techniques for wireless communication systems. Now, the time has come to demonstrate this progress in terms of applications, where the intermediate step towards a customized product consists in more or less rapid "prototyping". Due to the multitude of different algorithmic approaches, i.e. beamforming, space-time coding, spatial multiplexing, and of different standards, i.e. UMTS, WLAN's, an ideal prototyping platform requires a high degree of flexibility and modularity in order to be qualified for a wide range of potential applications. The aim of this contribution is to give an overview about the challenges, the rich variety and the usefulness of such MIMO hardware platforms.

1. INTRODUCTION

Without any doubt MIMO techniques count to the few emerging key technologies for wireless communications. The number of yearly publications in this area is tremendous and meanwhile it is almost impossible to keep an overview about the numerous proposed algorithms. Moreover, the number of research teams worldwide is astonishingly high. For example, only in Europe more than 100 scientific and industrial organisations are performing serious research in multiple antenna techniques. Interestingly, although multiple antenna techniques are so auspicious, no relevant breakthrough in terms of highly successful MIMO products has been observed yet. Intuitive simple reasons are high MIMO complexity as well as critical cost issues when trying to push the promising MIMO theory towards practice. Taking both reasons into account requires that algorithm testing has to be performed on a very flexible and modular MIMO hardware platform where significant insights with respect to cost issues can be obtained in parallel. This goal can be briefly denoted by *rapid MIMO prototyping* and it is the aim of this contribution to shed some light on this topic and related challenges.

In Section 2 general considerations about MIMO platforms will be given in a consecutive way. Section 3 highlights a universal rapid prototyping methodology, needed to achieve quick and meaningful results and required to avoid common pitfalls. The final section covers a typical design example and discusses the essential hardware components from a more fundamental point of view.

2. GENERAL CONSIDERATIONS ABOUT MIMO PLATFORMS

MIMO platforms for current and future wireless communications can characterized by a large number of different properties like

- environment (indoor/outdoor)
- wireless standard
- carrier frequency
- bandwidth
- sampling frequency
- measurement device, demonstrator, prototype
- real-time, off-line
- number of antennas
- with or without feedback
- single or multiple users
- type of modulation
- basic algorithms
- achievable data rate.

It is obvious that with a single platform, even in case of highest flexibility and modularity, not all of the listed items can be served satisfactory. Hence, in order to figure out the relevant properties of such flexible, scalable and modular platform, some a-priori thoughts are given in the following.

An easy decision can be made regarding the wireless environment of the experiments: while indoor systems can be even for a small organisation or a University tested in reality, significant investigations in outdoor scenarios require a more expensive *mobile* equipment. Based on this first distinction, the classes of corresponding available wireless standards is approximately halved. Although not obligatory that experimental research follows wireless standards, we here have in mind only those investigations with strong impact to real world applications, so that *standards*, or at least *regularisations* become meaningful. In the following we will primarily focus on indoor scenarios; initial indoor MIMO experiments are reported in [1, 2] not utilizing any specific modulation scheme (from BPSK to 64-QAM, all were used) or access scheme. The experiments were based on a Pentek platform with multi DSP C40 processor, and required to be programmed in assembler. Supporting a transmission rate of less than a 1Mbps and up to 16 antennas such effort and complexity was tremendous at the time but it did not offer a concept for growing complexity. A platform capable of outdoor transmission was reported in [3, 4, 5] supporting a 4×4 antenna system in UMTS standards allowing for 8Mbps downlink capacity.

Once the environment is fixed and the potential standards are determined, the available range of carrier frequencies is pretty limited which further facilitates the decision for the antennas and analog front-ends of the targeted platform. In fact, because of antenna spacing in the order less or equal than $\lambda_c/2 = c_0/f_c$, a higher carrier frequency f_c is principally favourable so as to keep the size of the whole antenna array limited for applications. In contrast, the higher the carrier frequency, the stronger the attenuation by objects (e.g. walls). For example, a carrier frequency of, say, 10 GHz is considered to be blocked by walls. So, reasonable carrier frequencies are those given by the ISM band 2.4 GHz and 5.2-5.7 GHz, leading to array sizes up to a few tens of centimetres. These frequencies correspond to the family of wireless local area network standards IEEE 802.11 [6]. However, some extended platform flexibility can be achieved by use of analog standard connectors (e.g. SMA), so that different analog front-ends can be easily connected with the baseband hardware. This is possible by usage of standard 50 Ω terminated analog inputs and outputs for clocks, trigger signals, A/D and D/A interfaces.

An important question not only for baseband processing but also for the design of the RF front-end is the required bandwidth.

This work has been funded by the Christian Doppler Pilot Laboratory for Design Methodology of Signal Processing Algorithms.

The IEEE 802.11a standard requires a bandwidth up to the order of 20 MHz per channel. Because of possible channel merging, an even larger bandwidth may be required. Moreover, oversampling, beneficial to synchronisation and equalization can increase the internal baseband-bandwidth substantially. Due to IQ imbalance, it is advisable to operate analog-digital-converters at low-IF rather than at baseband [5, 7]. State-of-the-art in sampling are a few hundred Mega-samples, which indicates the limits and opportunities of such platforms.

The last relevant decision concerns the kind of the platform, i.e. either a measurement device, a demonstrator or even a prototype [7]. While a measurement device and a demonstrator are usually working off-line and are the right tools for pure algorithm researchers, a prototype requires real-time operation. Based on our philosophy to contribute to the progress of MIMO technologies towards real world applications, a prototype supports real-time experiments (model independent) and reduces the risk of the anticipated future product, especially in case of new technologies like MIMO. However, a prototype can also work in off-line mode [8], e.g. to test software routines or to provide the algorithmic researcher with helpful real world data. In that case, the prototype requires a huge memory in the order of a few gigabytes, further increasing the platform flexibility.

The remaining hardware related aspects are more or less antenna-size and thus cost-related and less related to a basic MIMO strategy. The number of antennas, except for measurement systems (e.g. channel sounders), is typically limited - starting at two transmit and one receive antenna-systems, very typically are 4×4, and going up to 16×16 antenna systems (see bwrc.eecs.berkeley.edu/Research/MCMA/home.htm). A general decision has to be made between unidirectional or bidirectional data traffic. The targeted platform (see also [9]) will support both data directions, having in mind the additional benefit of a "feedback channel" in order to provide the transmitter with full or partial channel state information. Hence, the transmitter is able to adapt to the given channel so that the class of testable algorithms is extended further. While in the past such feedback and also synchronization needs have been often realized by cable, a fully implemented wireless back channel is a much more solid approach also supporting outdoor experiments.

Of particular relevance for meaningful applications of MIMO technology is the behaviour of algorithms in a multi-user scenario. Even if this costs additional hardware for each added user, it is of great importance because multi-antenna techniques enable space division multiple access, which is one of the most relevant MIMO benefits [10].

Up to here all mentioned aspects are related to hardware issues and therefore completely determine the technical specifications of the targeted platform. In contrast, the software in terms of signal processing algorithms represent the inner nucleus of MIMO technology. For that reason the platform should be able to handle all relevant classes of algorithm, like spatial multiplexing, beamforming and space-time coding and any combination among them. This means sufficient signal processing power provided by DSPs and FPGAs (see the sections further ahead for more details) and also an adjustable antenna spacing in order to obtain correlated or uncorrelated or partial correlated data, as well as to deal with coupling issues. In addition, the type of modulation (e.g. OFDM) and the access scheme should be selectable as well.

The most relevant reasons for the use of MIMO techniques are threefold:

- to increase the maximum data rate,
- to extend the coverage,
- and to serve a larger number of users.

Of course, all discussed items impact the success. For example, the bandwidth together with the feedback limit the data rate, the number of antennas the coverage as well as the maximum number of potential users, etc. Hence, the targeted platform reflects the challenges of MIMO technologies.

Before we proceed with the methodology of rapid prototyping, an overview about existing MIMO hardware platforms will be given. The following interesting results were found by an internal study at the University of Duisburg-Essen in close cooperation with IMEC, Belgium.

- 60%, 19%, 8%, 13% of existing MIMO platforms are owned by Universities, Research Centers, Large Companies, Small and Medium Sized Companies, respectively.
- 60% (40%) of the platforms are designed for indoor (outdoor) environments
- 30%, 23%, 19%, 15% are DSP, DSP& FPGA, FPGA, Chip-based platforms, respectively.
- Most of the analog front-ends are build by discrete electronic components.
- Most of the platforms are focussed on OFDM.
- Most of the platforms are designed for $2-3$ GHz carrier frequency.
- Number of antennas typically vary between 2 and 4 (at both sides).
- Universities tend to favour the measurement device/demonstrator approach.
- Universities tend to favour the off-line approach by use of a large memory.
- Multi-user capable platforms are very seldom.
- Channel state information at the transmitter is very seldom.

Based on these observations, it becomes clear that the targeted hardware platform is not often realized yet. Especially, the multi-user scenario and the feedback channel are to the best knowledge of the authors not simultaneously investigated yet.

3. RAPID PROTOTYPING METHODOLOGY

3.1 Commercially Available Tools

While many rapid prototyping platforms exist, most only supporting either DSPs (favoured are TI's C6x) or FPGAs (favoured are Xilinx Virtex-II), some supporting mixtures of DSPs and FPGAs, a general method for mapping the signal processing algorithms onto such platform is not available. Some platform providers offer simple methods though supporting more general tools for specific DSPs and FPGAs.

TI (http://www.ti.com) offers a rich design environment that can be customized to specific DSP evaluation boards, including also real-time operating systems. Due to the excellent C-compiler and optimizer, signal processing procedures can be specified in a high level language and mapped automatically onto a DSP. Many prototyping platform providers take advantage of this environment and customize their platforms accordingly. Mathworks (http://www.mathworks.com) offers Simulink, a graphical interface with a rich library of toolboxes, easing the design of communication algorithms and fixed-point design tools. Most interesting is the real-time workshop supporting automatic mapping of Simulink designs into C code for TIs C6x processors. However, so far the efficiency of the coding is quite limited. Harsh conditions on real-time as they are common in wireless designs are not supported. Nevertheless, due to relative inexpensive evaluation-boards many researchers were able to quickly adapt their efforts to such boards, extending them with high speed ADC and DAC and supporting software defined radio development [11]. Even a 2×2 wireless MIMO system has been reported on [12] allowing to test simple Alamouti Space-Time Codes.

FPGA providers like XILINX (http://www.xilinx.com) and Altera (http://www.altera.com) enrich Simulink designs by their own libraries, called Xilinx System Generator and Altera's DSP Builder, respectively. They allow to simulate predefined DSP functions under Simulink and map the designs direct unto the

corresponding FPGA chips. These tools lead to quite efficient FPGA designs, provided the DSP functions are available in the libraries. Especially, data-flow oriented designs can be performed quickly, utilizing such tools. In [13] a FPGA board (BEE) with flexible communication links between the FPGAs is reported.

Nallatech (http://www.nallatech.com) supports XILINX FPGAs Spartan and Virtex-II based development boards together with Simulink and Xilinx System Generator. The boards also carry fast AD and DA converters. Similarly Aptix (http://www.aptix.com) offers a scalable FPGA board (System Explorer) based on Aptix' Field Programmable InterConnect technology (FPIC) together with some design tool chain. However Aptix focus more on so-called Pre-Silicon Prototyping, thus closer to a final product with less flexibility in the sense of a real-time demonstrator.

Lyrtech (http://www.lyrtech.com) offers multi-DSP system boards containing one or two DSPs either TI C6713 or ADSP21160 and a Virtex-II FPGA from XILINX. With help of TI and Simulink tools the chips, can be programmed supporting even hardware-in-the-loop co-simulation.

Companies like Sundance (http://www.sundance.com), Hunt Engineering (http://www.hunteng.co.uk), and Pentek (http://www.pentek.com) offer the richest set of DSP and FPGA modules supporting also fast DA and AD converters as well as even IF conversion modules. Typically, a software library supports the intercommunication between chips and boards.

3.2 Requirements

In wireless MIMO prototyping the requirements for system design are of technical and of methodological nature and due to the large complex designs quite challenging: a large complexity of signal processing algorithms needs to be supported typically requiring to partition algorithms over several DSPs and FPGAs. Such partitioning is not supported by commercially available tools and thus requires engineering intuition and a lot of hand-optimization. It also requires the presence of interface drivers to support intercommunication between DSPs, FPGAs and mixed DSP-FPGA modes. In order to develop and test such high-complex systems rapidly, hardware-in-the-loop co-simulation of parts of the design is a very useful method. Also, observing internal data in DSPs and FPGAs while the chips are running can be very useful in finding bugs.

Next to technical requirements of the tools involved, there is also requirements on the design flow methodology itself. Since rapid prototyping in wireless MIMO is very challenging, a whole design team is required to cover the design flow from system level design, over partitioning to finally mapping the various signal processing functions onto a HW platform. Such design team needs to consist of diverse experts in DSP and FPGA programming and requires a clear organization of the various tasks. In short, the design effort is very similar to an industrial chip or system design just with fewer people an some constraints less.

3.3 The Five-Ones Approach

In order to support the technical as well as the methodological requirements of a design team, a design flow methodology called the Five-Ones approach has been proposed [14]. The Five-Ones approach considers the following points as crucial for a successful prototyping effort in a rapid manner:

1. **one** design environment
2. **one** automatic documentation by specification
3. **one** forward-backward compatible code revision tool
4. **one** code to be worked on by refinement steps
5. **one** team to improve communication.

1. In [14] it is shown that the design effort appears as a feed-forward system, handling the design process from one part of the design team to the next and a feedback loop concerning communication about the common design goals and progress achieved. Such feedback slows down the design process. Since communication is required, the feedback loops cannot be broken. Also the required skill sets are not present in all parts of the teams. However, the reaction time can be changed dramatically, once all design teams share **one design environment**. This observation on its own is not new and many tools in the EDA community exist (Simulink, COSSAP/CoCentric System Studio and SPW to name the most widespread). However, since they have been developed to support specifically chip design (and to some extent algorithmic design) the architectural level and its exploration, as well as testing and system integration on specific hardware platforms (so called platform based design) are not supported. Also, due to specific language constraints many researchers refuse to use such systems, since they believe their productivity is dramatically reduced by them. On top of that, high license costs prevent many companies from using them throughout the whole design chain.
2. A second aspect is the missing documentation of the research part of the design team. While such people focus on meaningful results of their simulations, the following design teams are mostly interested in functional specifications. Graphical systems like COSSAP/CoCentric System Studio, SPW and Simulink offer the possibility to define functional blocks with clearly defined input and output ports and corresponding data-rates. Using such a graphical system **induces a documentation** while specifying the functional blocks. Specification allows detection of flaws at an early stage and, much more importantly, it can be used to supply additional information into functional descriptions of algorithms. The graphical description has further advantages: it avoids global variables. Global variables can lead to the undesired effect that information from the transmitter and/or channel is known at the receiver and some (undesired) cheating can be the result. It is, for example, quite common in literature to present MMSE receivers having perfect knowledge of the Signal-to-Noise Ratio (SNR), while this value in reality needs to be estimated. The problem of estimating such a value is typically underestimated. In addition to the graphical specification, COSSAP for example, allows writing so-called *Generic-C*, an ANSI C program enriched by essentially a header specifying the names, types and rates of the input and output variables, a feature well preserved in the so-called PRIM models of the new version CoCentric System Studio.
3. A third aspect of the slowdown in the product flow is the required permanent re-coding. Although the research part of the design team defines a code for simulation, the system design team is not able to reuse the code, mainly due to its poor documentation and coding style used. Based on the anticipated hardware platform, other languages (assembler, VHDL) have to be used at a certain level of refinement, requiring time-consuming hand re-coding. Such foregoing is error prone and requires a solution based on automatic re-coding tools. See for example [15] where it is claimed that errors found late in the design process cost up to 100 times more than those found early. While graphical tools do not provide an immediate solution to the tedious recoding process, they can support it by allowing multiple, but different descriptions for each block. This alleviates the transformation and allows doing it piece by piece. Simulink, COSSAP/CoCentric System Studio (http://www.synopsys.com) and SPW (http://www.cadence.com) allow for such code versions, but only on a block level, i.e., if a modification impacts several blocks at the same time, the system does not work out the required consistency.

   Revision control tools (such as CVS (Concurrent Versions System - http://www.cvshome.org) or ClearCase (http://www.rational.com/products/clearcase/index.jsp)) can help here. At certain times in the design flow, the entire design

becomes frozen and can be re-instantiated at a later time allowing to track a bug that shows up at a certain stage in the design flow, but was not noticed before. Further aspects of revision level tools are personal responsibility: the blocks can be assigned to specific people in the team and cannot be altered by others. Using such blocks, it is guaranteed that everybody in the team is working in the same, rather than in a personal environment. In order to guarantee backward compatibility it is important to stick with one code and the same language for as long as possible.

4. Furthermore, graphical systems allow an easy method of **code refinement** by co-simulation. The code can be refined from one revision level to the next and by instantiating the two versions at the same time their output can be compared while they are fed by the same input. An important step in code refinement is the switch from float to fix-point code. The recent *SystemC* initiative [16] (http://www.systemc.org) supports this step by extending ANSI C with fix-point data types. A|RT-Library from Adelante Technologies (now owned by ARM) offered such C++ Library extension for many years. The underlying idea is that by providing more and more details in a code at a very high level, the automatic tools modify the code iteratively into the required (meta-) descriptions until the final product is defined in every (technical) detail. These final descriptions express the code in a desired form, i.e., assembler code to program a DSP chip, VHDL or VERILOG code to program an FPGA or synthesize the required masks for an ASIC. However, since the automatic tools map the code from a high level to each of the lower levels, the refinement of the code is only performed on the high level description, i.e., the C-code. By iteratively rewriting the original C-code used in simulation to suit the needs of a specific hardware platform, the code remains backward compatible at any state and thus allows all teams to share the code and investigate problems. Specifically, there is no need to switch design environments when transferring from one team to another. While commercial tools rarely support hardware-in-the-loop this becomes an important feature to check proper functionality and should be a requirement for future prototyping platforms. In [14] a so-called real-sync method is reported on which allows to map DSP algorithms automatically from Simulink to TI C6X DSPs and run them there while the rest of the environment still runs under Simulink.
5. One last aspect when analyzing the slow development process is the team size. Poor communication is a drawback of rather large teams. Fortunately, the required amount of people in a prototype team is much smaller and it is possible to keep all team members as **one team** supporting full information to everybody. This is clearly a particularity in rapid prototyping that cannot easily be realized in a large product design team.

Recent development of efficient chip design proposes to use so called *virtual prototoypes* [17] before building HW. In this process all HW aspects including busses and interfaces are modelled in SW first. The advantages of such design process are that it supports refinement steps and the one-code paradigm and thus consistency of the design avoiding errors. A further advantage in complex chip designs is that the SW development can start before the HW is build. Most of the refinement steps can be performed automatically by translational tools further reducing error-prone manual processes and increasing efficiency [18]. With the growing complexity of MIMO algorithms, prototyping becomes more and more challenging. Thus, new concepts like virtual prototyping as an intermediate step can be very useful.

4. DESIGN EXAMPLE

As a typical design example, we choose a MIMO transmission utilizing Orthogonal Frequency Division Multiplex (OFDM) technique - such a system will be shortly referred to as *MIMO OFDM system*. We focus on OFDM, because it will be probably one of the key technologies for the physical layer of 4G wireless communication standards. Today's communication standards, such as Hiperlan/2, IEEE 802.11a/g, IEEE 802.16 and DVB, already use OFDM for digital modulation. Even non-standardized wireless products tend to use OFDM (http://www.airgonetworks.com).

4.1 MIMO OFDM

The basic idea of OFDM is to divide the carrier into multiple orthogonal narrowband sub-carriers, so that the assumption of a frequency flat fading channel valid and equalization becomes very simple. In an OFDM system multipath fading in a wide band channel does not produce self-interference, intra-cell-interference and inter-symbol-interference. In addition a cyclic prefix is used to combat the effect of delay spread. Because the Inverse Fast Fourier Transform (IFFT) and the Fast Fourier Transform (FFT) can be used very efficiently for OFDM modulation and OFDM demodulation respectively, the complexity of an OFDM based system is lower than for one using DS-CDMA. Main advantages of an OFDM-system is that only a low-complex, memoryless equalizer (only adjusting phase and magnitude of each subcarrier) is needed while a DS-CDMA-based system has demanding numerical load for equalization (equalization of time, phase and magnitude), because of dealing with multiple delayed paths due to the channel's delay spread.

Typical MIMO techniques like Spatial Multiplexing or Space-Time Coding, can be easily applied to the physical layer of OFDM-based communication standards. In a straightforward way a MIMO OFDM transmitter can be realised by multiple copies of a SISO OFDM transmitter and some kind of multiplexing or encoding before a transmit multiplexer. In Figure 1 such an approach is illustrated, following the SISO physical layer in the IEEE 802.11a specification [6]. Coding in form of Space-Time Codes, as well as among groups of transmitter branches can be deployed, see e.g. [19]. Note that Figure 1 points out only the important base-band components, in order to simplify the block diagram.

Starting with a coded bit stream (e.g. bit stream after applying FEC-coding and interleaving), the bit stream is mapped to either BPSK modulation or Gray-coded QPSK, Gray-coded 16-QAM, Gray-coded 64-QAM modulation. Space-Time Coding can be realized by a suitable encoder between the constellation mapper and the succeeding multiple transmitter branches. Since each branch is identically constructed, the following discusses only a single transmitter branch.

A serial-to-parallel conversion in conjunction with a second mapper are used to map the modulated symbols to 48 inputs of a 64-point Inverse Fast Fourier Transform (IFFT), which is used in order to modulate the symbols to the different sub-carriers. In addition, pilot-sequences are inserted to four of the sub-carriers inputs. The applied mapping scheme follows [6]. A guard interval (GI) - also called cyclic prefix - is added by a modified parallel-to-serial conversion, because GI addition is equivalent to copying the last 16 sample outputs of the 64-point IFFT as prefix. After this operation the whole OFDM symbol is constructed. After pulse-shaping, the signal is up-converted to the radio frequency and then transmitted by the corresponding antenna.

The MIMO OFDM receiver can also be split into parallel SISO receivers. Figure 2 shows a straightforward scheme. Concentrating again only to one of the receiver branches, the signal received from the corresponding RF components is converted from serial to parallel. It follows a synchronization unit (Sync) to synchronize the timing and to correct the frequency offset of the signal. In addition, the optimum FFT window has to be determined in order to remove the guard interval (GI). A 64-point FFT demodulates the OFDM symbol.

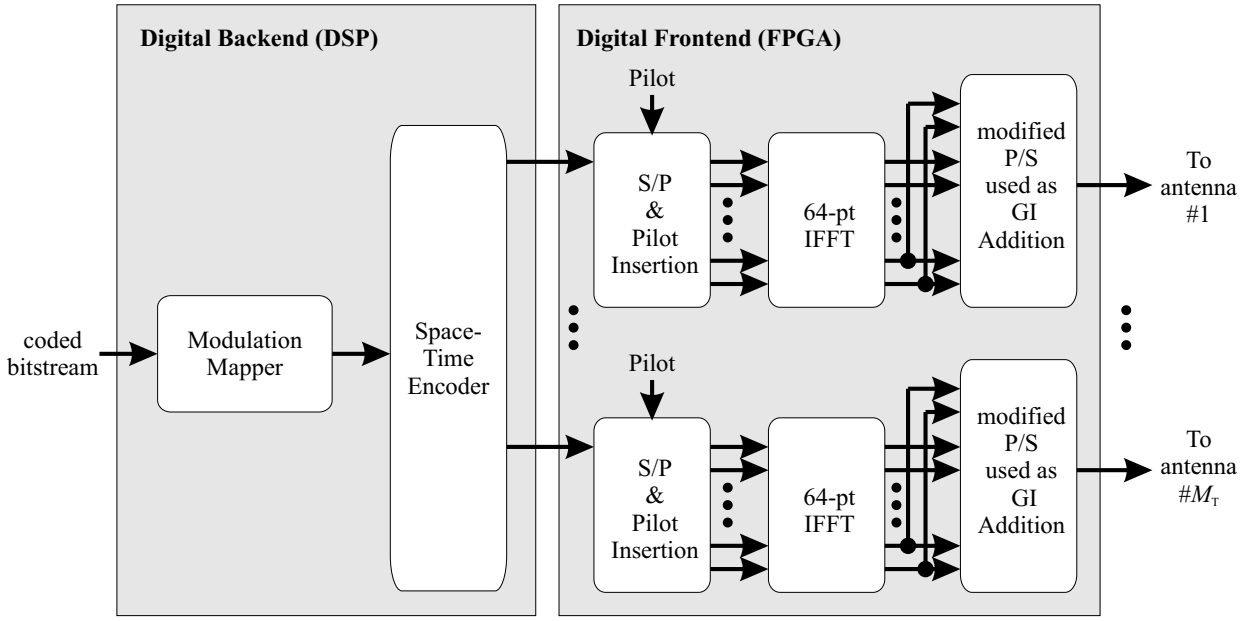

Figure 1: MIMO-OFDM Transmitter using Space-Time-Coding

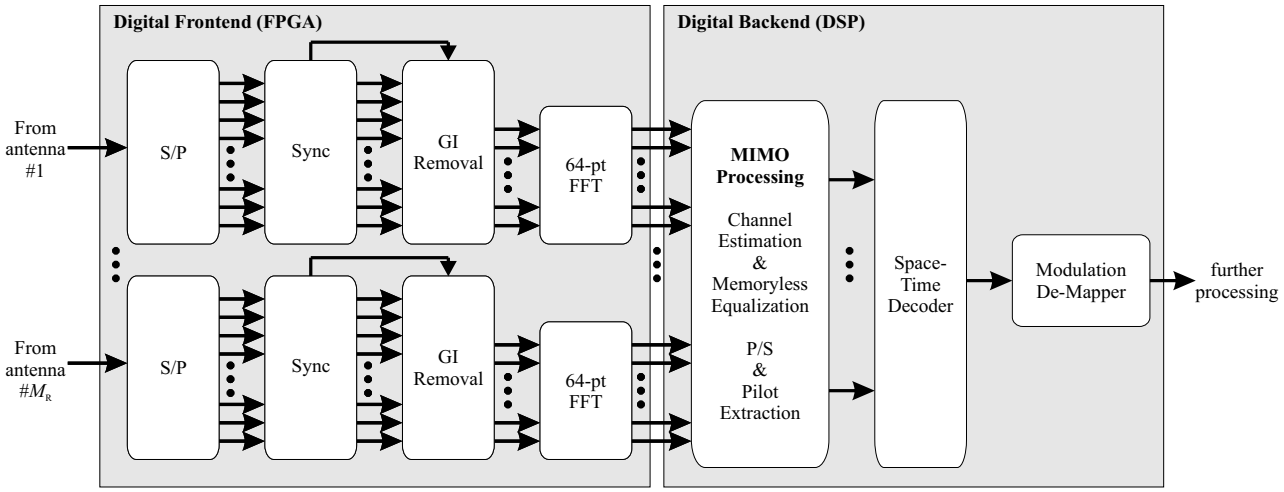

Figure 2: MIMO-OFDM Receiver using Space-Time-Decoding

All receiver branches are connected to a MIMO processing block, which is used to perform the MIMO channel estimation and memoryless equalization (phase, magnitude) to recover the transmitted data-streams of each antenna. Afterwards, a space-time decoder and a modulation de-mapper are used to recover the coded serial bit-stream, which is decoded later on by further processing.

4.2 Aptitude of digital signal processing devices: DSP

Before taking a look to the properties of modern digital signal processors (DSP), it is reasonable to briefly review two basic computer architectures both defined in the 1940's (see [24], [25] and [26]).
The famous **Von-Neumann-Architecture** represents the commonly used architecture of almost all computer systems. The architecture consists of one CPU (Central Processing Unit) core and one memory, which hold both program code and data. CPU and memory are connected by an address bus for choosing the memory address and a data bus to transfer the data. Because data and program code share one memory, the CPU needs at least two memory accesses - One to fetch the next instruction of the program code and afterwards for example one to fetch data from the memory.
In contrast, the **Harvard-Architecture** allows simultaneous access to program code and data, because of two independent memories - one for program instructions, one for data. A big drawback is, that one will need now, two independent address- and data-buses, which results in a doubled number of connection lines. Apparently, the Havard-Architecture has higher complexity making it more expensive. Nevertheless the Havard-Architecture with further improvements (e.g. introducing cache memories and dual-data fetch) is the commonly used architecture for DSPs, mainly because of the decoupled memory accesses and therefore doubled memory access bandwidth.
This leads to further differences of digital signal processors (DSP) to general purpose processors (GPP) as commonly used in PCs: They are optimized for operations in digital signal processing, i.e., a DSP offers special hardware and instruction sets supporting signal processing operations.

The most common task in digital signal processing is filtering. Let's assume an FIR-Filter with N coefficients. The output $y(n)$ of the filter is the convolution of the input signal $x(n)$ with the impulse response of the filter $h(n)$.

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{1}$$

As we can see from eq. (1) the basic operation is the multiplication of two operands ($h(.)$, $x(.)$) and adding the result to a so called accumulator, which represents after N iterations the value of $y(.)$. Two operations are involved, the multiplication and the addition. Normally, the overall operation is referred as MAC (Multiply-ACcumulate)-operation. Basically, this MAC-operation can be found in almost any algorithm dedicated to signal processing, including also the important discrete Fourier transformation.
For a DSP it necessary to execute such operations very efficiently. One way is to provide multiple MAC-Units directly in hardware, but one drawback is the increased complexity of the resulting DSP, which has basically to be also designed for low-power consumption and low-cost. Note also that for many MAC-operations, alternative formulations exist, for example CORDIC formulations for FFTs avoiding multiplications and multiplier-free implementations for FIR filters in communications [20]-[22].

Typical DSPs - for example TI's C6000 family or embedded processors like Starcore - are providing two and more independent data-paths in the DSP, each data path with a set of more or less specific Arithmetic-Logic-Units (ALU) performing logic (XOR, OR, AND, NEG) and/or arithmetic operations (+, -, shift) and one hardware multiplier (*) per path. Typical high-speed DSPs are operating in fixed-point arithmetic but there are also floating-point DSPs available. In order to store values (e.g. operands or results) each data path has it's own large register file (e.g. 32 or more registers (32bit)). This allows operating without penalty for cache or memory access. In addition one or more so-called cross paths are provided to exchange data between the two data paths.
The memory architecture is typical based on the Harvard-Architecture, but in extension further structured in three levels - sorted by access speed. At first level two small (1 to 16 KByte), but very fast 1st level caches are provided for caching program instruction or data for later re-use. In order to improve data operation both data path can read simultaneously and independently from the 1st level data cache and can normally fetch more than one operand per memory fetch. The second level consists of 2nd level caches which are build from slower, but even fast memory. Sometimes only one 2nd level cache memory is used to reduce the costs, so the memory is divided in address spaces for program code and data. Typical sizes for 2nd level cache are up to 1 Mbyte. 1st and 2nd level caches represent the internal memory of a DSP chip. The third level is the external memory normally applied as large SDRAM (128 MByte and more). Accesses to external memory and other external devices or interfaces are managed by an (E)DMA (Enhanced Direct Memory Access) controller. Because access to external devices or memory is slow, the DSP programmer always tries to reduce non-scheduled transfers to or from it during execution of an algorithm. If it is possible to schedule the transfers, for example an always incoming data-stream from an ADC, one can efficiently make use of the features of the E(DMA) controller, by establishing an (E)DMA channel. Using this channel the (E)DMA controller transfers data to the internal memory, so that the data becomes accessible without penalty. Only a latency for setting-up and establishing the channel is needed. Because using the external memory interface (EMIF) is the only way to get data in or out of the DSP, the bandwidth of this EMIF is the limiting factor for incoming or outgoing data-rate and for the usage in a MIMO prototyping system.

As an introductory example, let us first consider the SISO receiver case, assuming a signal bandwidth of a little bit less than 20 MHz (like used in IEEE 802.11a) so that a sampling rate of 40 MHz is needed (Nyquist-Rate). Also assume sampling with 16 bits. Thus, the incoming data rate is $40\,\text{MHz} \times 2\,\text{Byte} \equiv 80\,\text{MByte/s}$. This is the incoming data-rate for the SISO case without assuming any additional oversampling. In case of a MIMO receiver this data-rate needs to be multiplied by the number of receive antennas M_R so a four antenna system has an incoming data-rate of 320 MByte/s.

Such high-data rates and the need to process the data somehow will bring a DSP certainly to its limits. Let us consider a typical 32-bit EMIF with access to an SDRAM clocked with 100 MHz, so it is in theory able to transfer 400 Mbyte/s. In practice, this high data rate will not be achievable due to shared hardware and transfer blocking. Although this transfer can be handled by the (E)DMA controller, it will produce prohibitive load on the DSP, because the DSP needs to maintain the (E)DMA from time to time and perform operations on the data. In conclusion, a DSP only is appropriate when the data rate of in- and outgoing data is much smaller than the clock speed the DSP performs its operations.

But data-rate is not the only problem related to MIMO processing with a single DSP. Due to its large complexity parallelism is required and a MIMO receiver comprises of multiple data-paths, which must be processed somehow. A typical DSP will fail here, because it provides only limited parallelism of operation depending on the usage of its fixed-sized DSP core. For example, only two (up to four) multipliers are typically provided. Modern DSPs and compilers try to provide parallelism in software, so they try to process data serial, but fast enough so it seems to be parallel for the outer process. Techniques like software pipelining and SIMD (Single-Instruction-Multiple-Data) are most important here to reach this aim.

One of the biggest advantages of a DSP in rapid prototyping is the ability to use a common high-level programming language - namely C or C++. This makes a DSP easy-to-use and very flexible in programming. Another advantage is that it can handle irregular operations (e.g. protocol or controlling tasks), which are cumbersome to built in hardware.

However, the programmer needs some effort to program the DSP efficiently so as to meet the real-time condition. DSPs gain most of their performance by the maximum utilization of the functional units (ALUs, multipliers) and registers provided by the DSP core. DSP manufacturers offer *intrinsics* (highly optimized inline functions) for typical signal processing algorithms and tasks, facilitating the reduction of development time of applications. In addition, the compiler tries to achieve the maximum utilization by optimizing code and scheduling of loops for software pipelining. On the other hand software pipelining restricts the programmer to linear code, so the programmer should avoid branching.
Another limit of a DSP is the lack of **true** bit-level operations, because it is not possible to address bits directly[1]. The DSP programmer may therefore use masking and related tricks to make more efficient use of the DSP core resources - a resource efficiency which is highly dependent on the type of DSP.

In conclusion, due to high data-rates and other disadvantages, another signal processing device is needed in addition for regular and/or true parallel operations, which can be implemented in fixed specialised hardware structures. This processing devices should act as pre-processing units, thus reducing data-rate and performing regular operations - like filtering, FFT, IFFT - in multiple parallel data paths. Also, the additional usage of hardware accelerators (hardware for heavy computational tasks, e.g. Viterbi decoder) can be advantageous.

4.3 Aptitude of digital signal processing devices: FPGA

For any development of high data rate systems, most often digital front-ends, a high demand on numerical calculations in a design specific environment is given. Field Programmable Gate Arrays (FPGA) seem to be very suitable for this. A good overview of FPGA techniques can be found in [23]. They use a two-dimensional array of logic cells, consisting of Look-up tables (LUT) and Flip-Flops. Some specialized functions like multipliers and memory, depending on FPGA family and manufacturer, are also available. These cells can be individually accessed by a user defined routing, where a reload of the programming code after power up is required (RAM based technology). Typical FPGAs in wireless development are manufactured by *XILINX*, for example the *Virtex-II* and ALTERA.

Because of the lower development level, typically a hardware description language is required. Most often, VHDL or VERILOG is used. As discussed in 3.2 graphical descriptions like Simulink are very useful. Some tools offer direct conversion from graphical tools to FPGAs. These near to hardware description tools offer the possibility for highly flexible solutions in terms of speed and used gate array space economically, which perfect fits to the prototype system. However, due to the involved implementation complexity, FPGAs represent also a bigger source for errors and cause time consuming development, e.g. by manual porting and testing algorithms. On the other hand, these additional degrees of flexible design structures give the possibility for 'code' optimization into the designer's hand. The used external hardware, like bus systems and AD and DA converters, can be connected and be driven directly with their own protocols. There are no limiting pre-designs, like DSP bus systems. Also many parts of 'classic' digitally external hardware can be shifted into the FPGA code. Depending on the programming of the FPGA, the designed structures can run in parallel.

With view to modern wireless applications, a huge amount of Multiply Accumulate (MAC) calculations is required as well as bit level operations. Here, the use of an FPGA becomes strongly beneficial. Without a processor pre-design like bus width or interfaces, the designer can place the algorithms in form of dedicated hardware. Hence, the FPGA is not only *code optimized*, it can be *architecture optimized*. Note also that there is no difference between a bit level and a bus level operation, e.g. with an FPGA the required bit length can be implemented with optimal width. An additional benefit is that the design is compatible with future releases of the FPGAs when bigger and/or faster FPGAs enter the market. As already mentioned in a previous section, the development time VHDL or VERILOG code can be reduced by porting tools based on higher level description languages.

4.4 Proposed Mapping to FPGA/DSP

Taking all the mentioned properties into account, algorithms like S/P and P/S conversion, GI addition and removal, and the (I)FFT, are typically candidates for a solution in an FPGA on behalf of the DSP. The parallel antenna branches facilitate further the FPGA implementation of these front-end operations without any loss of flexibility. In order to gain some flexibility in MIMO algorithm choice, it is favourable to choose a fixed-point DSP as target platform for digital backend processing as shown in Figure 2. Furthermore, a floating point DSP can be added for sophisticated algorithms and controlling or protocol tasks as has been done in [5] for realizing the numerically demanding V-BLAST algorithm.

5. CONCLUSIONS

In this contribution an overview is given of prototyping for wireless MIMO systems. Embarking upon general considerations towards a flexible, scalable and modular MIMO platform for indoor scenarios, we proceed with a universal methodology and list the relevant issues - the five one's approach - in order to succeed in rapid MIMO prototyping. Then, we focussed on a particular design example, explained the challenges and proposed a basic strategy in order to adequately split the numerical load among DSPs and FPGAs. Regarding the analog front-end a standard RF-connection further increases the flexibility of the platform, so that either discretely implemented RF-front-ends, but also RF-chip-based evaluation boards can be easily deployed.

REFERENCES

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

[2] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture," Electron. Lett., vol. 35, pp. 14-16, Jan. 1999.

[3] M. Guillaud, A. Burg, M. Rupp, E. Beck, S. Das, "Rapid Prototyping Design of a 4×4 BLAST-over-UMTS receiver," 35. Asilomar conference, pp. 1256-1260, Nov. 2001.

[4] A. Burg, E. Beck, M. Rupp, D. Perels, N. Felber, W. Fichtner, "FPGA implementation of a MIMO receiver front-end for UMTS," Proc. International Zurich Seminar on Broadband Communications, pp. 8_1-8_6, Feb. 2002.

[5] A.Adjoudani, E.Beck, A.Burg, G.M.Djuknic, T.Gvoth, D.Haessig, S.Manji, M.Milbrodt, M.Rupp, D.Samardzija, A.Siegel, T.Sizer II, C.Tran, S.Walker, S.A.Wilkus, P.Wolniansky, "Prototype Experience for MIMO BLAST over Third Generation Wireless System," Special Issue JSAC on MIMO Systems, vol. 21, pp. 440-451, April 2003.

[1] Normally the smallest directly addressable data unit is byte (8 bits).

[6] IEEE Specification: ”High speed Physical Layer in the 5 GHz Band”, 802.11a, 1999.

[7] R.Morawski, Tho Le-Ngoc, O.Naeem, ”Wireless and wireline MIMO testbed Electrical and Computer Engineering,” IEEE CCECE, vol. 3, pp. 1913-1916, May 2003.

[8] P.Murphy, J.P.Frantz, E.Welsh, R.Hardy, T.Mohsenin, J.Cavallaro, ”VALID: custom ASIC verification and FPGA education platform,” Proc. of Microelectronic Systems Education, pp. 66-67, June 2003.

[9] A. Wilzeck, T. Kaiser, ”Towards a 4x4 MIMO testbed”, IEE DSPEnabled Radio Conference, Livingston(Scotland), Sept. 2003.

[10] S.Rajagopal, S.Bhashyam, J.R.Cavallaro, B.Aazhang, ”Real-time algorithms and architectures for multiuser channel estimation and detection in wireless base-station receivers,” IEEE Transactions on Wireless Communications, vol. 1, is. 3, pp. 468-479, July 2002.

[11] S.Abendroth, R.Heuze, S.Weiss, ”A Software defined radio front-end implementation,” IEE DSPEnabled Radio Conference, Livingston(Scotland), Sept. 2003.

[12] R.Gozali et. al.,”Virginia Tech Space-Time Advanced Radio (VT-STAR),” IEEE Radio and Wireless Conference, RAWCON 2001, pp. 227-231, Aug. 2001.

[13] C.Chang, B.Richards, B.Brodersen, ”DSP system design using the BEE hardware emulation environment,” Conf. Proc. Asilomar Conference, Nov. 2003.

[14] M. Rupp, A. Burg, E. Beck, “Rapid prototyping for wireless designs: the five-ones approach,” Signal Processing, vol. 83, Issue 7, pp. 1427-1444, July 2003.

[15] R.S. Janka, *Specification and Design Methodology,* Kluwer 2002.

[16] W. Mueller, J. Ruf, D. Hoffmann, J. Gerlach, T. Kropf, W. Rosenstiehl, “The Simulation Semantics of SystemC,” In Proc. Design Automation and Test in Europe, DATE’2001, Munich, pp. 64-70, March 2001.

[17] C. Valderrama, ”Virtual Prototyping For Modular And Flexible Hardware-Software Systems,”Design Automation for Embedded Systems Journal, vol. 2, no. 2, pp. 267-282, 1997.

[18] P. Belanović, M. Holzer, D. Mičušík, M. Rupp, ”Design Methodology of Signal Processing Algorithms in Wireless Systems,” International Conference on Computer, Communication and Control Technologies CCCT’03, pp. 288-291, July 2003.

[19] R.S. Blum, Y.G. Li, J.H.Winters, Q.Yan, ”Improved Space-Time-Coding for MIMO-OFDM Wireless Communications”’, IEEE Transactions on Communications, vol. 49, no. 11, pp. 1871-1878, Nov. 2001.

[20] M.Rupp, J.Balakrishnan, “Efficient chip design for pulse shaping”, SPAWC 99 in Anapolis, pp. 304-307, May 1999.

[21] H.L.Lou, M.Rupp, R.L.Urbanke, H.Viswanathan, R.Krishnamoorthy, “Efficient implementation of parallel decision feedback decoders for broadband applications”, IEEE Electronics, Circuits and Systems Conference, Cyprus, pp.1475-1478, Sept. 1999.

[22] M.Rupp, H.L.Lou, “On Efficient Multiplier-Free Implementation of Channel Estimation and Equalization”, Globecom 2000, San Francisco, pp. 6-10, Nov. 2000.

[23] Zoran Salcic, *Digital System Design and Prototyping,* Kluwer 2000.

[24] von Neumann, J., ”First draft of a report on the EDVAC”, Annals of the History of Computing, IEEE ,vol. 15 , is. 4 , pp. 27-75, 1993.

[25] William Aspray, ”John von Neumann and the Origins of Modern Computing”, MIT Press, ISBN: 0262011212, December 7, 1990.

[26] ”IBM’s ASCC a.k.a The Harvard Mark I”, http://www.ibm.com/ibm/history/exhibits/markI/markI_intro.html