Challenges in Building MIMO Testbeds

M. Rupp, S. Caban, C. Mehlführer

Vienna University of Technology
Institute of Communications and Radio-Frequency Engineering
Gusshausstrasse 25/389, A-1040 Vienna, Austria
http://www.nt.tuwien.ac.at/rapid-prototyping

ABSTRACT

This tutorial provides an overview on the state of the art in wireless testbeds and prototypes suitable for MIMO transmissions. We discuss the pros and cons of available tools on the market, report our experiences and present examples of recent MIMO HSDPA measurements with our Vienna MIMO testbed.

1. INTRODUCTION

In the past years, wireless testbeds and prototyping experienced much interest in academia and industry. In a hot market like telecommunications, reliable information about future designs and developments is very crucial to make the right decisions.

We distinguish between testbeds and prototypes [1, 2]. Testbeds support real-time transmissions over the air and thus allow for experimenting with true physical channels, including also analogue frontends. This makes the transmission process very realistic. On the other hand, rapid prototyping allows for sketching transmitter and receiver hardware architectures of future products. Thus, rapid prototyping is very close to the design of a final product, de-risking its financial investment. Rapid prototyping requires a lot more time than testbed measurements but can provide very detailed answers about technologies of potential future products while testbeds typically offer answers about new communication strategies. In [1] an overview of existing testbeds in academia and industry can be found.

This paper is organized as follows. In Section 2, an overview is provided on current commercial and academic tools for testbeds and prototypes. In Section 3, the newcomer is given a lot of valuable inside information on how to set up and use a testbed. Section 4 provides a case study for a recently made MIMO UMTS HSDPA measurement experiment. Section 5 closes the paper with some conclusions.

This work has been funded by the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms. See: http://www.nt.tuwien.ac.at/cdlab/

2. WIRELESS TESTBEDS AND PROTOTYPES

Since tool providing companies do not use a consistent terminology and try to offer as many features as possible, it is not easy to categorize available products into testbeds and prototypes. Most can be used for both purposes; however, some are suited better for one purpose than for the other. The selection here is thus somewhat subjective.

Complete out-of-the box testbeds are available from Lyrtech (www.lyrtech.com) and Signalion [3] (www.signalion.com). They provide 2x2 MIMO transmissions in burst mode at 2.4 GHz. They are very strongly coupled with MATLAB and can be run directly with a MATLAB interface. Thus, such testbeds are very suitable for students learning the first ropes of MIMO transmissions. Elaborate receiver designs, extensions to an arbitrary number of antennas, higher bandwidths, and real-time streaming experiments are typically not possible or only with high effort. A support for more antennas is planed for (Lyrtech offers a 4x4 RF frontend at 2.4/5.2/5.8 GHz) but complete out-of-the-box solutions were not available at time of writing.

At the other extreme, companies like Nallatech (www.nallatech.com) support flexible FPGA based boards with additional ADC and DAC boards to build prototyping setups for complex transmitter and receiver designs. The special software packages DIMEtalk and FUSE are offered to program the FPGAs and couple them as well as the I/O interfaces. DIMEtalk allows also for a C to VHDL conversion, supporting an automatic flow from a high level design language to FPGAs. Also a MATLAB toolbox is available allowing to tightly couple MATLAB development and FPGA design. Alternative approaches can be found in academia (see [4] for an overview) or at the Berkeley wireless research lab [5]. Recently appearing on the market are CatapultC from Mentor Graphics and a

Matlab to VHDL/C conversion tool from Catalytic (www.catalyticinc.com). Longer on the market is HandleC.

In between these two extremes, several companies are positioned offering relatively close ranges of products. Sundance (www.sundance.com) and Hunt-Engineering (www.hunteng.co.uk) offer carrier boards for PCs (USB or PCI interfaces) or stand alone carrier boards that support functional modules with DSPs, FPGAs, or I/O modules like ADCs and DACs. This modularity allows the designers for setting up their own platform, depending on their special needs. A third large provider is Pentek (www.pentek.com) offering complete boards with DSPs, FPGAs and I/O modules. This gives less modularity, however, the modules are often tightly coupled providing higher data throughput and more stable setups. All three providers support almost only TI DSPs of the C6x series, Xilinx FPGAs of the Virtex family and ADC/DAC modules up to 200 MSPS. They all offer special design and communication software to ease the multi-DSP/FPGA design (3L Diamond for Sundance, HEART for Hunt Engineering, TI code composer and Xilinx Foundation ISE for all of them). Also MATLAB tools are offered, in particular the Xilinx System Generator that allows for easy MATLAB/Simulink to FPGA conversion. Note, however, that only relatively simple and well structured (synchronous data flow) designs are supported by such systems. A complex UMTS receiver, for example, with highly irregular structure and a lot of control mechanisms cannot be build by them. Better tools allow for a high level language (C or SystemC) to FPGA conversion [4] but very few products are on the market due to their complex requirements.

Apart from the complete solutions by Signalion and Lyrtech, most other providers do not offer RF-frontends making it very hard to include the true physical channel. The development of RF frontends is costly and cumbersome and requires a completely different design expertise than in digital circuit design. Also the development equipment is rather expensive making such designs in low amounts of pieces relatively costly. In particular, available frontends are lacking flexibility in terms of supported carrier frequencies and bandwidths. A possible solution may come from IAF GmbH (www.iaf-bs.de) who provides FPGA and DSP modules simular to Sundance and Hunt Engineering but also offers RF frontends up to 2.7 GHz. Applying such a system, Siemens and the Heinrich-Hertz Institute of Berlin report a successful 3x5 transmission with more than 1 Gbps rate. Recently, ARC (www.smart-systems.at), Signalion, and Sundance are offering flexible front-end boards in the operating range from 2.4 GHz to 5.8 GHz with up to 40 MHz bandwidth. Signalion offers a board with four RF chains, while Sundance offers a scalable board with two RF chains that can be concatenated to build larger units. Such front ends are built by available chip sets (MAX2829 for Sundance and Lyrtech) that are used in WLAN and WiMAX products. Therefore, do not expect them to be extremely linear.

Since specific design tools are typically missing in commercial testbeds and prototyping equipment, EDA (Electronic Design Automation) tools used in the chip design process are also used for prototyping. For example, EDA tools are available from Synopsys and CoWare (and many smaller companies) but the licence costs are typically too high for prototyping. In particular, when heterogeneous systems including DSPs and FPGAs are required, no general supporting design tools are available. Very helpful in this context are Hardware-In-the-Loop techniques (HIL). They allow for programming a part of the entire simulation code directly on a DSP or FPGA and run it together with the simulation environment. Early academic developments for DSPs [6] and for FPGAs [7] were followed by first products from Sundance [8].

3. FROM THEORY TO PRACTICE - MIMO TESTBEDS

If a complete testbed for the desired investigations is not commercially available and an entirely new design is too cumbersome and time consuming, the natural solution is to buy as many off-the-shelf components as possible and make them working together. However, the required integration work should not be underestimated.

3.1 Digital Baseband Hardware

At first glance, finding and buying digital baseband hardware that seems to fit ones demand is easily achieved.

However, our experience over the years showed that it takes a considerable amount of time and manpower to get even the most basic demo programs to work. Reasons for this are:

- missing, misleading, or carelessly written and not updated **manuals**, pinout documentations, and sample programs,
- a lack in **support** or extremely long support-answer times,
- **compatibility** problems between different versions of hardware and software,
- and **stability** problems. Even a at first glance well running demo programme provided by the manufac-

turer may reveal astonishing and unforeseen instabilities if executed for a very long time period of time.

One has to be careful that for marketing reasons

- bandwidths and transfer **speeds** proclaimed by hardware manufacturers usually represent **pure marketing** numbers. Busses are often not able to transfer bus-clock times bus-width bits on a permanent basis.
- It also sometimes happens that **different features** proclaimed for a product cannot be achieved at the same time, but only on an **either-or basis**. For example, a synchronous transfer is only possibly at a lower speed, the proclaimed higher speed only works in asynchronous mode.

Another problem that arises with digital baseband hardware is that products are sometimes sold *not* including the possibility to flexibly reprogram them as advertised—and this fact is not obvious to the customer:

- Typically, **firmware** can be used in its delivered form but it **cannot be modified**. In order to modify it, additional licenses and tools are required, costing considerably additional amounts of money.
- The **software** included with the product may also be **limited** "on purpose" in its functionalities (e.g. just work on one FPGA but not on two). The unlock keys required imply additional expenditures often overseen at the date of purchase.
- And even if the software keys are not limited, one has to be careful that all the **documentation needed to modify** the firmware is included and does not have to be paid additionally. At such situations one may start to perform the programming and development work of what has been purchased for.

3.2 Tool and Component Selection

One also has to be careful that the *specific* software tools sold with a product require other *specific* software packages—in a *specific* version—to operate.

- It often happens that errors found are corrected in newer software versions, but this also implies that all the other specific **software packages have to be updated** in order to work correctly—a vicious circle that consumes a lot of time and money since usually other software packages from other vendors are also affected.
- It has been reported that companies "charge for maintenance," which means that one has to pay special attention whether charge-free bugfixes are included in the delivered software.
- In addition, highly specialized software **tools** used to program hardware are, unfortunately, often **not working reliable**.
- Due to poor documentation and the endless amount of possibilities why bugs could occur, tracking, reporting, and getting these errors fixed is usually an endless challenge. Therefore, one should consider only using extensively tested "**standard tools**", for example: TI Development Environment or Xilinx ISE, instead of spending money on tools that seem to save time at first glance, but afterwards turn out to be error prone.

Baseband hardware is often sold on a module basis in complete packages. The hardware develops so rapidly, that hardly any company has the time to extensively test its hardware and prepare well written manuals and source codes. The best way to deal with these problems is to buy from the company with the best support. A fast and competent support often makes the difference between achieving anticipated goals or not.

In multiple antenna architectures, special emphasis has to be put on "synchronous" operation:

- It must be possible to **synchronize the digital signal paths** used for different antennas, even if they are spread among several chips or modules. Memories, interpolators, filters, and digital mixers may not allow for this, especially if instead of FPGAs, dedicated hardware is used.
- Because they are just scaled SISO solutions, many offered MIMO solutions do not allow for a **synchronous radio frequency oscillator** for analog up/down mixing—even if advertised.
- One has to make sure that the word "synchronous" really means "equal in phase and frequency without any jitter" if used in product advertising brochures. Failure in synchronicity may result in dramatic performance loss compared to perfectly synchronous transmission.

3.3 Analog Front-Ends

When it comes to analog front-ends (e.g. analog upconverters, filters, amplifiers), things look opposite. It is on one side very difficult to buy the hardware needed, since very few products exist (most only in components and not complete). One has to buy everything on a per module basis sometimes even from different suppliers. On the other side, once the hardware is obtained, components can be rather quickly setup and made cooperating. Some very expensive measurement equipment is required though in order to check if the hardware is working within the desired specifications or not. There are usually no hidden costs afterwards, no software

tools needed, no carelessly written manuals and demo applications. On the downside, analog high frequency hardware is hardly ever flexible. Once one changes the center frequency, one has to rebuy most of the hardware — in the digital domain, this only requires the modification of some bits.

Therefore, when buying a new analog RF frontend, it is very important to choose the "proper" center frequency:

- It is a lot easier and cheaper to obtain hardware for free **ISM bands** (e.g. 2.4 GHz, or 5.2 GHz). Other bands (1.5 GHz, 2 GHz, 3.5 GHz, 5.8 GHz) are sufficiently close to draw the right conclusions for the experiments.
- Choosing a frequency within a free ISM band implies that other interferers like Bluetooth, WLAN, microwave ovens, cordless computer peripherals, etc. may inherently influence every single measurement. If this **unpredictable interference** is desired, a transmit frequency within an ISM band should be chosen.

3.4 Costs

As pointed out, setting up a testbed requires a considerable amount of money, manpower, and—most important—time, but in many cases, this may be still more economical than e.g. buying an extremely expensive but little flexible channel sounder.

A high quality channel sounder costs typically between 300 k€ and 1 M€, the hardware for a good testbed (100 k€) plus four person-years for setting it up may add up to 250 k€—still considerably cheaper. Furthermore, one can now perform more research than just extracting channel coefficients and is able to test transmissions over the air with the signals that will be applied in the final product.

The main downside of a testbed, however, is the time needed to set everything up and get it working[1]. This makes testbeds very suitable for basic research where time to market is usually not the primary directive, but often uninteresting for other purposes. Companies, on the other hand, are well recommended to continuously put effort into testbeds in order to constantly have them available and not to start from scratch every time a new product design cycle is started. Note also that *if* a testbed is set up and working, it may allow for carrying out measurements within minutes, especially when the data is processed off-line in tools like MATLAB. Clever consideration of similar experiments, that can utilize the testbed without timely hardware modifications, can allow to catch up the "lost" time easily.

[1] As a rule of thumb this time cannot be reduced to less than a year because of delivery times and unforeseeable problems.

3.5 From MATLAB Code to a Testbed

Once a MATLAB code works well in simulation, a testbed would clarify whether the algorithms are suitable for real over-the-air communications. Using MATLAB code with a testbed is not a simple process. There are many things that have to be taken into account. For the simple case of off-line processing in MATLAB these are e.g.:

- MATLAB simulations often operate **in the discrete baseband** only. Therefore, transmit and receive filters, interpolators, etc. have to be added.
- Interpolating to a fixed, given hardware sample-rate may also introduce impractical interpolation factors (e.g. 3.84 MHz (UMTS) to 100 MHz sampling) requiring interpolation filters with extreme length. Alternatively, decreasing and optimizing the filter complexity may result in a lengthy project on its own.
- MATLAB simulations often assume **perfectly synchronized signals**. In measurements, one now has the choice to:
  - synchronize transmitter and receiver perfectly (typically by cables)
  - nearly perfectly synchronize them using rubidium frequency normals and GPS receivers (which may be required if e.g. the receiver is mounted in a car)
  - use special training sequences prior to the transmitted data (which may only be possible in static scenarios)
  - implement proper synchronization algorithms.

  Even several of these options may be used together for all required synchronizations (e.g. local oscillator frequency, timing, block start,...).
  In some cases, perfect synchronization may be the method of choice to avoid all undesired effects (e.g. for reference purposes to test the performance loss of proper synchronization algorithms). Even the first famous MIMO experiments carried out were using cables for synchronizing transmitter rand receiver clocks [9]. In other cases, implementing proper synchronization algorithms in the receiver may deliver a better view of the reality. Unfortunately, this is not always possible, e.g. if only a limited number of blocks is available and synchronization requires averaging over long periods of time.
- **The channel is never known** to the receiver. Therefore, channel estimation cannot be omitted as it is often done in MATLAB simulations. In quasi-static scenarios, long training sequences can be used to

nearly perfectly estimate the channel (for reference purposes).
- **For many receiver algorithms the noise variance also has to be estimated** at the receiver. However, the simple trick to measure the noise variance in the absence of transmitting signals may often save coding time and provides accurate estimates regardless of the modulation scheme used.

Once working properly, a testbed (plus subsequent off-line processing of the received data in MATLAB) is a very powerful and quick method for evaluating algorithms using realistic over-the-air transmissions. One has the choice to measure the absolute performance of a transmission scheme or to compare two transmission schemes relative to each other. It is especially easy to measure the relative difference between two types of receivers because

- The same stored receive data can be evaluated, thus making the comparison fair.
- Debugging is also made easier, because the received data remains equal.
- The number of channel realizations can be significantly reduced since for measuring relative performance a much smaller number compared to absolute performance is sufficient.
- Systematic errors in relative performance measurements play a less dramatic role than in absolute performance measurements.

Note that we only report here the effort for a testbed using as much available tools as possible. Once a prototype is anticipated, a lot more effort is required. Converting a MATLAB code into a working VHDL code for an FPGA is a rather large step and is by far not fully supported with available tools even if many vendors may claim this.

4. A CASE STUDY: MIMO UMTS-HSDPA

In the following, an example measurement obtained with our Vienna MIMO testbed [10] is given. The aim of this measurement was to investigate the performance gain achieved when using multiple transmit and receive antennas in an HSDPA (High Speed Downlink Packet Access) system as they are currently being employed in Europe as single antenna solutions.

4.1 Measuring UMTS-HSDPA transmissions

The following parameters were chosen for the experiment:

- Off-line processing of data in MATLAB.
- A center frequency of 2.5 GHz.
- Transmitting at a constant total available transmit power (I_{or}) of 20 dBm. Therefore, dependent on the number of antennas currently in use, the transmit power per antenna is adjusted such that the total power remains constant. The pilot channel is transmitted with a constant total power of 10 dBm.
- Perfect synchronization of the sampling rates, the local radio-frequency oscillators, and the block start by cable.
- Four transmit antennas (patch antennas) mounted on the roof of a building (Figure 1).

Figure 1: Transmit amplifiers and antennas placed on the roof.

- Urban outdoor propagation channel (in this case the receiver is placed indoors).
- Four receive antennas located indoors in the same building five floors lower. The receive antenna array (quarter wavelength monopole antennas mounted on a common groundplane) was moved and rotated within an area of $4\times4\,\lambda$ by an xy-positioning table to generate 4000 channel realizations (Figure 2).

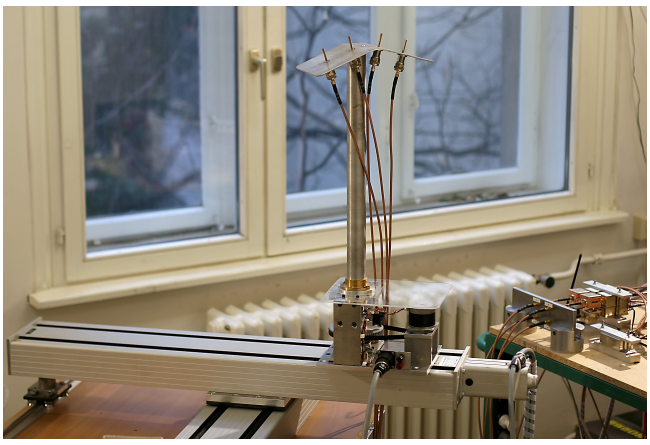

Figure 2: Receive antennas (moveable and rotateable) placed indoors.

- Transmit data generated according to Lucent's PARC (Per Antenna Rate Control) proposal for MIMO HSDPA [11].
- Channel estimation by correlating the received signal with the pilot sequences.
- MMSE equalization at the receiver [12].
- Coding and modulation according to the HSDPA standard. Specifically, we used 4-QAM modulation and turbo coding with rate-matching at a rate of 0.7.
- Ten spreading codes, each of length 16, assigned to the user.

Figure 3 shows an exemplary impulse response. The delay spread measured is about 5-7 chips requiring MMSE equalization at the receiver.

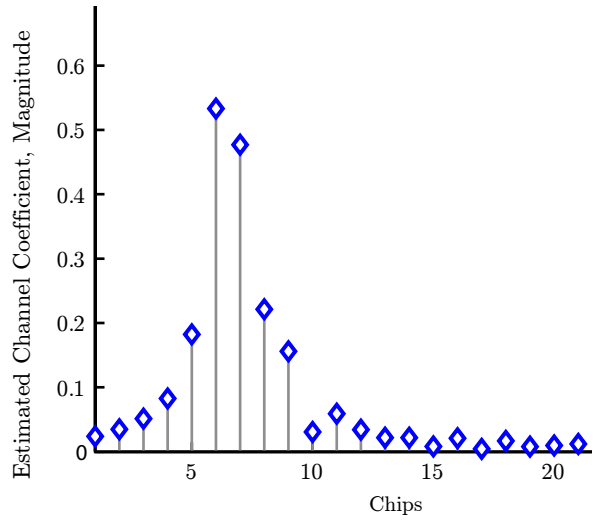

Figure 3: Magnitude of estimated impulse response, 1×1 system

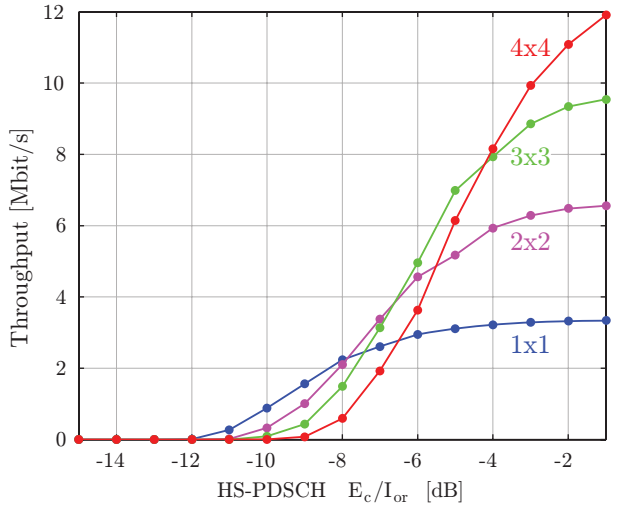

Figure 4: Measured throughputs averaged over channels for a 1×1, a 2×2, a 3×3, and a 4×4 system

Block error ratio (BLER) (Figure 4) and throughput (Figure 5) measurements have been carried out for different numbers of transmit and receive antennas. For an increasing number of receive antennas, we observe a significant gain in signal to noise ratio, diversity, and throughput. Note that for small E_c/I_{or} values, the MIMO systems with more transmit antennas show poor performance due to the simple channel estimation implemented.

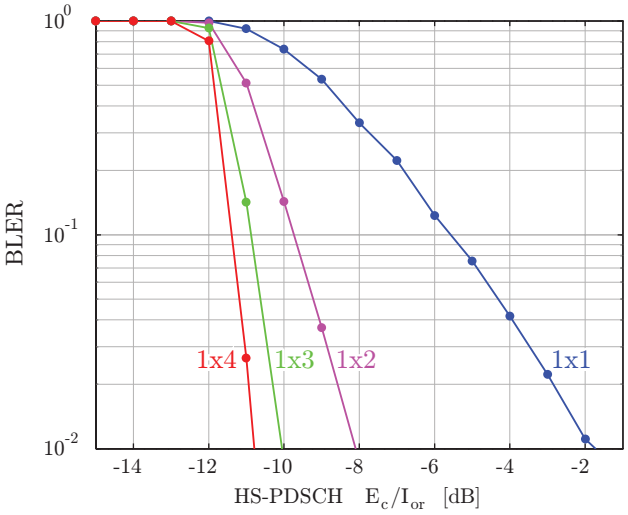

Figure 5: Block error ratios measured for a 1×1, a 1×2, a 1×3, and a 1×4 system

4.2 Time required for the HDSPA Experiment

Obtaining the curves in the previous experiment took about three hours for setting up the testbed, one hour for the measurement, and two hours for the evaluation[2]—just six hours in total.

While this is a relatively quick experiment, note that most time was spent in developing the testbed and in gaining testbed experience. This experiment time example did not include:

- Writing and testing the MATLAB code. This needs to be done for MATLAB-only-simulation anyway.
- Adapting the simulation code for the testbed. Due to our experience this can be done quickly now but was a time consuming process in the first experimental steps.
- Designing and preparing of a measurement experiment. Most time can be saved by carefully considering the experiment. One can reuse setups that have shown to work properly. One can also use well pre-

[2] The evaluation was carried out using a cluster of 20 standard personal computers.

pared and often used test sequences to ensure that the setup is correct.

- Initial time for developing and bringing a testbed into service. (four person-years, one year minimum!)

For comparison, a MATLAB-only-simulation would have taken about two hours, approximately the same time as was needed for the evaluation of the measured data. In MATLAB one has to select and simulate the channel model, but saves time on e.g. omitted synchronization, omitted interpolation, and omitted root raise cosine matched filtering.

5. CONCLUSIONS

Testbeds and the possibility of rapid prototyping are indispensable in wireless designs when high investment costs are necessary to develop a new wireless product. However, testbeds supporting arbitrary antenna numbers and operating frequencies are not available off-the shelf. They can only be assembled component-wise. This tutorial gives an overview of which components are currently on the market and what one has to consider before and after buying such equipment. In particular, the newcomer will find valuable information before starting a costly and frustrating adventure. A case study of successful MIMO UMTS HSDPA measurements closes the tutorial.

Acknowledgment

The measurements reported here could not have been performed without the prior design work of Robert Langwieser, Lukas W. Mayer, Klaus Doppelhammer, and Arpad L. Scholtz.

REFERENCES

[1] A. Burg and M. Rupp, *EURASIP Book on SMART Antennas, Part 5, Chapter 6: Demonstrators and Testbeds*. Hindawi, 2006.

[2] M. Rupp, C. Mehlführer, S. Caban, R. Langwieser, L. W. Mayer, and A. L. Scholtz, "Testbeds and rapid prototyping in wireless system design," *EURASIP Newsletter*, vol. 17, no. 3, pp. 32–50, Sep. 2006. [Online]. Available: http://publik.tuwien.ac.at/files/pub-et_11232.pdf

[3] M. Stege, T. Hentschel, M. Löhning, M. Windisch, and G. Fettweis, "Ieee 802.11n mimo-prototyping with dirty rf using the hardware-in-the-loop approach," in *Proc. of the 14th European Signal Processing Conference (EUSIPCO)*, Florence, Italy, Sep. 2006.

[4] E. Casseau, B. L. Gal, P. Bomel, C. Jego, S. Huet, and E. Martin, "C-based rapid prototyping for digital signal processing," in *Proc. of the 13th European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, Sep. 2005.

[5] C. Chang, J. Wawrzynek, and R. W. Brodersen, "BEE2: A high-end reconfigurable computing system," *IEEE Design & Test of Computers*, vol. 22, no. 2, pp. 114–125, Mar. 2005.

[6] B. Jones, S. Rajagopal, and J. Cavallaro, "Real-time dsp multiprocessor implementation for future wireless base-station receivers," in *TI DSPS Fest, Wireless Applications*, Aug. 2000.

[7] G. Brandmayr, G. Humer, and M. Rupp, "Automatic co-verification of FPGA designs in SIMULINK," in *Proc. MBD Conference 2005*, Munich, Germany, Jun. 2005.

[8] M. Ahmadian, Z. Nazari, N. Nakhaee, and Z. Kostic, "Model based design and SDR," in *Proc. of the 2nd IEE/EURASIP Conference on DSP enabled Radio*, Southampton, UK, Sep. 2005.

[9] P. Wolniansky, G. Foschini, G. Golden, and R. Valenzuela, "V-blast: an architecture for realizing very high data rates over the rich-scattering wireless channel," in *International Symposium on Signals, Systems, and Electronics*, Sep. 1998, pp. 295–300.

[10] S. Caban, C. Mehlführer, R. Langwieser, A. L. Scholtz, and M. Rupp, "Vienna MIMO testbed," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 54868, 2006.

[11] 3GPP, "Technical specification group radio access network; Multiple-Input Multiple Output in UTRA," 3GPP, Tech. Rep. 25.876 V1.7.0, Aug. 2004. [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/25_series/25.876/

[12] C. Mehlführer and M. Rupp, "A robust MMSE equalizer for MIMO enhanced HSDPA," in *Conference Record of the Fourtieth Asilomar Conference on Signals, Systems and Computers, 2006*, Pacific Grove, CA, USA, Oct. 2006.