Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 54868, Pages 1–13
DOI 10.1155/ASP/2006/54868

Vienna MIMO Testbed

Sebastian Caban, Christian Mehlführer, Robert Langwieser, Arpad L. Scholtz, and Markus Rupp

Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, Gusshausstrasse 25/389, 1040 Vienna, Austria

Received 2 December 2004; Revised 20 April 2005; Accepted 4 May 2005

While the field of MIMO transmission has been explored over the past decade mainly theoretically, relatively few results exist on how these transmissions perform over realistic, imperfect channels. The reason for this is that measurement equipment is expensive, difficult to obtain, and often inflexible when a multitude of transmission parameters are of interest. This paper presents a flexible testbed developed to examine MIMO algorithms and channel models described in literature by transmitting data at 2.45 GHz through real, physical channels, supporting simultaneously four transmit and four receive antennas. Operation is performed directly from Matlab allowing for a cornucopia of real-world experiments with minimum effort. Examples measuring bit error rates on space-time block codes are provided in the paper.

1. MOTIVATION

Investigating the performance of highly sophisticated wireless systems, in particular with multiple transmit and receive antennas, is a difficult task. In most cases, this can only be performed via simulation, which means modeling complex behavior by simpler mathematical descriptions [1]. Matlab simulation [2], with its highly accurate double-precision numerical environment, is on the one hand a perfect tool for the investigation of algorithms. On the other hand many imperfections of the *real-world* are neglected. In particular, fixed-point aspects of future products are often underestimated [3] as well as the true physical behavior of the wireless channel which is quite complex and not too well understood. Following the concept of simplification, uncertainties in MIMO decoding algorithms usually do not attract much attention [4]. These uncertainties are mainly simplified assumptions like perfectly known noise levels, additive Gaussian noise, omitted synchronization, linear power amplification, and perfect channel knowledge, which often show too optimistic receiver performance in simulation.

1.1. Motivation for rapid prototyping

Prototypes were used in the past to allow for early testing of future products. In particular when new technology was involved, such prototyping was a crucial element in the design path for a new product. The advantage of prototyping was to reduce the investment risk of the new product in case the new technology would hide unforeseen challenges. Also, one often obtained a flavour of how the new technology needs to be realized and a feeling for how the future product would look like. One could thus present it to potential customers, to work with them on particular features, long before the final product was available.

In wireless mobile communications, like personal communication, UMTS, WLAN, and so forth, time to market has become a real pressure. Since prototyping is cumbersome work, it is time and cost intensive, and thus a prototype with the above features may cost as much as the final product and may not be available earlier. Due to such reasons, prototyping is nowadays often skipped, building products while relying *only* on previous experience and simulation results.

However, experience has shown that such a procedure is quite risky, and a new movement, called rapid prototyping, has been established. In rapid prototyping, not the entire product is built but only the new and risky parts, possibly connected with already existing parts stemming from older products. It is not as thorough as the conventional prototyping approach, but rapid prototyping can prevent high-risk decisions and allows for a first location of challenging areas.

Such rapid prototyping is only successful if very flexible, scalable, and powerful hardware platforms are available that meet the complexity requirements of state-of-the-art MIMO algorithms. Furthermore, a consistent development environment is needed to reduce development time even further [3].

Note that depending on the purpose, different technical terms are being used (see also [5]).

(i) A *prototype* is the initial realization of a research idea or a standard, either as a reference, a proof of concept, or as a vehicle for future development and improvement.
(ii) A *demonstrator* mainly serves as a sales vehicle to show technology to customers. In general it will implement a new idea, concept, or standard that has already been established and has been finalized to some degree.
(iii) A *testbed*, on the other hand, is generally used for research. It is a vehicle for further development, for verification of algorithms, or ideas under real-world or real-time conditions. This results in the requirement for scalability, modularity, and extendibility.

We will thus refer to our development as testbed, rather than a prototype, throughout this article.

1.2. Motivation for the testbed

A multitude of development kits and prototyping boards are available, in particular supporting fast DSPs (e.g., C6x from TI [6] and SHARK from Analog) as well as FPGAs (e.g., from Xilinx and Altera). However, there are few concepts supporting multiple DSPs cooperating with multiple FPGAs, as only such a flexibility can support the required complexity in real time. Very few concepts support fast AD and DA converters, as they are required in this field. The next stage in the radio transmission chain requires intermediate-frequency (IF) signal generation. While we found very few companies supporting this, we did not find any supporting radio-frequency (RF) experimental boards. Thus we concluded that we have to at least develop the RF parts on our own, while for most other parts it would be advantageous to use existing platforms. We decided to utilize DSP and FPGA boards from Sundance [7] and IF boards from ICS [8, 9].

Once such a flexible and scalable platform is available, only a part of the rapid prototyping environment has been built. A typical user, that is, an algorithmic designer investigating coding and decoding schemes, is usually not able to program DSPs and FPGAs due to a lack of time and specialized knowledge. Therefore, he must be able to use such a testbed easily with nearly no effort directly out of an environment he knows well. It is thus advisable to use programming tools like Matlab that can support at least a few DSPs and FPGAs. Although such environments [10, 11] exist, they are still not suited to support an algorithmic designer without hardware experience, and they are not able to convert much more than a toy problem into a real-time experiment.

Note however, that many real-time experiments can be performed over a relatively small time period, say several milliseconds. In such cases, the complex DSP and FPGA platform is not really required. The data sequences can be preprocessed via Matlab and transmitted in burst mode over a real time air interface. Only the capturing of data needs to be performed in real time while the postprocessing can take place later in Matlab. Such operations simply require extensive data storage, which is not much of a cost factor today. We therefore decided to support scalable and flexible hardware platforms but put emphasis on real-time experiments performed directly out of Matlab.

Furthermore, multiuser operation can be quite useful, allowing several people to perform different experiments on a single wireless platform. Such feature is supported by a local area network (LAN) interface, allowing people to be distantly located from the measurement equipment.

1.3. Further goals for the testbed

Currently, our research activities using the MIMO testbed are the following.

(i) Investigation and comparison of equalizer structures for the high-speed downlink packet access (HSDPA) mode for UMTS. First results are published in [12].
(ii) Investigation and comparison of corporate proposals for the MIMO HSDPA mode, which are expected to be standardized in UMTS Release 7.
(iii) Other experiments include channel sounding and channel capacity measurements at 2.45 GHz.

Note that all these experiments are carried out at the same time by different research groups operating the single testbed from their own user PCs. More information on recent results can be obtained from [13].

After presenting the motivation and further goals in Section 1, Section 2 provides the details of the testbed concept and the realizations of the various subsystems. Section 3 demonstrates first real-time transmissions and shows the measurement results achieved. Section 4 finally reports on results when space-time codes are transmitted.

2. MIMO TESTBED DEVELOPMENT

2.1. Existing MIMO testbeds

Starting in 1998 with the first narrowband MIMO prototype [14], several MIMO prototypes were set up in the past to investigate a particular field of interest. This was, for example, MIMO WCDMA transmission for 3G telephone systems (by Lucent Technologies) [15], a MIMO 3G prototype chip development for high-speed downlink packet access reception (by Lucent Technologies) [16], or MIMO OFDM transmission for 4G telephone systems (by ETRI, Korea) [17]. While these prototypes and testbeds were very expensive, initial realizations of a research idea or standards, nowadays mostly testbeds are used in research due to their scalability, modularity, and extendibility.

About 60% [18] of these testbeds are operated by universities, usually assembled as a concatenation of some or all of the following flexible modules: PC, evaluation boards with DSPs and/or FPGAs, DA conversion, IF-to-RF upconversion, and the reverse chain for the receiver. In an early testbed design stage, signal processing is mostly performed in Matlab in form of a simulation, but some testbeds only focus on this "off-line" approach, as there is, for example [19], where data is generated completely on a PC, stored in files,

and then transmitted and received to be stored in files and processed by a PC again. Nevertheless, all other testbed designs approximately follow the above described structure.

At the University of Duisburg, Germany, a MIMO testbed was set up by utilizing Sundance components for baseband processing and Atmel RF components [20]. At the McGill University, Montreal, Canada, a similar structure is used to create any signal which fits into 3.5 MHz by utilizing ICS boards and extensive hard-disk storage [21]. At the Rice University, Houston, Tex, Nallatech FPGA boards are favored for their modular testbed approach [22]. Slightly different, at Young University, Brigham, Utah, USA, data samples are generated in "real-time embedded PCs" plus TI DSPs [23]. At Georgia Tech, Ga, USA, several synchronized Agilent 4438 arbitrary waveform generators are utilized as signal generators for their testbed [24]. To show some more examples, MIMO testbeds are also assembled and operated at RWTH Aachen, Germany [25], at Virginia Tech, USA [26], and at ETH Zurich, Switzerland [27].

2.2. Our MIMO testbed concept

Our testbed resembles the above-mentioned examples and extends them by an easy to use Matlab interface to allow for a multitude of experiments with very different constraints. Our testbed is not developed for a single type of experiment or signal, which means that there is no need to redesign parts of the design or signal processing in order to allow several researchers and research teams to share the same expensive testbed hardware for different experiments. Any transmission can be performed directly out of Matlab from any PC in the local area network (LAN) by utilizing the same convenient interface, the Matlab interface. Easy, flexible, and without requiring specific hardware knowledge, this platform independent interface leads to reduced development time for a working prototype. Cost for hardware is reduced by sharing the testbed among several research teams.

The developed MIMO testbed [28, 29] primarily consists of a PC with transmitter capabilities (transmit PC), channel realizations (in form of emulation or physical channel), a PC with receiver capabilities (receive PC), and one or more user PCs from where the transmit and receive PCs are being operated via a LAN connection as depicted in Figure 1.

The various components will be described in more detail in the following. Note that this concept allows the algorithmic designer to focus on digital baseband data processing on his *own* PC. Via the so-called Matlab interface, the data streams of up to four transmit antennas are submitted to the transmit PC. There, these samples are automatically up-converted to the intermediate frequency (IF), transmitted over a radio-frequency (RF) channel, and once they have been captured by the receive PC, they are finally down-converted to the baseband to provide the received raw (unfiltered) baseband data samples. The developed Matlab interface is so flexible that the length of the data stream, the data rate, and the selected IF can be selected from a wide range of parameters, simply handled by setting such parameters in Matlab.

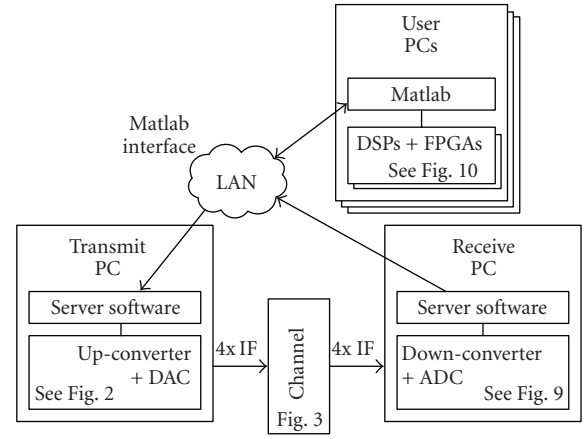


Figure 1: Block diagram of the testbed developed.

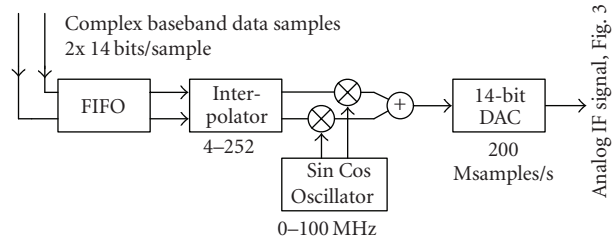


Figure 2: Per channel data flow in the transmit PC.

This platform-independent approach allows multiple users to access the testbed directly out of Matlab from anywhere in the LAN. This not only makes it very convenient to integrate real-time air transmissions into existing Matlab simulations by the use of a few Matlab commands, but also saves a lot of costs since very expensive hardware can be operated efficiently by several researchers simultaneously.[1]

2.3. The transmit PC

The user, operating the transmit PC directly out of Matlab, provides baseband data samples and a set of options, for example, the interpolation rate and the desired intermediate frequency. A server software processes these samples and also controls the ICS 564 plugin PCI board [8] which performs the signal conversion to the IF for all four channels as illustrated in Figure 2.

The incoming samples (up to 50 Msamples/s on up to 4 channels) are buffered by a FIFO (128 ksamples per channel) and then interpolated (user definable between 4 and 252) to reduce the required baseband data rate. Up-conversion to the IF is carried out digitally before conversion to the analog

[1] The transmit PC with its up-converter PCI board costs approximately 30 000 €, the receive PC nearly the same, the RF components for the channel cost approximately 10 000 €, the two channel emulators 150 000 € each, and the noise source 30 000 €.

domain. Utilizing a low IF (user definable between 0 and 100 MHz), rather than baseband signals, avoids the problem of DC offset, carrier leakage, and IQ imbalance while saving half the numbers of DACs.

Furthermore, the Matlab interface allows for various extended options. A user may introduce a delay (user definable in milliseconds) between subsequent blocks and/or repeat one or more blocks automatically as desired. This is especially useful if a lot of different channel realizations are of interest.

2.4. The RF channel

The channel setup can either be a physical channel operated via the RF air interface at 2.45 GHz or channel emulators (e.g., Spirent Communications TAS4500 FLEX [30]), allowing to repeat experiments with a wide range of channel parameters. The setup of Figure 3 was taken into operation to realize a 4×1 MIMO channel.

The RF front end consists of independent RF transmitters and receivers for each channel with the exception of a common oscillator for all transmitters and receivers. The use of separate modules provides high flexibility for development and testing. Since no restrictions should apply with respect to the modulation scheme, all RF modules were designed to be highly linear. In order to fulfill the requirements for the WLAN standard IEEE 802.11b [31], the modules provide a maximum bandwidth of 20 MHz and about 20 dBm transmit power per channel. Unfortunately, the RF front end cannot be built as flexible as the Matlab interface. We thus had to fix the IF to 70 MHz and RF to 2.45 GHz.

Optionally, additional noise can be added by a UFX-EBNO-IF1 [32] noise source. This is in particular useful when comparing MIMO decoding algorithms for a single user operating at a dedicated SNR level.

2.4.1. Transmit via channel emulators

The usage of two Spirent TAS4500 FLEX channel emulators [30] is very beneficial when developing MIMO transmission systems. With the channel emulators, a controllable, predictable, and repeatable environment can be set up, allowing for various typical channel conditions. The ability to reproduce different measurement scenarios as accurately as possible is of great advantage for algorithmic evaluation and debugging. Furthermore, in this mode there is no need for operating licenses even when operating outside the ISM band because no radiation is emitted.

2.4.2. Transmit via antennas

Since all channel models are based on simplified assumptions, a transmission via a realistic, physical channel may cover some important and maybe yet unknown aspects which are not taken into consideration by channel emulators.

To transmit and receive the RF signal at 2.45 GHz, $\lambda/4$-monopoles mounted on a ground plane (see Figure 4) are used as antennas. They are placed in a uniform linear array with variable element spacing. The MIMO testbed itself is not limited to this kind of antennas and can support other configurations.

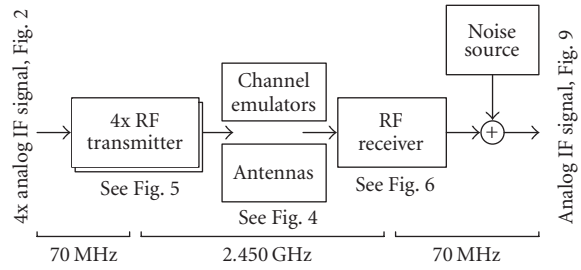


Figure 3: MIMO channel setup utilizing emulators or the air interface.

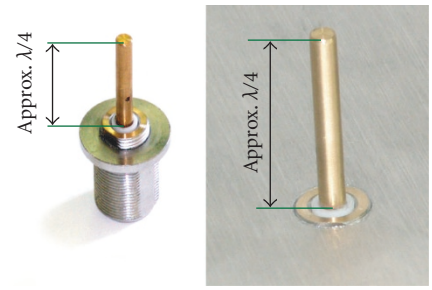


Figure 4: Antenna (a) without and (b) with ground plane.

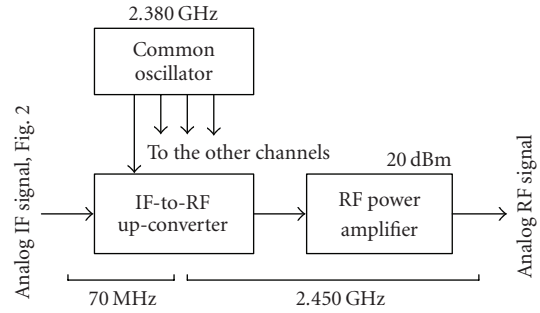


Figure 5: RF transmitter block diagram, per channel.

2.4.3. RF transmitter

The RF transmitter consists of an IF-to-RF up-converter, an RF power amplifier with 20 dBm maximum output power for each channel, and a common oscillator for all up-converters. For transmissions through the channel emulators, only little mean power of -20 dBm is required so that the power amplifiers are not used in this mode. Figure 5 depicts the block diagram for one of the four channels.

IF-to-RF up-converter

Since the IF output of the transmit PC is not filtered, higher harmonic frequencies are also created by the digital-to-analog conversion in addition to the desired IF signal. In the up-converter, at first this 70 MHz IF signal is filtered by using a low-loss surface acoustic wave (SAW) filter with a 3 dB bandwidth of 20 MHz. In order to reduce out-of-band reflections distorting the DAC operations, an isolating attenuator is placed in front of the IF filter. The up-conversion from the IF to the RF is performed in one step by a double-balanced mixer. This mixer is followed by an amplifier and an RF bandpass filter to create an output signal with enough signal level to drive the channel emulators. The up-converter module is operated on a single 3 V voltage supply and a −15 dBm signal from the common oscillator.

RF power amplifier

Amplification to the transmit power level of 20 dBm is carried out in two stages, with RF bandpass filters placed in between. The power consumption of the power amplifier is about 2 W and it is operated on a single 5 V voltage supply.

2.4.4. RF receiver

The RF receiver consists of three modules for each channel: a low-noise amplifier (LNA), an RF-to-IF down-converter, and a gain-controlled amplifier (GCA). The common oscillator can be used by up to four down-converters. Figure 6 shows the RF receiver block diagram.

Low-noise amplifier

At the input of the low-noise amplifier module, a receive filter is needed to protect the following low-noise transistor from blocking because of many other radio applications in the 2.4 GHz ISM band and the nearby UMTS band. A second filter after the LNA is used to additionally suppress image frequency signals. This filter is followed by an amplifying buffer.

RF-to-IF down-converter

Down-conversion to IF is performed by a double-balanced mixer, using a common oscillator signal with a power level of −10 dBm. A low-loss SAW IF filter follows to limit the noise bandwidth of the RF system to 20 MHz and to filter undesired spectral components to prevent aliasing in the following AD conversion. The mixer uses a single 3 V supply, the low-noise amplifier is operated on a single 10 V supply.

Gain-controlled amplifier

The last module of the RF receiver is a gain-controlled amplifier. Its gain can be defined in a range of 70 dB by a control voltage (GCTRL); therefore, it is possible to provide a constant mean power to the receive PC for receive signal levels

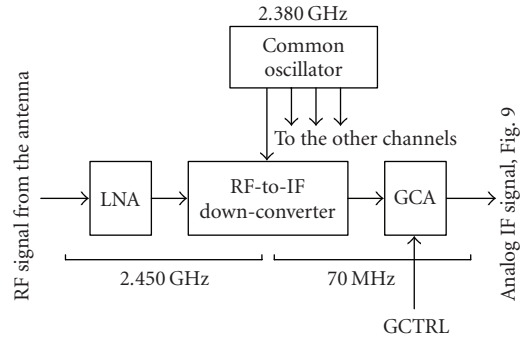


Figure 6: RF receiver block diagram, per channel.

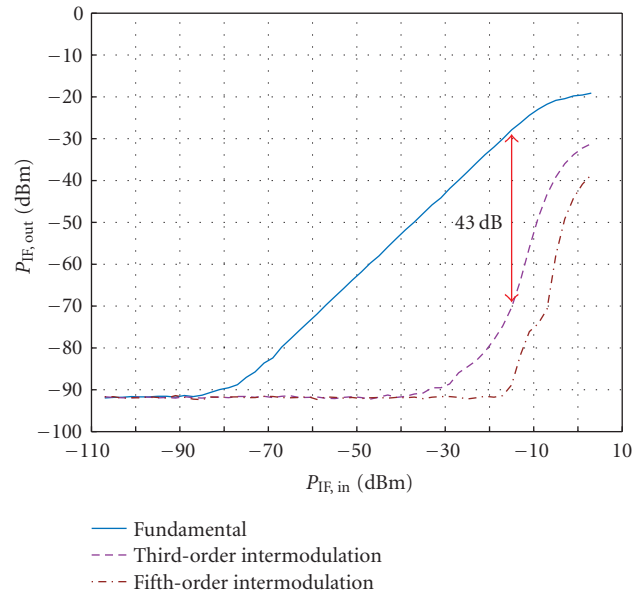


Figure 7: Measured intermodulation for concatenated analog IF-to-RF up- and RF-to-IF down-converter modules.

from the noise floor (−100 dBm) to the maximum signal level expected for 1 m distance between transmit and receive antennas (−20 dBm).

2.4.5. RF measurements

Figure 7 shows the result of a two-tone experiment where the overall RF system was measured with the up- and down-converter modules concatenated via a 20 dB attenuator placed in between. Two synthesizers were used to create the 70 MHz ± 10 kHz input signal.

Figure 8 shows the overall frequency response of the concatenation of the analog IF-to-RF up-converter and RF-to-IF down-converter modules. The overall 6 dB bandwidth is 20 MHz since the two modules were each designed to have a 3 dB bandwidth of 20 MHz. The maximum deviation from the desired flat response within the maximum transmission bandwidth of 6.25 MHz was measured to be 1.5 dB.

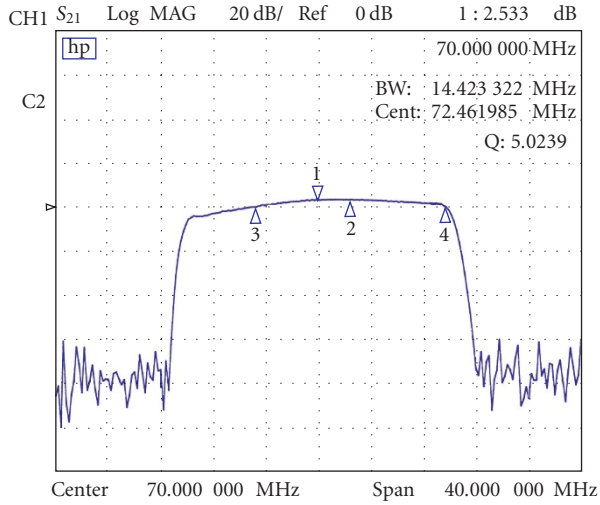


Figure 8: Measured frequency response for concatenated analog IF-to-RF up- and RF-to-IF down-converter modules.

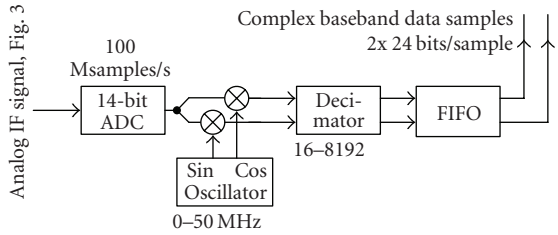


Figure 9: Data flow in the receive PC, per channel.

2.5. The receive PC

The receive PC is remote-controlled by the transmit PC. It automatically down-converts the four incoming analog IF signals to digital baseband data samples at a maximum baseband sample rate of 6.25 MHz (see Figure 9). This limits the maximum possible system bandwidth to about 6 MHz.

Down-conversion to baseband is carried out by an ICS 554 [9] PCI plug-in board and controlled by a server software which finally provides the received raw samples to the user of the testbed. Like in the up-converter boards, the down-conversion to baseband is carried out after the AD-conversion to avoid DC offsets and IQ imbalancing.

2.6. DSP/FPGA boards

In a rapid prototyping design flow, the first step for the algorithmic designer is to show the proof of concept by a Matlab implementation. In a second step, the algorithms can be mapped step-by-step on more realistic hardware platforms. To achieve this, the designer may extend his own PC (see Figure 1) by FPGA and DSP boards (like the SMT 365 DSP/FPGA PCI plug-in boards [7] from Sundance) to investigate critical parts of the design in a suitable real-time environment.

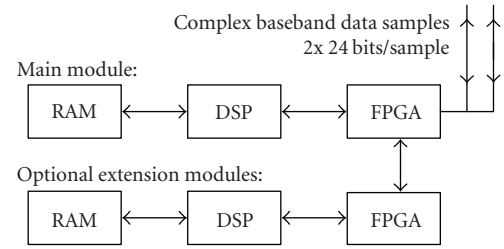


Figure 10: Data flow in the DSP/FPGA modules.

These DSP/FPGA boards are equipped with a fixed-point DSP (600 MHz, 4800 MIPS peak performance, Texas Instruments TMS320C6416), a Xilinx FPGA (Virtex II XC2V1000-4-FF896), and 8 Mbytes of RAM as depicted in Figure 10.

By adding more DSP modules, a prototyping engineer can decide at a later stage how many DSPs are really required to meet real-time constraints since the 3L-Diamond operation system also allows to map compiled procedures onto several DSPs.

High-speed data transfer (approximately 25 MBps) from/to Matlab via the PC's internal PCI bus is also possible; therefore, the initial, working, and tested receiver design implemented only in Matlab can be replaced part-by-part by signal processing algorithms executed directly on DSPs and/or FPGAs (Matlab MEX-files). This step-by-step replacing approach eases debugging and saves on code development time.

2.7. Matlab interface

An easy to use Matlab interface was developed to operate the overall testbed from one or more user PCs. Utilizing ordinary files to handshake and exchange data, this interface reduces code development and debugging time in a Matlab client application, in addition to making the testbed accessible from anywhere in the LAN (or even the internet by using a secure-FTP connection). Platform independence is guaranteed because Matlab supports various operating systems. Optimized for the use as file servers, the transmit and receive PCs are able to store and provide user data without disrupting ongoing transmissions.

The following Matlab code presents a code example which transmits four identical 16-QAM signals on four channels. Note that no hardware knowledge is needed to perform this transmission. In order to transmit data via the Matlab interface, a user sets up a channel and then the following steps need to be performed, all together not more than 15 lines of simple Matlab code.

(I) *Generate complex baseband data samples* on a local PC in Matlab and it is optional to use DSP/FPGA boards. For example, four identical 16-QAM signals, RRC-filtered, four times oversampling, roll-off factor = 0.22 on each channel:

```
xTXData = [     randsrc(5000, 1, [-3 1 1 3]/3)...
           + j*randsrc(5000, 1, [-3 1 1 3]/3) ];
xFilter = rcosine (1, 4, 'fir/sqrt', 0.22, 40);
xTXData = rcosflt (xTXData, 1, 4, 'filter', xFilter);
xTXData = int16   ((xvTXData*ones(1,4)).*8176);
```

(II) *Provide parameters*. For example, transmit and receive (3) the signal sixty times (60) with a delay of 40 ms (40) between subsequent blocks at a center frequency of 70.0 MHz (70.0) utilizing an interpolation of 64 (64) at the transmitter and of 32 (32) at the receiver. The other parameters can be used to utilize a lot of additional features as there are, for example, external clock synchronization, predesigned received signal filters implemented in the testbed hardware, and several possibilities of repeating measurements in order to receive data for different channel realizations:

```
xTXOptions = [3,60,70.0,64,0,40,0,0,0,0,32,0,0]
```

(III) *Save all* this to a .mat-file on a dedicated network path and create a second file to trigger the transmission:

```
xTXDir     = '\\Pollux\MIMO_TXData\';
xFileName  = 'Test 16 QAM';
save([xTXDir xFileName '.mat'],'xTXData','xTXOptions');
xFID=fopen([xTXDir xFileName '.do'],'w'); fclose(xFID);
```

(IV) *Wait for the received data* to appear in a corresponding .mat-file file on a dedicated network path:

```
while size(dir([xTXDir xFileName '.done']),1)==0
pause(0.2); end;
```

(V) *Process the received* complex baseband data *samples* in Matlab (and it is optional to use DSP/FPGA boards to realize parts of the design in a real-time environment close to the final product). For example, print the magnitude of channel three of the 15th reception:

```
xRXDir  = '\\Castor\MIMO_RXData\';
load([xRXDir xFileName '_15.mat']);
stem(abs(double(xTXData(:,3))))
```

By using the Matlab interface, the user gains full flexibility while it is still easy for him to perform and understand the first transmission of baseband data samples out of Matlab. Therefore, the user can focus on the data generation and reception algorithms (items 1 and 5) without "wasting" time devoted to hardware issues.

While the transmit PC and the receive PC load, process, and store data in real time and thus the whole data transmission is carried out in real time, the data transfer from and to the user PC (by the use of the Matlab interface) is *not* real time. Especially if a form of real time feedback is not required, this non-real-time transmission of data samples from and to the user PC does not change the investigation of, for example, an algorithm at all, but reduces cost on setting up the testbed in addition to making the overall testbed more convenient to use.

Up till now (April 2005), the testbed has been working flawlessly without any changes in hard- or firmware[2] for over 10 months.

[2] Only the RF frontend has been adapted for the needs of specific experiments.

Table 1: Key parameters for typical testbed operation.

Channels	Up to 4×4
Signal type (modulation)	Any
Samples per TX/RX block[a]	< 131 072
TX DAC resolution	14 bits
TX intermediate frequency	70 MHz
Radio frequency	2.45 GHz
Transmit bandwidth	< 6.25 MHz
RX intermediate frequency	70 MHz
RX ADC resolution	14 bits
Data transfer to hard disk	< 50 MBp/s
Time to fill hard disk (60 GB)	approximately 20 min
LAN transfer to client PC	< 9 MBp/s

[a] If the sample data rate is smaller than 25 Msamples/s, a block can be up to 20 000 000 samples long. (Otherwise the maximum block length is 131 072.)

2.8. Testbed summary

Real-time transmission experiments over an air interface are generally avoided due to the enormous costs, the hardware skills and software knowledge required, the uncontrollable and unpredictable environment of a real channel, and the unaffordable time until the first results may be achieved. Using the MIMO testbed developed, neither hardware programming skills nor a lot of time is needed to transmit and receive complex baseband data samples via air (or a predictable channel built up of channel emulators).

The MIMO testbed developed (see Table 1) is able to transmit and receive complex baseband data samples on up to 4×4 antennas. Users can access this system from anywhere in the LAN by the use of simple Matlab commands (see code example). By implementing algorithms in Matlab and migrating their critical parts step-by-step in DSP/FPGA boards, researchers are enabled to detect design flaws in an upcoming product very early in the design process with minimum effort. The experimental setup was kept very flexible (parametric), allowing for various experiments supporting a multitude of wireless standards. While still maintaining flexibility, sharing the same RF hardware among several researchers and research teams significantly reduces the arising costs.

3. FIRST MEASUREMENTS

Since most MIMO systems are based on underlying SISO systems, it appears useful to develop the baseband processing for a MISO implementation on the basis of a suitable successfully tested SISO link. Therefore, a 16-QAM transmitter and receiver was implemented in Matlab before continuing with further MISO transmissions.

As previously described, the Matlab code is executed on the "user PC" while the actual transmission of complex baseband data samples is carried out via the Matlab interface. The need to synchronize for frequency offset, phase, and frame timing in the receiver turned out to be the first obstacle to overcome before the successful implementation of a

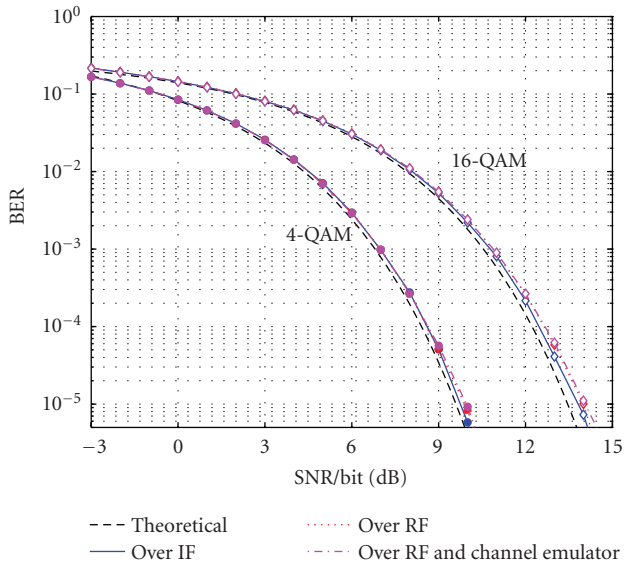


Figure 11: Measured BER for a static SISO RF channel.

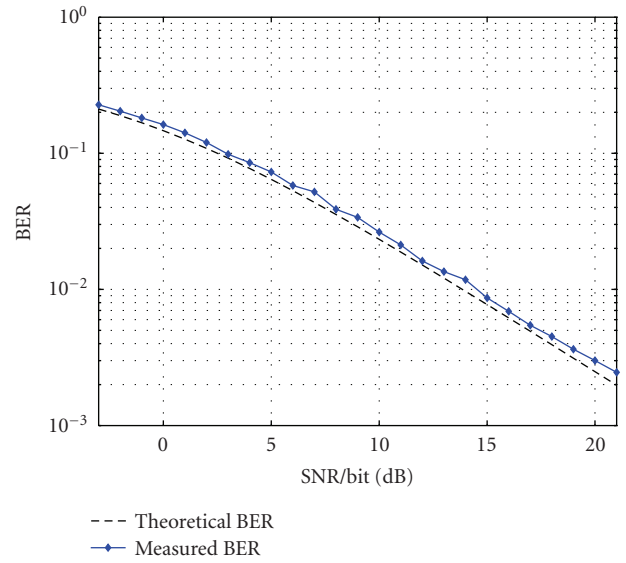


Figure 12: Measured BER for a flat Rayleigh fading SISO RF channel (4-QAM).

ZF, MMSE, and ML receiver. A detailed description of the applied synchronization will be given in Section 4 when the succeeding MIMO implementation will be discussed.

3.1. End-to-end operation

First measurements were carried out by bypassing the channel (see Figure 1) and thus directly connecting the IF outputs of the transmit PC to the IF inputs of the receive PC, using a lowpass filter to suppress higher harmonics generated by the DA conversion. One Gbyte of data was transmitted error free using this setup.

3.2. Measurements via channel emulators

After successfully proving the basic transmitter and receiver operation, further SISO measurements were carried out utilizing channel emulators, and a noise generator. For a static channel (AWGN, Figure 11), as well as a slowly varying Rayleigh fading channel (Figure 12), the measured bit error rate curves for 4-QAM modulation showed exact matching with well-known theoretical results within a margin of 0.2 dB.

3.3. Wireless SISO measurements

First measurements via an air interface were performed to get an impression on how the testbed works under real channel conditions. The testbed configuration used in the above measurements was extended by a power amplifier and two of the earlier described $\lambda/4$-monopole antennas. The receiver antenna (RX) was placed at a distance of about eight meters from the transmit antenna (TX) in a non-line-of-sight (NLOS) scenario (Figure 13). This kind of scenario was chosen because MIMO experiments will be accomplished using the same scenario.

An RRC-filtered (roll-off = 0.22) 4-QAM signal with a block length of 8 000 symbols was transmitted in order to measure the BER of the received signal. The block length corresponds to a transmission time of approximately 5 milliseconds. Since the measurement was carried out inside a laboratory, and only slowly moving scatterers occurred, a constant channel can be assumed during the transmission of one data block. Additionally, since the symbol duration of 640 nanoseconds is about one order of magnitude larger then the delay spread of the indoor channel, we assume a channel without any time dispersion. Because of the nonconstant measurement environment, the SNR of every received data block has to be estimated. Together with the calculated BER value, this gives the coordinates for one BER measurement point.

Figure 14 depicts the theoretical result for an AGWN channel compared to the measured data block results (crosses). This comparison of BER values measured in a small-scale scattering environment with an AWGN channel is indeed possible because of the separate SNR estimation of each received data block. Different SNR values were generated by varying the transmit power. By using the maximum available transmit power, an SNR value of about 25 dB was measured at the receiver.

4. MEASUREMENTS WITH MISO SPACE-TIME BLOCK CODES

Channel adaptive extended Alamouti code

Well described in [33], the channel adaptive extended Alamouti code (CAEAC), which uses partial feedback of channel information to the receiver, "exploits nearly full diversity with a zero forcing receiver as well as with a maximum likelihood detector." Measurements using the MIMO testbed

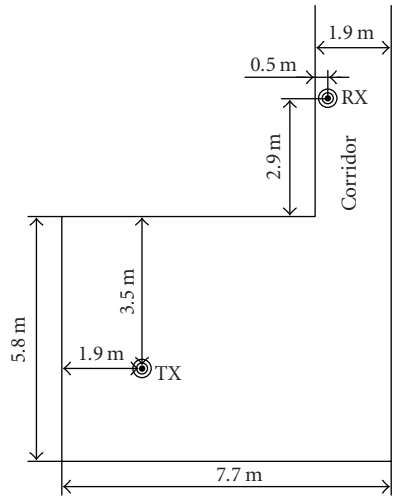


Figure 13: Measurement setup for the SISO transmission over the air: antennas are 1.4 m above the floor; room height is 3.5 m; the room contains a lot of tables, chairs, computers, and so forth.

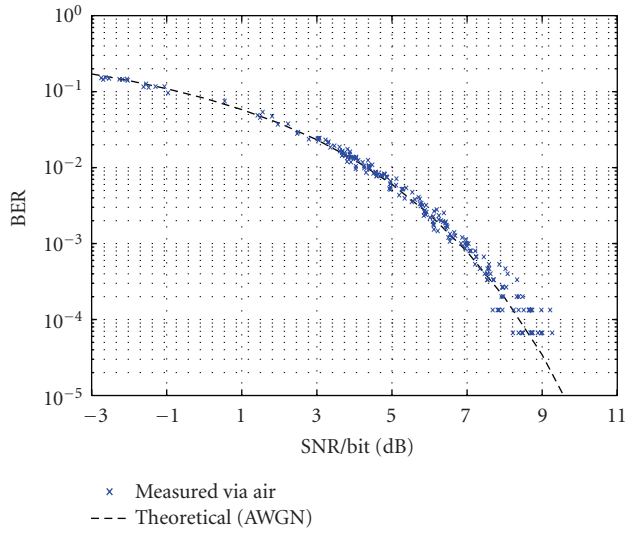


Figure 14: Measured BER in smale-scale scattering setup.

were carried out to investigate this behavior for physical, imperfect channels. Different channel realizations were created by channel emulators to obtain reproducible results.

4.1. Matlab transmitter

The transmitter, fully implemented in Matlab, generates complex data symbols from a 4-QAM constellation and performs extended Alamouti coding for four transmit antennas. Root raised cosine (RRC) filters produce the complex baseband samples of the transmit signals from the given symbols. These samples are transferred to the transmit PC via the Matlab interface as depicted in Figure 15.

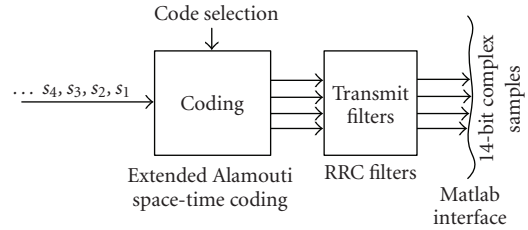


Figure 15: The CAEAC transmitter implemented in Matlab.

4.2. Channel setup

As described in Section 2.4, the channel is mainly composed of RF up- and down-converters and channel emulators which were used to generate flat Rayleigh fading. After the down-conversion process, the noise generator corrupts the IF signal with white Gaussian noise to allow BER over SNR measurements (see Figure 3). The emulators can be substituted by amplifiers and antennas to facilitate measurements over a real-time air interface once the correct functionality of the codes-under-test has been successfully proven.

4.3. Matlab receiver

The Matlab receiver, which obtains samples of the received IF signal via the Matlab interface, consists of the following blocks as illustrated in Figure 16.

Since the receiver PC does not know about the optimum sampling time instant, at least timing and frame synchronization have to be performed in Matlab. The frequency offset estimation was initially also performed in Matlab, but to reduce calculation time, we decided to synchronize the frequency by using an external cable. This optional cable provides a direct connection between the internal oscillator of the transmit PC (Figure 2) and the internal oscillator of the receive PC (Figure 9) to lock them in frequency.

The synchronization process is further simplified by the fact that the Matlab interface itself ensures (by triggering via the LAN) that the data is transmitted and received approximately at the same time. Although this does not by any means exactly synchronize transmitter and receiver, it does help to approximately know where the received data block starts. This knowledge allows for the later described maximum search. Figure 17 illustrates the synchronization algorithm which was implemented in Matlab to synchronize timing and frame separately for every transmitted data block.

The synchronization algorithm correlates interpolated versions of the input data samples with the four training sequences of the transmitters. These training sequences are Gold sequences of length 511, allowing for nearly perfect channel estimation. The training sequences are transmitted as a midamble within the data (Figure 18). The output functions of the four correlators are summed up quadratically and the maximum is searched for within approximately twice the length of the training sequence. This maximum indicates the optimum sampling time instant and the beginning of the

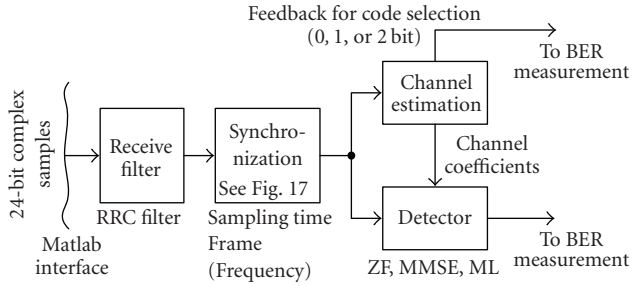


Figure 16: CAEAC receiver implemented in Matlab.

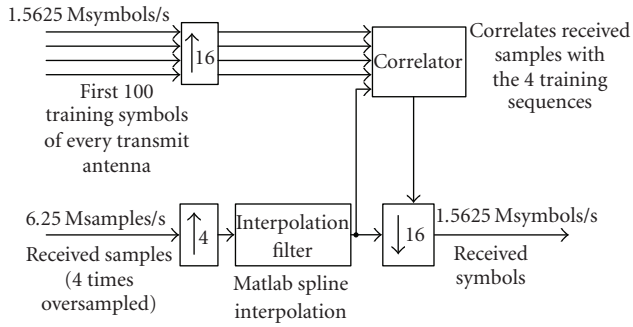


Figure 17: Synchronization scheme in Matlab.

transmission frame. At last a down-sampling to symbol rate is performed and the receive symbols are delivered to the succeeding stages of the Matlab receiver.

The synchronized data samples are then used for estimating the channel coefficients in a least squares estimator. The channel estimator also selects the optimum extended Alamouti space-time code for the transmission. This code selection uses a feedback of at most two bits per frame to select one out of four codes. The actual implementation of the feedback is described in Section 4.4.

By the use of the estimated channel coefficients, the detector reconstructs the transmitted symbols. A low-complexity solution of the maximum likelihood receiver [34] was implemented to measure the bit error rates for different numbers of feedback bits.

4.4. "Virtual" feedback

In the case of channel adaptive extendeded Alamouti codes, channel information is required to be fed back to the transmitter in order to select a code. The therefore needed real-time operation would typically require an implementation in FPGAs and/or DSPs. To avoid this time intensive task, the feedback necessary for the CAEAC was only implemented "virtually" which means transmitting all four possible codes in one large transmit block as depicted in Figure 18 and selecting the optimum block in the receiver, rather than in the transmitter.

The transmitted block consists of four small data blocks, each coded with a different EAC, and a midamble as training

320 symbols	code 3
320 symbols	code 1
4×511 training symbols	
320 symbols	code 2
320 symbols	code 4

Figure 18: Structure of one transmit block.

Table 2: Number of channel realizations per measurement point.

$-3\,\text{dB} \le \text{SNR/bit} < 8\,\text{dB}$	15 000 channels
$8\,\text{dB} \le \text{SNR/bit} < 12\,\text{dB}$	30 000 channels
$12\,\text{dB} \le \text{SNR/bit} \le 17\,\text{dB}$	60 000 channels

sequence. The calculated feedback bits are not sent back to the transmitter; they are only used to select the optimum extended Alamouti code block (for evaluating the BER) in the receiver. This procedure also allows for measuring the BER for 0 (just code 1 is used), 1 (either code 1 or code 2 is used), and 2 bit feedback (all four codes are used) with the same blocks of received data.

4.5. Measurement results with channel emulators

Smooth BER curves require a huge amount of data to be transmitted. Table 2 indicates the number of different channel realizations used for measuring the following BER curves.

We used less channel representations for the lower-SNR regions since the BER is expected to be higher there. This amount of transmit data also required a large computational capacity. A cluster of ten personal computers running Matlab needed about one day to evaluate the received data.

Since the received data provided by the Matlab interface was already split up in files for the different SNR values, it suited perfectly the needs of a parallel processing cluster: self-written server-software-managed 14 client PCs, each evaluating the BER for a specific block. By collecting all these results, the final BER plot (e.g., Figure 19) was obtained. The effort needed to set up this processing cluster was still negligible compared to the effort which would have been necessary to code all these algorithms into DSP/FPGAs in order to gain more speed.

The measured BER curves for the CAEAC are compared with theoretical results (obtained by performing a Matlab simulation) in Figure 19. The results show that the simulated diversity improvement of the feedback is verified by the measurements, although they are shifted about 0.8 dB at a BER of 10^{-3}. This shift is mainly due to some imperfections in the analog measurement setup. During all measurements more than 5 terabytes of data were transmitted, received, and evaluated. Using a single PC, this would have taken two weeks—utilizing the processing power of a PC cluster, the evaluation time becomes comfortable, allowing for a complete BER curve measurement per day as, for example, Figure 19.

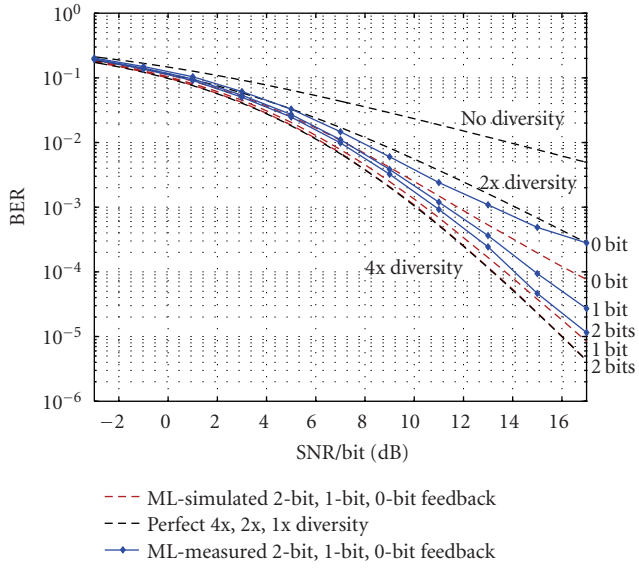


Figure 19: Measured BER curve for CAEAC coding (maximum likelihood receiver).

4.6. Summary of measurements

The first SISO measurements performed on the MIMO testbed showed the perfect interaction of Matlab, the Matlab interface, the digital baseband hardware, the radio-frequency hardware, and the channel emulators. For static and fading channels, the measured bit error rate curves showed exact matching with well-known theoretical results.

The MISO measurements verified the simulated diversity improvement of the feedback used in channel adaptive extended Alamouti codes.

5. CONCLUSION

Combining the advantages of Matlab and a DSP/FPGA environment close to the final product, the MIMO testbed developed allows for rapid verification of baseband algorithms and their critical parts with minimum effort. Proof of operation has already been achieved by the successful implementation of a channel adaptive extended Alamouti space-time code transmission showing excellent performance. Sharing very expensive equipment while still maintaining flexibility and ease of use directly out of Matlab (from any PC in the network), a testbed has been provided that allows for many new coding and modulation schemes to be conveniently investigated in shortest time.

ACKNOWLEDGMENTS

The authors would like to thank Lukas Mayer, Ernst Aschbacher, Werner Keim, Stefan Geirhofer, Georg Maier, and Biljana Badic for their help and suggestions on improving the testbed. This work has been funded by the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms (http://www.nt.tuwien.ac.at/cdlab/).

REFERENCES

[1] D. Gesbert, M. Shafi, D.-S. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, 2003.

[2] The MathWorks, MATLAB, http://www.mathworks.com/.

[3] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, no. 7, pp. 1427–1444, 2003.

[4] M. Rupp, "On the influence of uncertainties in MIMO decoding algorithms," in *Proceedings 36th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 570–574, Monterey, Calif, USA, November 2002.

[5] A. Burg and M. Rupp, *Chapter: Demonstrators and Testbeds*, EURASIP Book on SMART Antennas, 2004, http://www.nt.tuwien.ac.at/cdlab/ConferencePresenation/DemonstratorsTestbeds.ps.

[6] Texas Instruments Incorporated, TMS320C6XXX DSPs, http://dspvillage.ti.com/docs/allproducttree.jhtml?pageId=C6.

[7] Sundance Multiprocessor Technology, SMT 365, http://www.sundance.com/edge/files/productpage.asp?STRFilter=SMT365.

[8] Interactive Circuits Systems, ICS 564 Analog Output Boards, http://www.ics-ltd.com/info/productInfo.cfm?ID=13&prod=2.

[9] Interactive Circuits Systems, ICS 554 Analog Input Boards, http://www.ics-ltd.com/info/productInfo.cfm?ID=10&prod=1.

[10] Xilinx, System Generator, http://www.xilinx.com/products/software/sysgen/features.htm.

[11] The Mathworks, Embedded Target for TI C6000, http://www.mathworks.com/products/tic6000/.

[12] S. Geirhofer, C. Mehlfuhrer, and M. Rupp, "Design and real-time measurement of HSDPA equalizers," in *Proceedings of 6th IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC '05)*, New York, NY, USA, June 2005.

[13] C. Mehlfuhrer, S. Geirhofer, S. Caban, and M. Rupp, "A flexible MIMO testbed with remote access," in *Proceedings of 13th European Signal Processing Conference (EUSIPCO '05)*, Antalya, Turkey, September 2005.

[14] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-Blast: an architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proceedings of International Symposium on Signals, Systems, and Electronics (ISSSE '98)*, pp. 295–300, Pisa, Italy, September–October 1998.

[15] A. Adjoudani, E. C. Beck, A. P. Burg, et al., "Prototype experience for MIMO BLAST over third-generation wireless system," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 440–451, 2003.

[16] D. Garrett, G. Woodward, L. Davis, G. Knagge, and C. Nicol, "A 28.8 Mb/s 4×4 MIMO 3G high-speed downlink packet access receiver with normalized least mean square equalization," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC '04)*, vol. 1, pp. 420–536, San Francisco, Calif, USA, February 2004.

[17] S. H. Won, D.-S. Lyu, and H. J. Park, "Physical layer implementation and evaluation of multiple input multiple output - orthogonal frequency division multiplexing (MIMO-OFDM) system," in *Proceedings of International Conference on Communication Technology (ICCT '03)*, vol. 2, pp. 1348–1352, Beijing, China, April 2003.

[18] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems - an overview," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO '04)*, pp. 681–688, Vienna, Austria, September 2004.

[19] J. Rinas, R. Seeger, and K. D. Kammeyer, "A demonstrator for multi-antenna transmission - real channels and their impact on MIMO algorithms," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[20] T. Kaiser, A. Wilzeck, and R. Tempel, "A modular multi user testbed," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[21] R. Morawski, T. Le-Ngoc, and O. Naeem, "Wireless and wireline MIMO testbed," in *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '03)*, vol. 3, pp. 1913–1916, Montreal, Canada, May 2003.

[22] P. Murphy, F. Lou, A. Sabharwal, and J. P. Frantz, "An FPGA based rapid prototyping platform for MIMO systems," in *Proceedings of 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 900–904, Monterey, Calif, USA, November 2003.

[23] J. W. Wallace, B. D. Jeffs, and M. A. Jensen, "A real-time multiple antenna element testbed for MIMO algorithm development and assessment," in *Proceedings of IEEE Antennas and Propagation Society Symposium (APS '04)*, vol. 2, pp. 1716–1719, Monterey, Calif, USA, June 2004.

[24] G. L. Stuber, J. R. Barry, S. W. McLaughlin, Y. Li, M. A. Ingram, and T. G. Pratt, "Broadband MIMO-OFDM wireless communications," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 271–294, 2004.

[25] D. Borkowski and L. Brühl, "Hardware implementation for real-time multi-user MIMO systems," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[26] S. Ellingson, R. Mostafa, and J. H. Reed, "MIMO development efforts at virginia tech," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[27] S. Häne, D. Perels, D. S. Baum, et al., "Implementation aspects of a real-time multi-terminal MIMO-OFDM testbed," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[28] S. Caban, R. Langwieser, C. Mehlfuhrer, et al., "Design and verification of a flexible and scalable 4×4 MIMO testbed," in *Proceedings of IEEE Radio and Wireless Conference (RAWCON '04) Workshop 2 on MIMO Implementation Aspects*, Atlanta, Ga, USA, September 2004.

[29] E. Aschbacher, S. Caban, C. Mehlfuhrer, G. Maier, and M. Rupp, "Design of a flexible and scalable 4×4 MIMO testbed," in *Proceedings of 11th IEEE Digital Signal Processing Workshop (DSP '04),* pp. 178–181, Taos Ski Valley, NM, USA, August 2004.

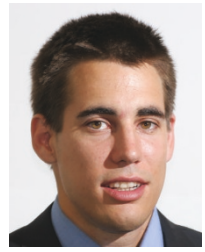
[30] Spirent, TAS4500 FLEX RF Channel Emulator, http://www.atsweb.it/Images/Documenti/TAS_DS_4500.pdf.

[31] *802.11b, Part 11: Wireless LAN Medium Access Control (MAC) And Physical Layer (PHY) Specifications: Higher-speed Physical Layer Extension In The 2.4 GHz Band*, pp. i–90. IEEE Standard 802.11b-1999, 2000, http://ieeexplore.ieee.org/iel5/6642/17714/00817038.pdf.

[32] Noisecom, UFX-EBNO-IF1 Precision Noise Generator, http://www.noisecom.com/content/Products/Components/UFX_EbNo/UFX_EbNo.pdf.

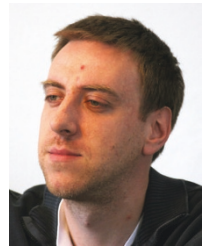
[33] B. Badic, M. Rupp, and H. Weinrichter, "Quasi-orthogonal space time block codes for data transmission over four and eight transmit antennas with very low feedback rate," in *5th International ITG Conference on Source and Channel Coding (SCC)*, pp. 157–164, Erlangen-Nürnberg, 2004.

[34] C. F. Mecklenbräuker and M. Rupp, "Generalized Alamouti codes for trading quality of service against data rate in MIMO UMTS," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 5, pp. 662–675, 2004.



Sebastian Caban was born in Vienna, Austria, on March 13, 1980. Studying electrical engineering at the Vienna University of Technology, he received several scholarships as the best student of the college. During his diploma thesis he worked at the university's Institute of Communications and Radio-Frequency Engineering where he codeveloped the MIMO testbed presented in this paper. Currently he is continuing his study of communication engineering at the University of Illinois at Urbana-Champaign, USA. His research interests focus on rapid prototyping of MIMO systems.

Christian Mehlführer was born in Vienna, Austria, in 1979. In 2004 he received his Dipl.-Ing. degree in electrical engineering from the Vienna University of Technology. Besides his diploma studies he worked part-time at Siemens AG where he performed integration tests of GSM carrier units. After finishing his diploma thesis on implementation and real-time testing of space-time block codes at the Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, he started working on his doctoral thesis at the same institute. His research interests include rapid prototyping, experimental investigation of MIMO systems, the upcoming UMTS MIMO HSDPA mode, and the MIMO extension for WLAN (802.11n).



Robert Langwieser received his Dipl.-Ing. degree in electrical engineering from the Vienna University of Technology, Austria, where he is currently working towards his Ph.D. degree at the Institute of Communications and Radio-Frequency Engineering. He is employed as a Project Assistant and is Member of the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms. The focus of his recent work is the development of a modular radio-front-end for MIMO testbeds. His research interests include radio frequency and implementation aspects of MIMO systems.

Arpad L. Scholtz was born on January 19, 1947 in Kecskemet, Hungary. In 1956 he moved to Austria and received the Austrian citizenship in 1960. He studied telecommunications at the Vienna University of Technology, where he earned his Masters degree in 1972 and the Ph.D. degree in 1976, both with distinction. From 1972 to 1982, Dr. Scholtz worked as an Assistant Professor for radio-frequency engineering at the Institute of Communications and Radio-Frequency Engineering. Since 1982, Dr. Scholtz additionally has been serving at the same Institute as a Lecturer, and in 1992 he was given the title of Associate Professor. Arpad L. Scholtz is Author or Coauthor of more than a hundred scientific publications. He teaches radio-frequency engineering with emphasis on electronic circuit design, antennas, and point-to-point communications. His hobbies are amateur radio and digital photography.

Markus Rupp received his Dipl.-Ing. degree in 1988 from the University of Saarbrücken, Germany, and his Dr.-Ing. degree in 1993 from the Technische Universitaet Darmstadt, Germany, where he worked with Eberhardt Haensler on designing new algorithms for acoustical and electrical echo compensation. From November 1993 until July 1995, he had a postdoctoral position at the University of Santa Barbara, California, with Sanjit Mitra where he worked with Ali H. Sayed on a robustness description of adaptive filters with impacts on neural networks and active noise control. From October 1995 until August 2001, he was a member of the technical staff in the Wireless Technology Research Department, Bell-Labs, where he was working on various topics related to adaptive equalization and rapid implementation for IS-136, 802.11, and UMTS. He is presently a Full Professor of digital signal processing in mobile communications at the Vienna University of Technology. He is Associate Editor of IEEE Transactions on Signal Processing, of EURASIP JASP Journal on Applied Signal Processing, of EURASIP JES Journal on Embedded Systems and is an elected AdCom Member of EURASIP. He authored and coauthored more than 150 papers and patents on adaptive filtering, wireless communications, and rapid prototyping.