



Project Number:	IST-026905
Project Title:	Multiple-Access Space-Time Coding Testbed
Project Coordinator:	C.F. Mecklenbräuker
Deliverable Number:	D2.3.2c

Title of Deliverable:	Concluding report on the ETHZ real-time FPGA testbed
Workpackage:	WP-2
Nature:	R
Dissemination level:	PU
Editor:	Peter Lüthi
Authors:	see list inside
Contractual Date of Deliverable:	Feb. 28, 2009
Actual Date of Delivery:	Feb. 22, 2009

Abstract:

This deliverable reports on the advances of the multi-user (MU) MIMO testbed during the last year of the EU FP6 MASCOT research project. The first part of this report describes the high-level architecture and operating principle of the offline MU-MIMO testbed. The second part is dedicated to the real-time testbed and its extension towards QR-based MU-MIMO detection. The extensions include the successful testbed integration of a QR decomposition ASIC for MIMO preprocessing, the implementation of QR-based MIMO detection schemes such as successive interference cancellation (SIC) and sphere decoding, and the addition of provisions for enabling MU-MIMO communication. Finally, measurement results of both testbeds are presented. MIMO gains and multi-user aspects are demonstrated and measurements of the real-time testbed concerning the implemented extensions for QR-based MIMO detection complete this report.

Contents

1	Offline Testbed Description	5
1.1	Hardware Overview	5
1.2	Operating Principle	6
1.3	Hardware Integrity Tests	7
1.4	Signal Processing	8
2	Real-Time Testbed Extensions	10
2.1	MIMO Preprocessing Extensions	10
2.1.1	System Model	10
2.1.2	System-Level Overview	11
2.1.3	MIMO Preprocessing Architecture	12
2.1.4	Testbed Integration of the MMSE-SQRD ASIC	13
2.1.5	MIMO Preprocessing Performance	17
2.2	MU-MIMO Implementation Aspects	19
2.3	Extensions for MU-MIMO Detection	21
2.3.1	Constellation-Point Normalization	21
2.3.2	Successive Interference Cancellation (SIC)	23
2.3.3	Sphere Decoding	24
2.3.4	Implementation Results	26
3	Measurement Results	28
3.1	Setup	28
3.2	Results	29
3.2.1	MIMO Gains	29
3.2.2	Space-Time Block Codes	31
3.2.3	Transmit Noise	33
3.2.4	Detector Comparison	34
3.2.5	Over-the-Air Measurements	37
4	Conclusion	39

Authors

Peter Lüthi, Markus Wenk, Thomas Koch, Patrick Mächler
Eidgenössische Technische Hochschule Zürich
tel.: +41 44 632 50 08 (Peter Lüthi)
e-Mail: {luethi,mawenk,koch,maechler}@iis.ee.ethz.ch

Executive Summary

The aim of Task 2.3 of the MASCOT project is the development, testing, and demonstration of multi-user (MU) MIMO algorithms on a real-time FPGA-based testbed. This deliverable reports on the final setup, capabilities and results of the MU-MIMO testbed developed within the MASCOT project and demonstrated live at the second ETHZ open house event on February 13, 2009.

The first part of the report focuses on the ETHZ MIMO-OFDM offline testbed, which constitutes a convenient solution for experimenting with new signal processing algorithms and ideas in a floating-point software environment, while still incorporating all real-world wireless signal propagation effects.

The second part of this deliverable is devoted to the latest extensions of the real-time MIMO-OFDM testbed. The extensions include the successful testbed integration of a QR decomposition ASIC for MIMO preprocessing, the implementation of QR-based MIMO detection schemes such as successive interference cancellation (SIC) and sphere decoding, and the addition of provisions for enabling MU-MIMO communication. Furthermore, the system requirements and implementation aspects for MU-MIMO communication are outlined and discussed. Finally, measurement results of both testbeds are presented. MIMO gains and multi-user aspects are demonstrated using the offline testbed. The measurements involve different detection and coding schemes and specific multi-user aspects. The corresponding measurement results are presented and discussed. The individual measurements have either been carried out directly over the air, or by using a channel emulator allowing for application of well-defined channels and control of interference. Transmit noise was identified as a serious impairment for the sphere decoder and a solution using a whitening filter is presented.

Chapter 1

Offline Testbed Description

The motivation for developing the offline testbed was to have a flexible and easy-to-use prototyping platform, where different MU-MIMO algorithms can quickly be assessed over real world channels under realistic conditions (including RF impairments and alike). This is achieved by carrying out most of the signal processing tasks offline in a high-level computing environment such as Matlab for example and where only the time-domain baseband signals are written to the hardware, transmitted and read back.

1.1 Hardware Overview

The offline testbed consists of a BAT board containing a *Xilinx Virtex 4* FPGA and up to four attached Wing boards. Each Wing board includes a full RF chain with analog-digital/digital-analog conversion, mixing, and amplification. [4]. Dedicated transmit and receive buffers and individual configuration registers for each Wing are implemented on the FPGA and can be accessed from the PC through a USB interface. On the PC, a customized software framework in Matlab for user interaction and offline MIMO processing is available.

Fig. 1.1 shows a high-level block diagram of one stream of the offline testbed. Only functions which can not be realized offline because of tight timing restrictions, are implemented as hardware blocks on the FPGA. In total four streams are available on each MIMO testbed terminal. Data to be sent is transferred to the transmit buffer and read out when a transmission is initiated. In receive mode, the frame start detector triggers the start signal for the receive buffer based on the received signal strength indication (RSSI) signal, which activates the receive buffer to record the incoming data.

The automated gain control (AGC) adjusts the signal level of the analog

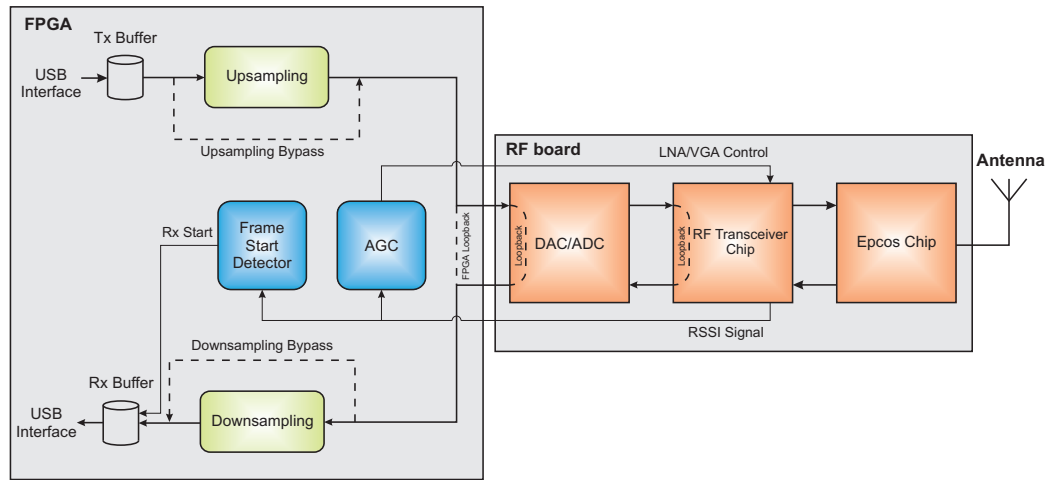


Figure 1.1: Overview of a single stream including loop back and bypass paths for calibration purposes and hardware integrity tests.

baseband signal for optimal analog-to-digital conversion. This is done by controlling the gains of an low noise amplifier (LNA) and a variable gain amplifier (VGA) located on the transceiver chip. Again, the RSSI signal is used as a power indicator.

1.2 Operating Principle

The offline testbed has been designed such that each of the four streams can be used as transmit or receive chain individually. With this architecture, each terminal can be configured for different transmission scenarios (see Fig. 1.2):

Intra-board communication Data transmission from one RF chain to another RF chain on the same board is a helpful feature for running integrity tests over the air and to examine the configuration of the RF transceiver chip as well as the automatic gain control (AGC) and frame start detection.

Inter-board communication Data transmission between selected antennas on one board to another board. For instance, one board can be configured as user in transmit mode and the other as base-station in receive mode. This enables MU scenarios with up to four users on the same testbed terminal.

Multi-user communication Two boards can simultaneously transmit data to a third board, acting as a base-station. In this mode, the two users

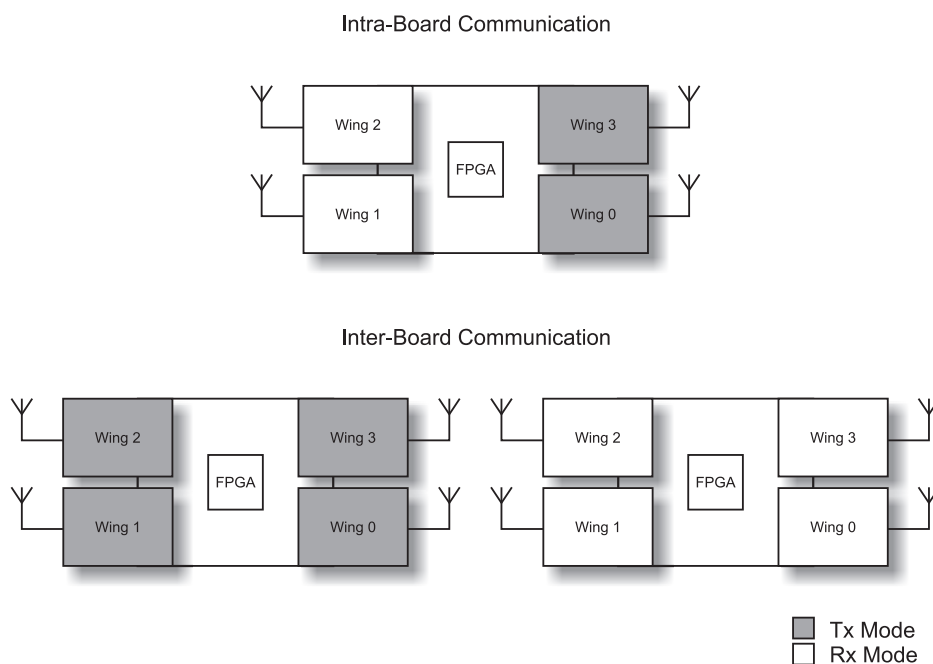


Figure 1.2: Intra-board communication versus inter-board communication

share a common clock source to avoid any frequency offset. An additional cable between those two boards allows to synchronize the start time of the transmission (for more informations see Sec. 2.2).

1.3 Hardware Integrity Tests

The offline testbed provides three different hardware integrity tests (refer to Fig. 1.1 for the location of the loop-back modes):

FPGA loop-back test can be used for the verification purposes of the digital signal processing tasks in the offline testbed as well as in the real-time testbed. The up- and downsampling blocks can be bypassed.

ADC loop-back test includes the ADC for verification of the timing of the receive path at the interface between the ADC and the FPGA.

RF transceiver loop-back test for calibration of the ADC and transceiver chip.

1.4 Signal Processing

Data is transmitted using an OFDM system adapted from the 802.11a standard, as it is implemented on the real-time Testbed. The offline testbed also uses the same frame structure as the real-time testbed including preambles and trainings. The following processing steps are completed in Matlab before the signal can be decoded:

Symbol time synchronization The periodic short preamble is detected by autocorrelation. As soon as the autocorrelation-value begins to drop, the end of the short preamble is found and the start time of the next symbol can be determined.

Channel estimation Each transmit antenna sends a training symbol time-orthogonal to other transmit antennas. A frequency domain ML algorithm is used to determine the channel coefficients for each tone.

SNR estimation The SNR estimation is performed in the time domain. The periodic short preamble is used to estimate the average signal power and its variance. Since one period of the preamble is repeated several times, an estimation of the original signal can be obtained. Deviations of this signal are interpreted as noise and result in a SNR estimation.

Frequency offset estimation and compensation The frequency offset between the clocks of two boards is estimated by the phase offset between two identical sections of the preamble. Afterwards, the whole frame is rotated back with the estimated frequency offset.

Tracking and compensation of residual frequency offset Each OFDM symbol includes four pilot tones, which are used to track the residual frequency offset.

Different MIMO detectors can be chosen, such as a linear MMSE detector or a sphere decoder [3]. They compute either hard decisions or soft decision information. A BCJR-decoder is used if a convolutional encoder was applied at the transmitter.

The whole environment is highly configurable. The number of transmit and receive antennas can be set independently and space-time block codes can be applied. The software determines which RF chains are transmitting and which are receiving and configures them accordingly.

The support of multi-user up-link communication requires that independent data signals are generated for each user and distributed to different

boards. Modulation and coding schemes can be individually chosen for the two users and the receiver must be able to deal with different modulations on different streams.

The software is controlled by a graphical user interface (GUI), shown in Fig. 1.3. A scattering plot and the estimated channel coefficients of each received frame is also shown. This allows to detect errors quickly and gives a first feedback on the quality of the link.

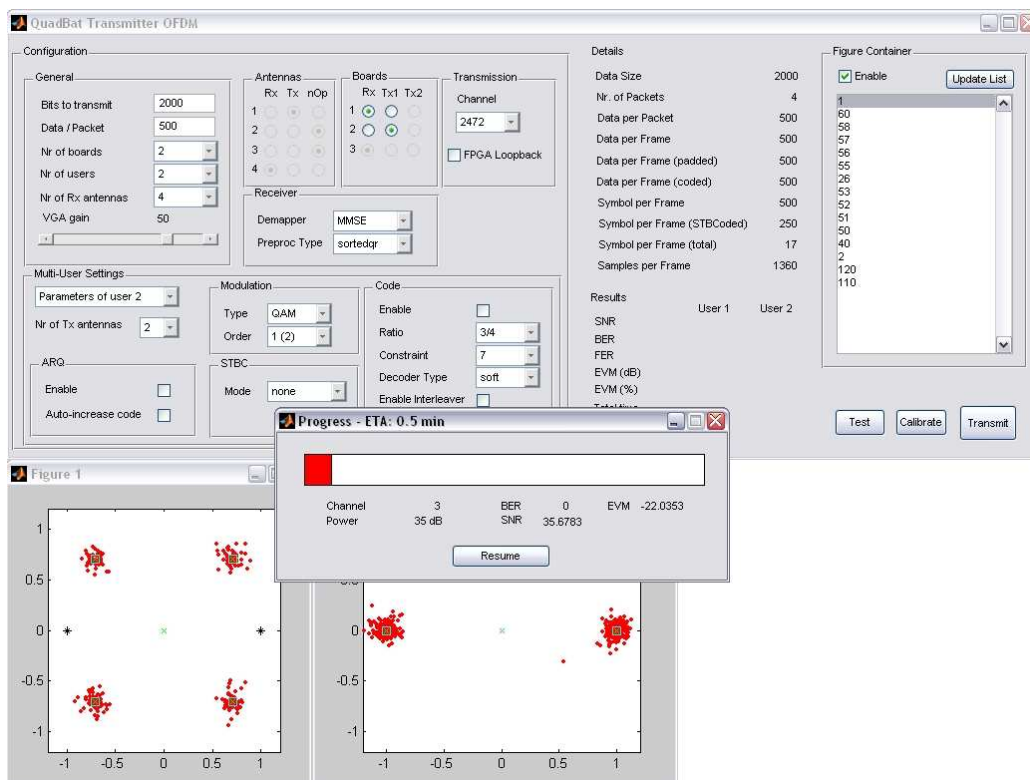


Figure 1.3: GUI of the Offline Testbed

Chapter 2

Real-Time Testbed Extensions

2.1 MIMO Preprocessing Extensions

2.1.1 System Model

A MIMO communication system can be modeled as

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \quad (2.1)$$

with the M_R -dimensional received signal vector \mathbf{y} , the transmitted symbol vector \mathbf{s} is of dimension M_T , the complex-valued channel matrix denoted by $M_R \times M_T$ dimensional matrix \mathbf{H} , and \mathbf{n} representing the M_R dimensional additive zero-mean i.i.d. complex Gaussian noise with variance N_o per complex dimension.

Applying the QR decomposition to \mathbf{H} and multiplying by \mathbf{Q}^H leads to

$$\hat{\mathbf{y}} \triangleq \mathbf{Q}^H \mathbf{y} = \mathbf{R}\mathbf{s} + \mathbf{n} \quad (2.2)$$

where \mathbf{Q} is of dimension $M_T \times M_R$ and unitary and \mathbf{R} is of dimension $M_T \times M_T$ and upper triangular.

A MIMO detector estimates the transmitted symbol vector based on the received signal \mathbf{y} and the channel matrix \mathbf{H} . The estimate $\hat{\mathbf{s}}$ denotes an estimate of the transmitted symbol vector according to the underlying detection algorithm. Finally, the estimated symbol vector $\hat{\mathbf{s}}$ is de-mapped to binary-valued estimates of $\hat{\mathbf{s}}$ as follows

$$\hat{\mathbf{b}} = \text{map}^{-1}(\hat{\mathbf{s}}) \quad (2.3)$$

by, e.g., using Gray mapping.

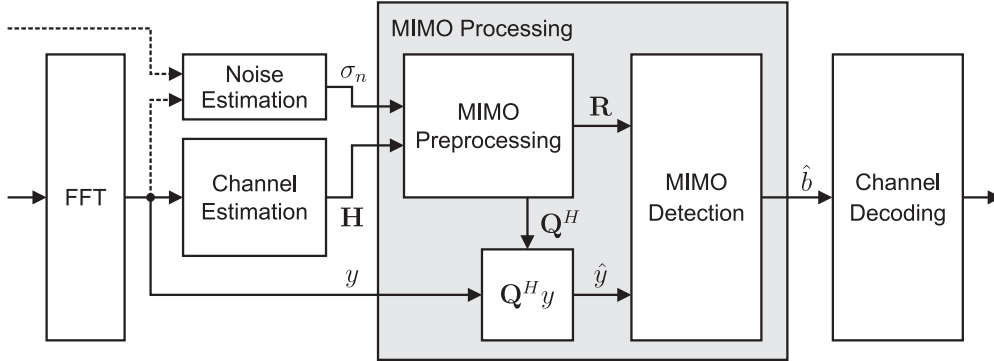


Figure 2.1: System-level overview of the MIMO processing block in the receiver consisting of a QRD-based MIMO preprocessing block and a corresponding MIMO detection unit.

2.1.2 System-Level Overview

A system-level overview of the restructured and extended MIMO processing block in the receiver part of the MASCOT real-time 4×4 MIMO-OFDM testbed is shown in Fig. 2.1. The redesigned MIMO processing block consists of a preprocessing unit which performs the QR decomposition (QRD) and a QRD-based MIMO detection unit. The QRD-based MIMO preprocessing allows for a variety of MIMO detection schemes ranging from linear detection to advanced detection methods, such as the sphere decoder, for example.

The MIMO preprocessing unit computes the *minimum mean squared error sorted QR decomposition* (MMSE-SQRD) of the complex-valued channel matrix \mathbf{H} and passes the result, i.e., the unitary matrix \mathbf{Q}^H and the upper-right triangular matrix \mathbf{R} , to the corresponding detection blocks. The input data for the MIMO preprocessing unit, i.e., the channel matrix \mathbf{H} and the standard deviation σ_n of the noise N_o , are estimated in dedicated channel and noise estimation blocks, respectively.

Upon frame reception, the training data from the frame header are first processed by the channel and noise estimation blocks. The estimated channel matrix \mathbf{H} and the noise term σ_n are then passed to the MIMO preprocessing unit, which performs the MMSE-SQRD of \mathbf{H} . Once the MIMO preprocessing unit has completed its computation, the received (and buffered) symbol vector \mathbf{y} is processed by pre-multiplying \mathbf{Q}^H , i.e., the hermitian-transpose of \mathbf{Q} . The resulting vector $\hat{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$ is then passed to the MIMO detection block. Finally, the detected bits $\hat{\mathbf{b}}$ are handed over to the channel decoder in order to compute estimates of the transmitted bits.

2.1.3 MIMO Preprocessing Architecture

The architecture of the MMSE-SQRD-based MIMO preprocessing block is depicted in Fig. 2.2, showing the top-level block diagram of a combined FPGA-ASIC realization. The entire MIMO preprocessing block consists of three major parts:

First, the entity *MIMO Preprocessing Top* provides the system-level interconnection to the MIMO physical (PHY) layer. As primary input, it retrieves the estimated channel matrices from the channel estimation block in the MIMO receiver – once the training data has been successfully processed after frame reception. The fixed-point data of the channel estimation is properly pre-scaled and passed upward to the next major block by the *Channel Estimation Memory Read* units. Next to the memory read units, the QR-decomposed data is retrieved from the upper block by the *SQRD Memory Write* units. The retrieved SQRD data is post-scaled and individually written to the preprocessing memories, namely the *Config*, *Q* and *R* memories. During the memory write sequence, a specific memory address translation scheme is applied in order to optimize the arrangement of the SQRD data in the preprocessing memories for the subsequent MIMO detection process. As primary outputs of the MIMO preprocessing top-level block, a dual-channel memory interface to the MIMO detection unit provides sufficient memory bandwidth for the detection process.

The second major block, i.e., the FPGA-ASIC interface top-level depicted as entity *SQRD IF Top* in Fig. 2.2, incorporates the individual ASIC data load and retrieval units with dedicated provisions to satisfy the challenging FPGA-ASIC interface timing. Moreover, a specifically designed built-in self-test (BIST) checks and ensures the integrity of the FPGA-ASIC interconnect with respect to functionality and timing.

The third major block of the MIMO preprocessing top-level architecture is constituted by the two MMSE-SQRD ASICs, designed as dual-core variant of the Givens-rotations-based MMSE-SQRD architecture as published in [10]. The employed MMSE-SQRD ASIC, which is depicted in Fig. 2.3, is manufactured in UMC 0.18 μm 1P/6M CMOS technology and the corresponding VLSI implementation results are reported in [11]. Each core of the ASIC is able to perform regularized (i.e., according to the MMSE criterion) or non-regularized QR decomposition, either unsorted, one-time sorted, or iteratively sorted. All computations are carried out in fixed-point data format. The column-norm computation employs the ℓ^2 -norm. Additionally, various configurations of number of transmitted streams and number of receive antennas are supported by the ASIC. The algorithmic specifications of the implemented MIMO preprocessing block and the corresponding ASIC

implementation results are summarized in Tbl. 2.1.

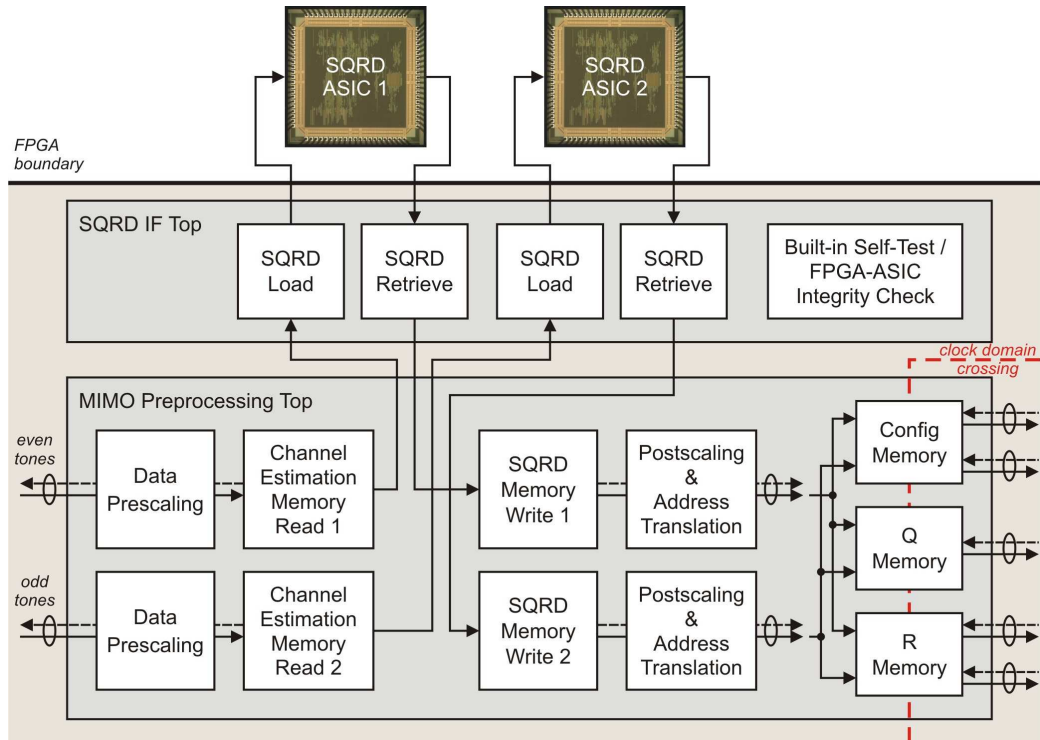


Figure 2.2: Top-level diagram of the MIMO preprocessing unit, which is split into three major parts: PHY logic (*MIMO Preprocessing Top*), FPGA-ASIC interface logic (*SQRD IF Top*), and MMSE-SQRD ASIC processing (*SQRD ASIC 1 & 2*).

2.1.4 Testbed Integration of the MMSE-SQRD ASIC Printed Circuit Board

The electrical and functional correct integration of the MMSE-SQRD ASIC in the MASCOT real-time testbed required the design and manufacturing of a dedicated printed circuit board (PCB) as shown in Fig. 2.4. The customized PCB connects two instances of the MMSE-SQRD ASIC to one of the FPGAs on the VAMP prototyping board [12]. Moreover, the PCB was designed to connect two instances of a video digital-to-analog converter (DAC) to the FPGA. These video DACs are designed for the industrial-standard and widely-used VGA interface specification and are meant to be used in the testbed as versatile debugging infrastructure for different real-time aspects

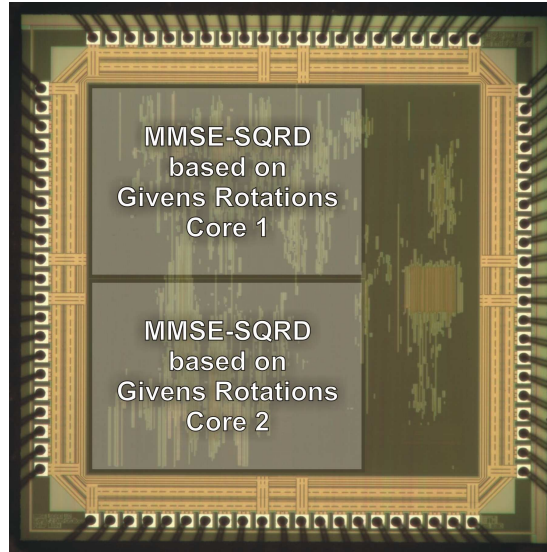


Figure 2.3: Chip micrograph of the MMSE-SQRD ASIC containing two SQRD cores. The ASIC architecture is designed to support simultaneous operation of both SQRD cores. The circuit has been integrated in UMC 0.18 μm 1P/6M CMOS technology.

Table 2.1: MIMO preprocessing specifications

Algorithm	Givens-rotations-based SQRD
QR decomposition	non-regularized, regularized (MMSE)
Column norm computation	ℓ^2 -norm
Sort metrics	minimum, second minimum [9]
Sort methods	none, one-time, iterative
Supported configurations	$N_{TX} \in \{1, 2, \dots, 4\}$ $N_{RX} \in \{1, 2, \dots, 4\}$
Implementation	MMSE-SQRD ASIC [10], manufactured in UMC 0.18 μm 1P/6M CMOS technology
Clock frequency	designed for 152 MHz ^a
Core area	$2 \times 0.785 \text{ mm}^2$
Aggregated processing time per matrix [4 \times 4 MMSE-SQRD]	40 cycles, 263 ns
Aggregated preprocessing throughput [4 \times 4 MMSE-SQRD]	3.8 M SQRD/s

^apost-layout clock frequency

of the MIMO PHY layer and as visualization resource for demonstration purposes. Last but not least, the different integrated circuits on the PCB ask for two different supply voltages provided by two separate power supply modules. The specifications of the MMSE-SQRD ASIC PCB are summarized in Tbl. 2.2.

Table 2.2: Specifications of the MMSE-SQRD ASIC PCB

Devices	2× MMSE-SQRD ASIC [10] 2× Video DAC with VGA support (ADV7125)
Power supply	3.3 V ASIC pad and video DAC power supply 1.8 V ASIC core power supply
FPGA interface	ASIC: 2× 68 I/O signals Video DAC: 2× 18 I/O signals

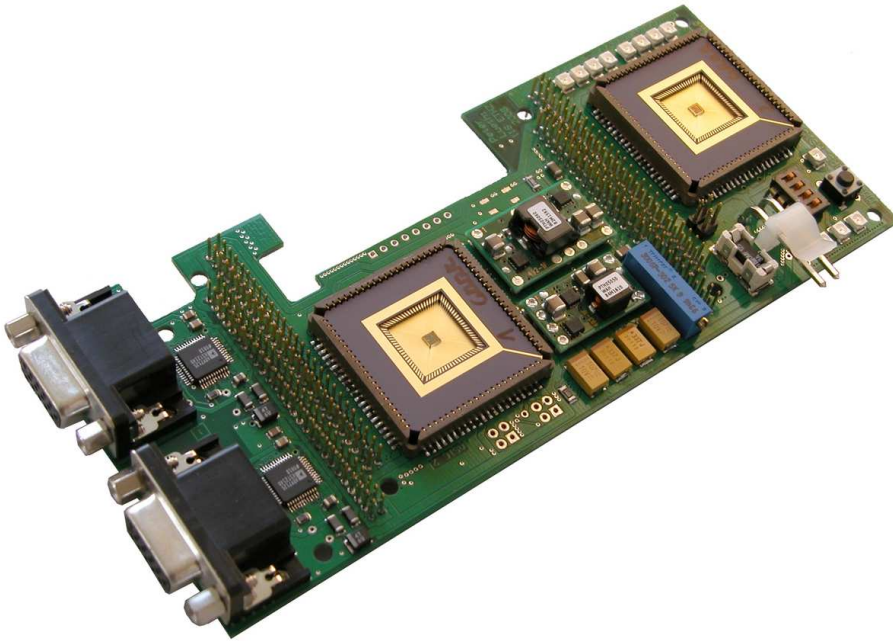


Figure 2.4: Custom-designed PCB for integration of the MMSE-SQRD ASIC in the MASCOT real-time testbed.

System-Level Integration Challenges

The system-level architecture of the MIMO preprocessing block described in Sec. 2.1.3 was designed for a target clock frequency of 80 MHz. The 80 MHz

clock was chosen as reasonable compromise between aggregated preprocessing performance and implementation feasibility with respect to I/O interface timing closure. But still, choosing a design frequency of 80 MHz continued to pose several technical challenges, which had first to be accomplished before obtaining a fully operational system.

The hand-shake protocol applied for the I/O interface between FPGA and ASIC requires tight timing constraints in order to guarantee flawless operation. Using 3.3 V supply voltage for the pads, the ASIC has input and output pad delay figures of approximately 1.0 ns, and the FPGA has input and output pad delays of 1.0 ns and 1.9 ns, respectively, all of them specified for standard switching characteristics. Unfortunately, the proper implementation of the underlying hand-shake protocol specification asks for a direct input-to-output path, i.e., a purely combinational input-to-output relation, on the receiving part of the interface. In the ASIC, this path exhibits an acceptable delay of 2.5 ns, in the FPGA, the incurred delay can be significantly higher depending on the synthesis and place and route (P&R) results. The weak spot of direct input-to-output relations for practical I/O realizations is exhibited when pad and interconnect delays considerably come into account for high-speed links. Moreover, with direct input-to-output relations for an interface protocol, there is conceptually no head-room reserved for I/O timing relaxation using local clock skew tricks.

The technical challenge in obtaining post-P&R timing closure is clearly illustrated with the following consideration: At 80 MHz clock frequency, the clock period amounts to 12.5 ns. The direct input-to-output path in the ASIC requires 2.5 ns of the interface timing budget, and twice 0.5 ns are spent for the interconnection delay on the PCB. The aggregated FPGA pad delay asks for another 2.9 ns of the timing budget. At the end, the remaining 6.1 ns must account for the entire FPGA logic and FPGA-internal P&R delays and clock skews. Furthermore, the electrical aspects of slew rate control and drive strength matching must also be properly handled.

A viable solution – if not the only one – for accomplishing all mentioned technical challenges is the strict control of the placement process for the FPGA logic in order to enforce the proper placement of critical components. The subsequent routing process still influences the outcome of the final design, but does not contribute significantly.

The MIMO preprocessing top-level architecture has conceptually foreseen a fall-back option with respect to timing closure: Since the entire MIMO preprocessing block forms an isolated clock domain, it allows for specific clock speed reductions in case increased FPGA placement and routing congestion deteriorate the critical FPGA-ASIC interface timing and thus rendering the FPGA system-level timing closure with the target clock speed impossible.

There exist two clock domain boundaries relevant for the MIMO preprocessing top-level: The first one resides in the channel estimation memory located inside the channel estimation block depicted in Fig. 2.1, the second one crosses the preprocessing memories located at the output side of the MIMO preprocessing top-level, illustrated as dashed line in Fig. 2.2.

Deployment

The practical deployment of the MIMO preprocessing block has been carried out using a *LeCroy WaveRunner 204 MXi* oscilloscope in combination with a *LeCroy MS-500* mixed-signal extension, shown in Fig. 2.5. This sophisticated and powerful technical solution greatly alleviated the burden imposed by the challenging task of achieving FPGA-ASIC interface timing closure at the target clock frequency of 80 MHz.

2.1.5 MIMO Preprocessing Performance

The implemented MIMO preprocessing block employs two MMSE-SQRD ASICs clocked at 80 MHz and achieves an aggregated preprocessing throughput of four million matrices per second. The processing time for 48 complex-valued 4×4 matrices using MMSE-SQRD-based preprocessing amounts to $13.75 \mu\text{s}$. The system-level characteristics of the implemented MIMO preprocessing block are summarized in Tbl. 2.3.

The MIMO preprocessing block can start its operation as soon as four subcarriers are completely estimated and available as matrices in the channel estimation memory. From a system-level perspective, this early start procedure allows to reduce the MIMO preprocessing latency by $2.54 \mu\text{s}$. As a result, the MMSE-SQRD-based preprocessing in the MASCOT real-time testbed essentially introduces an effective MIMO preprocessing latency of $11.2 \mu\text{s}$.

Table 2.3: System-level specifications of MIMO preprocessing

Algorithm	4×4 MMSE-SQRD
Implementation	$2 \times$ MMSE-SQRD ASIC [10]
Clock frequency	80 MHz
Aggregated preprocessing throughput [4×4 MMSE-SQRD]	4 M SQRD/s
Effective system-level preprocessing latency (48 complex-valued 4×4 matrices)	$11.2 \mu\text{s}$

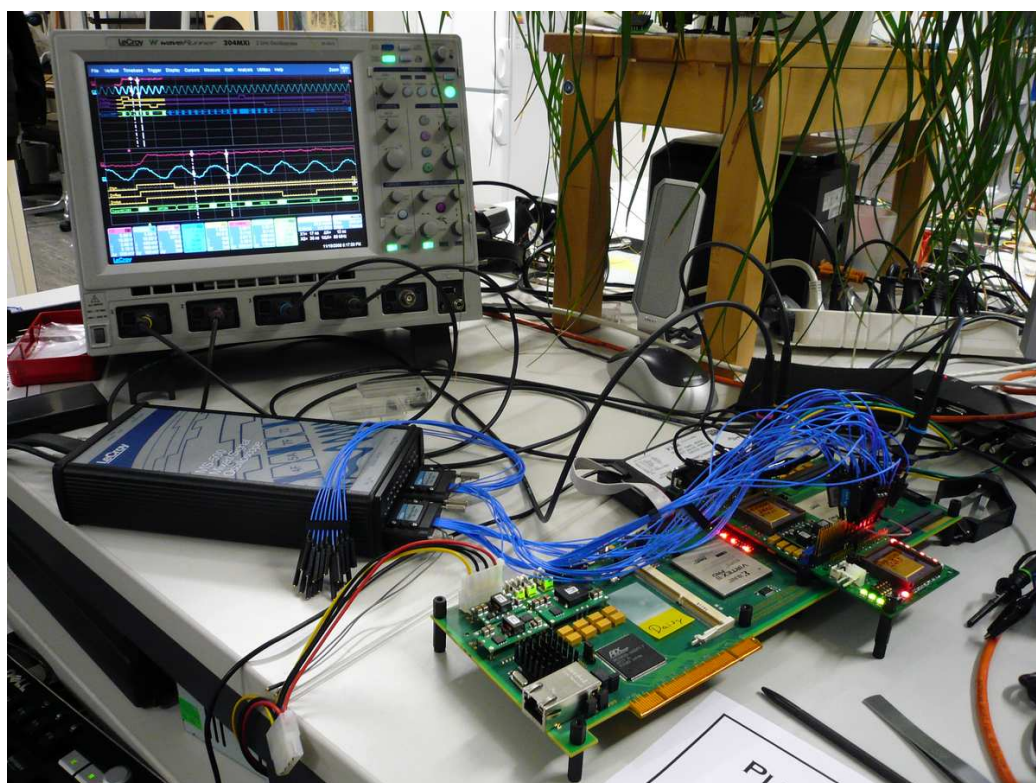


Figure 2.5: Workspace showing the *LeCroy WaveRunner 204 MXi* oscilloscope and the *LeCroy MS-500* mixed-signal extension for assessing electrical and timing aspects of the FPGA-ASIC interface during system-level deployment of the MIMO preprocessing block.

2.2 MU-MIMO Implementation Aspects

The implementation of MU-MIMO communication differs in several aspects compared to single-user MIMO communication [8]. It also has to be distinguished between the up-link or multiple access channel and the down-link or the broadcast channel as the challenges and requirements for the two scenarios differ.

With the real-time testbed, we decided to focus on the up-link. The realization of a testbed being able to do simultaneous MU-MIMO up-link communication involves the following tasks and challenges:

- The PHY layer must be configurable to select on how many and on which of the four streams the data are to be transmitted.
- The receiving base station must be able to handle simultaneously transmitted signal fields containing modulation scheme and frame length from each user.
- The receiving base station must be able to simultaneously compute and check the CRC of each user's data stream.
- All users need to be synchronized such that their signal arrives at the same time at the base station.
- The frequency offset and sampling rate offset must be compensated in the transmit path as the base station is not able to compensate different frequency offsets in the receive path.
- The PHY layer in the base station must signal all simultaneously received frames to the MAC layer.
- To follow the paradigm "high-complexity in the base-station, low-complexity in the user terminals", more involved detection algorithms for the base station need to be implemented (e.g. ML-detection) and evaluated.

MU-MIMO Configuration Additional configuration registers were added to the PHY layer and additional logic was necessary to be able to select on how many streams and on which streams data are transmitted. The following PHY layer blocks need to be configured:

- Interleaving over antennas needs to be turned off.

- The transmit buffer (PHY-MAC layer interface) must be configured such that data are only transmitted on the selected streams. For example, one user needs to transmit on stream 1 and stream 2, while the other user transmits on stream 3 and stream 4. This is necessary to assure that the MIMO training is received in the correct way.
- In the RF front-end, only the used RF chains shall be enabled.
- The receive buffer needs to be configured such that it knows, which receive stream belongs to which user. This is necessary to compute the cyclic redundancy check (CRC) of each user and signal the results to the MAC layer.

User Synchronization and Frequency-Offset Compensation In general, the carrier frequency of each user differs slightly, as each user is equipped with an own local oscillator. In a collision-based up-link scenario, frequency offset estimation and compensation becomes more challenging and differs from the single-user scenario. In a single-user scenario, the frequency offset between transmitter and receiver is estimated and compensated in the receiver. However, in a MU up-link scenario, there are several users with different local oscillators transmitting at the same time. Therefore, estimation and compensation of all frequency offsets in the receiver becomes more challenging or even impossible as a superposition of the transmitted signal is observed. One solution to counter this problem is to estimate the frequency offset in the receiver and compensate the frequency offset before the signal is actually transmitted. However, this approach requires a major redesign of the synchronization block in the real-time testbed. Therefore and in order to keep complexity manageable, we decided to attach all users to a common clock generator. Thus, the frequency offset and sampling rate offset between the users and the base station is the same for all users. This setup is shown in Fig. 2.6.

To realize simultaneous up-link communication, all users must transmit within a certain time interval, such that the multi-path propagation from each user reaches the base station within the guard interval. This synchronization is a difficult task and the authors in [13] state that timing synchronization represents the most challenging task in multimedia wireless scenarios, especially true, in a completely asynchronous up-link transmission. In the real-time testbed, this is solved by having a counter in the transmitter of each user that starts counting, when a new frame is received from the base station. In the meantime, the MAC layer is processing the received frame and prepares the response. Once the counter has expired, the frames are

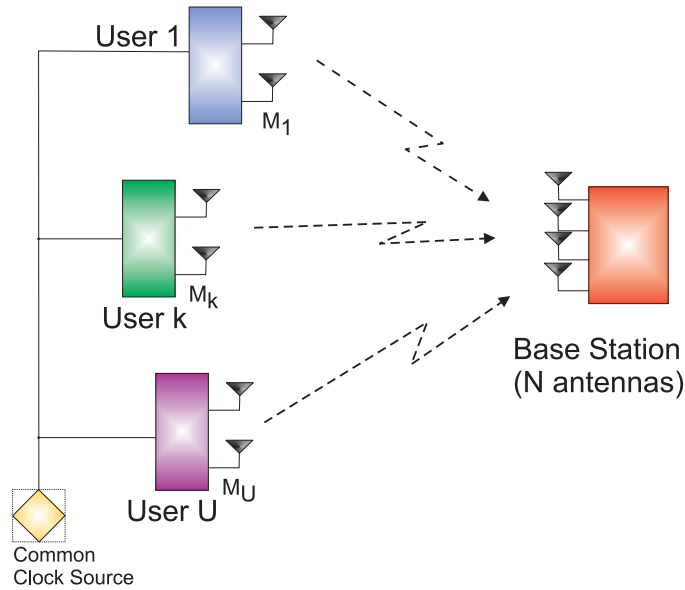


Figure 2.6: Transmitter synchronization

transmitted.

2.3 Extensions for MU-MIMO Detection

A MU-MIMO detector differs from a single-user detector that it must be capable to process different modulation schemes on different streams used by each user. So far, the Riccati-based MMSE detector used in the MIMO testbed was only able to process the same modulation scheme for all streams. Recently, a successive interference cancellation (SIC) detector and a sphere decoder (SD) were implemented and added to the testbed. The two detectors are designed to process different modulation schemes on each stream. In the following, the FPGA implementation of the two detectors as well as the integration into the entire system will be explained.

2.3.1 Constellation-Point Normalization

The implementation of a MIMO detector being able to process different modulation schemes on each stream introduces additional computation complexity. As the constellation points of each modulation scheme are chosen such that the average transmit power equals one, the decision regions need to be adjusted in the receiver according to the used modulation scheme on each stream. To avoid this additional effort in the timing-critical detection pro-

cess, it is advantageous, if each modulation scheme is a subset of the highest order modulation scheme, i.e., if the following relation holds:

$$\mathcal{O}_{\text{BPSK}} \subset \mathcal{O}_{\text{QPSK}} \subset \mathcal{O}_{16\text{-QAM}} \subset \mathcal{O}_{64\text{-QAM}} \quad (2.4)$$

To fulfill (2.4), the constellation points of QPSK and 16-QAM have to be scaled such that they are a subset of the 64-QAM constellation points. Furthermore, the 64-QAM constellation points are scaled to fall onto the grid of Gaussian integers, i.e., $\mathcal{O}_{64\text{-QAM}} = \{\pm 1 \pm i, \pm 3 \pm 3i, \pm \dots\} \in (\mathbb{CZ})$. For BPSK, an elegant solution is to scale and rotate the BPSK constellation points onto the QPSK constellation set. The rotation angle is 45 degrees, which can be implemented efficiently in hardware by interchanging real and imaginary part and taking care of the signs. The scaling and rotating process of constellation points can be described in the following form:

$$\tilde{\mathbf{s}} = \mathbf{C}\mathbf{B}\mathbf{s} \quad (2.5)$$

where \mathbf{B} is a real-valued diagonal matrix containing the scale factors for each stream and \mathbf{C} is a complex-valued diagonal matrix containing the appropriate rotation values. Note that this transformation needs to be compensated according to:

$$\hat{\mathbf{y}} = \mathbf{R}\mathbf{B}^{-1}\mathbf{C}^{-1}\tilde{\mathbf{s}} + \mathbf{n} \quad (2.6)$$

Since \mathbf{C}^{-1} rotates columns of \mathbf{R} , the diagonal elements of $\mathbf{R}\mathbf{B}^{-1}\mathbf{C}^{-1}$ are no longer guaranteed to be real-valued and positive. To maintain this important property, the received vector can be multiplied by the rotation matrix \mathbf{C} such that

$$\mathbf{C}\hat{\mathbf{y}} = \mathbf{C}\mathbf{R}\mathbf{B}^{-1}\mathbf{C}^{-1}\tilde{\mathbf{s}} + \mathbf{C}\mathbf{n} \quad (2.7)$$

and $\mathbf{C}\mathbf{R}\mathbf{B}^{-1}\mathbf{C}^{-1} = \tilde{\mathbf{R}}$ has real-valued non-negative entries on the diagonal.

The entries of \mathbf{B} and \mathbf{C} depend only on the modulation schemes being used on each stream. Therefore, $\tilde{\mathbf{Q}}^H = \mathbf{C}\mathbf{Q}^H$ and $\tilde{\mathbf{R}}$ can be computed in the preprocessing stage. Since both, \mathbf{B} and \mathbf{C} are diagonal matrices, each element of $\tilde{\mathbf{Q}}^H$ and $\tilde{\mathbf{R}}$ is computed according to

$$\tilde{Q}_{ij}^H = C_{ii}Q_{ij}^H, \quad i, j = 1, 2, \dots, M_T \quad (2.8)$$

$$\tilde{R}_{ij} = \frac{C_{ii}}{C_{jj}B_{jj}}R_{ij}, \quad i, j = 1, 2, \dots, M_T \quad (2.9)$$

The latency introduced by this additional preprocessing step in the current architecture is negligible (i.e., 6 clock cycles, 75 ns), but simplifies the implementation of a MIMO detector significantly and enables to process different modulation schemes on different streams.

2.3.2 Successive Interference Cancellation (SIC)

The input-output relation in (2.2) can be formulated according to

$$\hat{y}_i = \sum_{j=i}^{M_T} R_{ij} s_j + n_i \quad (2.10)$$

with R_{ij} being the ij th element of \mathbf{R} . Successive interference cancellation (SIC), also known as decision feedback equalization (DFE), performs

$$\hat{x}_i = \frac{\hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij} \hat{s}_j}{R_{ii}} \quad (2.11)$$

$$\hat{s}_i = Q(\hat{x}_i) \quad \text{for } i = M_T, M_T - 1, \dots, 1. \quad (2.12)$$

The function $Q(\cdot)$ describes the slicing operation that chooses the constellation point \hat{s}_i closest to \hat{x}_i . In the hardware implementation, the division by R_{ii} can be avoided by taking it into the slicing operation. The SIC iteration now becomes

$$R_{ii} \hat{x}_i = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij} \hat{s}_j \quad (2.13)$$

$$\hat{s}_i = Q(R_{ii}, R_{ii} \hat{x}_i) \quad \text{for } i = M_T, M_T - 1, \dots, 1 \quad (2.14)$$

The slicing operation can also be formulated as a minimization problem:

$$\hat{s}_i = \arg \min_{s_i \in \mathcal{O}} |b_i - R_{ii} s_i|^2 \quad \text{where } b_i = \hat{y}_i - \sum_{j=i+1}^{M_T} R_{ij} \hat{s}_j. \quad (2.15)$$

SIC Architecture The main task of the SIC detector is to compute (2.15) and return the constellation point, that minimizes (2.15). In (2.15), it can be observed that the term b_i is the same for all possible s_i at iteration step i . Therefore, this term can be computed in advance. The partial distance unit (PDU) shown in Fig. 2.7 computes b_i and is responsible for the closest constellation point. The closest symbol is found by checking a finite set of decision boundaries consisting of multiples of R_{ii} . By exploiting the symmetry of the constellation points, this task can be implemented with 6 comparators (3 for each dimension).

Fig. 2.7 shows the high-level block diagram of the implemented SIC detector. To complete the entire iteration, the PDU needs to be used M_T times. Afterwards, the detected symbol vector is reordered according to the permutation matrix \mathbf{P} delivered by the QR-decomposition and finally de-mapped into single bits.

Implementation results of the SIC detector can be found in Tbl. ??.

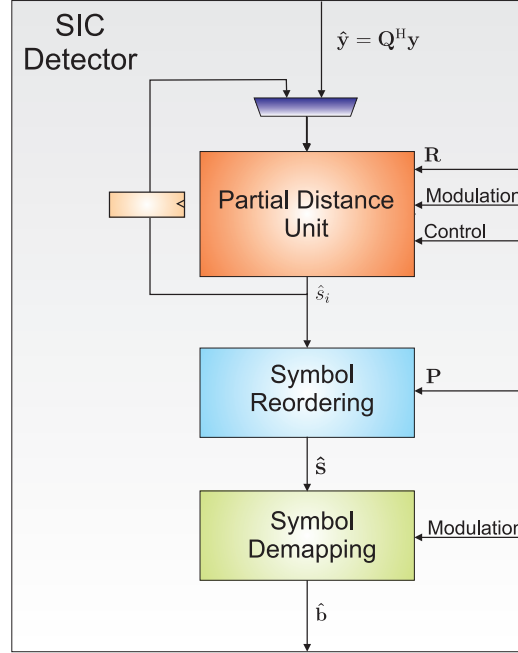


Figure 2.7: High-level block diagram of the SIC detector implementation

2.3.3 Sphere Decoding

Sphere decoding (SD) [7] can be illustrated as tree search problem, where each branch of the tree is associated with a metric and the distance of a node is the sum of all branch metrics along that path. The goal of a tree-search-based detector is to find the leaf node with the smallest Euclidean distance. Tbl. 2.4 summarizes this analogy:

Table 2.4: Analogy between tree search and SD algorithm.

Tree-search	Sphere algorithm
layer	transmit stream j
path to node on layer j	$\hat{\mathbf{s}}^{(j)} = [\hat{s}_j, \hat{s}_{j+1}, \dots, \hat{s}_{M_T}]$
branch metric $e(\hat{\mathbf{s}}^{(j)})$	$\hat{y}_i - \sum_{j=i}^{M_T} R_{ij} \hat{s}_j$
node distance T_i	$\sum_{j=i}^{M_T} e(\hat{\mathbf{s}}^{(j)}) = T_{i+1} + w_i$

The SD algorithm searches the tree in a depth-first manner. Starting at the top of the tree, the child node with the smallest partial Euclidean distance (PED) is computed and selected. Afterwards, the child node with

the smallest PED of this selected node is computed and selected. This is continued until the first leaf node with its associated Euclidean distance (ED) is reached. The path to the first leaf node also corresponds to the SIC solution.

The SD algorithm now takes this ED as new radius r into the sphere constraint

$$T_i(\hat{\mathbf{s}}^{(i)}) < r^2$$

which means, that all nodes whose PED violate this constraint can be pruned. The algorithm now continues with the next node that fulfills the sphere constraint. Whenever the algorithm finds a leaf node with a smaller ED than the current radius, r is replaced by this value. The algorithm continues until either all nodes are visited or pruned.

The main properties of the SD algorithm can be summarized as follows:

- Achieves maximum likelihood (ML) performance
- Average search complexity is strongly reduced compared to an exhaustive search ML detector
- Variable run-time

System Requirements In the real-time testbed, 48 data subcarriers need to be processed within 4 μs . With a clock frequency of 80 MHz, this makes 6.67 clock cycles in average per sub-carrier. Simulations showed that a sphere decoder with early termination and scheduling needs about 13 clock cycles in average to achieve acceptable BER performance. Therefore, at least two SD units need to be available on the real-time testbed.

Sphere Architecture Fig. 2.8 shows a high level block diagram of the sphere decoder implementation. The chosen architecture is 5 times interleaved pipelined, which means that 5 symbols from different subcarriers are processed at the same time. This allows to run the sphere decoder at the system frequency of 80 MHz. The first three stages are responsible for computing the PED and carry out the metric enumeration one layer above in parallel. The fourth pipeline stage carries out the addition with the previous PED and the last stage is responsible for the radius check and the input handling. Once a symbol is found, it needs to be reordered according to the permutation matrix computed by the QR-decomposition ASIC.

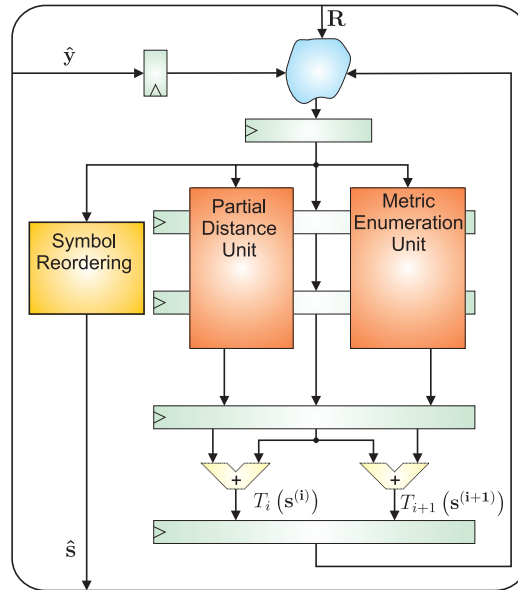


Figure 2.8: High-level block diagram of the sphere detector implementation.

System Level Architecture Fig. 2.9 shows the system level architecture of the sphere decoder implementation. Two sphere cores are instantiated to meet the throughput requirements. The two cores are fed by a common control unit which takes a new symbol out of the FIFO buffer whenever a sphere core has finished the computation of one MIMO symbol. The processed symbols are stored according to their sub-carrier in another buffer. This scheduling method is called FIFO scheduling and balances the variable run-time of the SD algorithm. Furthermore, the control unit is able to abort the computation of a symbol in order to meet the real-time requirement. Aborted symbols get a pseudo-soft output value as described in [5].

2.3.4 Implementation Results

Tbl. 2.5 summarizes the achieved implementation results and compares the SIC detector and the sphere decoder. The presented results include all instantiated cores, control logic and input scheduling. It can be observed, that the implementation resources of SIC on the real-time testbed are about 4 times less than for sphere decoding. Performance measurements and comparison between the detectors are shown in Sec. 3.2.4.

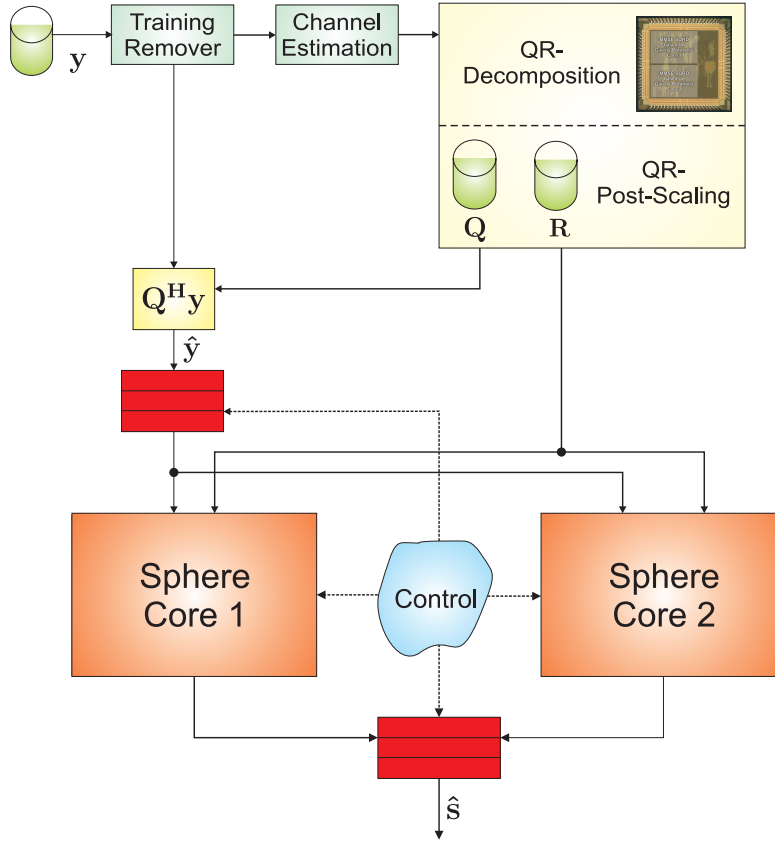


Figure 2.9: High-level block diagram of the sphere detector implementation.

Table 2.5: Figures of merit for the SIC detector and the sphere decoder, both including the normalization of the constellation points implemented on a XILINX Virtex-II Pro VP70 FPGA with speed grade -5.

	SIC Detector	Sphere Decoder
Slices	1748/33088 (~ 5%)	7967/33088 (~ 24%)
MULTs	33/328 (~ 10%)	75/328 (~ 23%)
BRAMs	8/328 (~ 3%)	14/328 (~ 5%)
Latency [clock cycles]	16	variable
Clock frequency [MHz]	80	80

Chapter 3

Measurement Results

3.1 Setup

The measurement of point-to-point MIMO communication is done with two terminals, each equipped with four RF chains. One of the two terminals serves as transmitter, the other one as receiver. For MU-MIMO measurements, three terminals are used, two of them are configured as transmitters with two antennas each, while the other terminal serves as receiver.

The measurements can either be carried out on the off-line testbed, where all the signal processing is done in Matlab or on the real-time testbed, where everything is processed in real-time and the statistics is tracked in the MAC layer. The results presented in the following were mostly measured on the off-line testbed. The reason is the easier setup of the measurement infrastructure. Nevertheless, all measurements can also be carried out on the real-time testbed.

Channel Emulator

In order to perform reproducible measurements and to be able to apply channels from defined models, a channel emulator is used. This emulator can model MIMO channels up to four streams and impulse responses with up to 12 taps. The channel emulator accepts RF signals which are internally down-converted and digitized. The channel model is applied in the digital domain before the signal is mixed to RF again. In our measurement setup, we use a block fading TGn type C channel model [6]. The channel coefficients are precomputed and changed after each frame (i.e., we measure a block fading scenario).

The RF chains are connected to the channel emulator by cables. The channel emulator is controlled by the PC via the Ethernet (Fig. 3.1). This

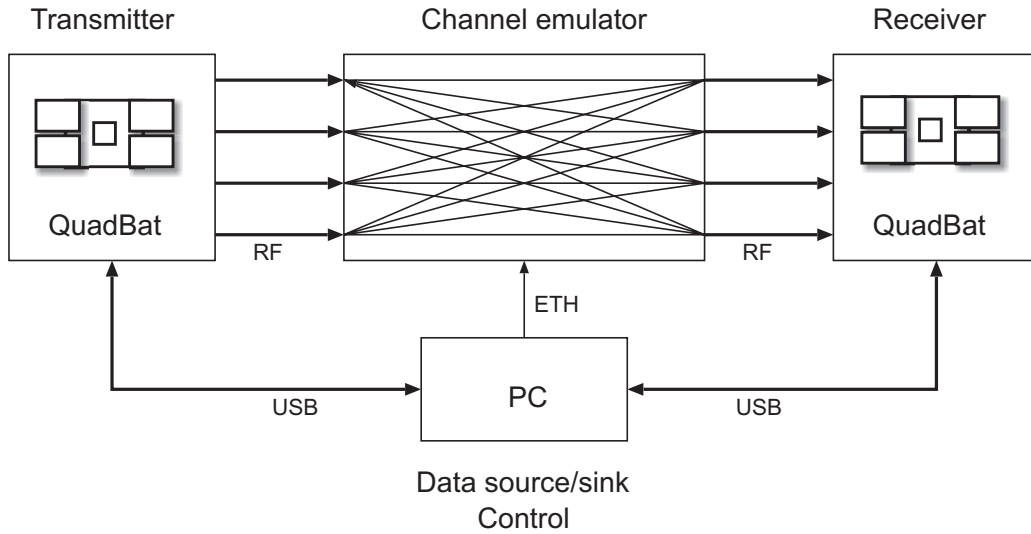


Figure 3.1: Measurement setup overview

allows to remotely change channel realizations and to sweep over a range of output powers to measure the performance at different SNRs.

3.2 Results

We measure the bit error rate (BER) while sweeping the output power of the channel emulator from -61 dBm to -26 dBm and average over at least 200 channel realizations. The first plot (Fig. 3.2) shows a 4×4 MIMO transmission with a rate $1/2$ convolutional code, de-mapped with a soft sphere decoder. For each output power level, one can estimate the SNR at the receiver and plot the BER with respect to this SNR, which is shown in Fig. 3.2(b). At higher output power, the amplifiers of the channel emulator and the receiver may switch to stages with higher noise figures, which explains non-continuous behaviour of the BER curve.

3.2.1 MIMO Gains

In this section, we want to assess the benefits of diversity, array, and multiplexing gains achievable with a real-world MIMO system.

The easiest way to obtain diversity is to increase the number of receive antennas. In this case, we also benefit from an array gain, but no multiplexing gain. The effect of introducing one or three more receivers can be seen in Fig. 3.3.

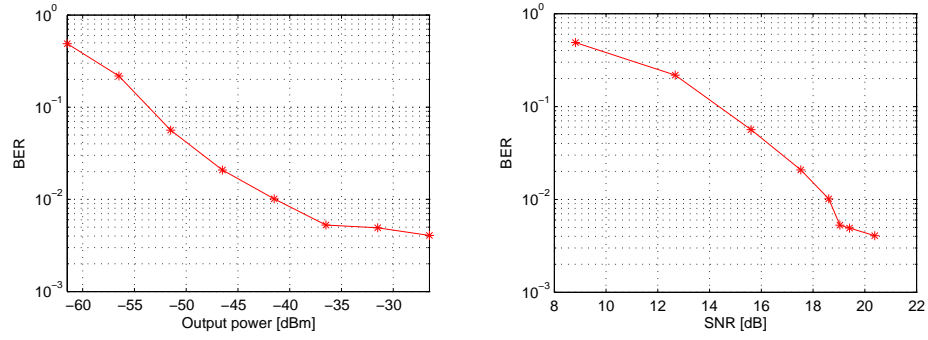


Figure 3.2: 4x4 coded MIMO transmission (QPSK, MMSE detector)

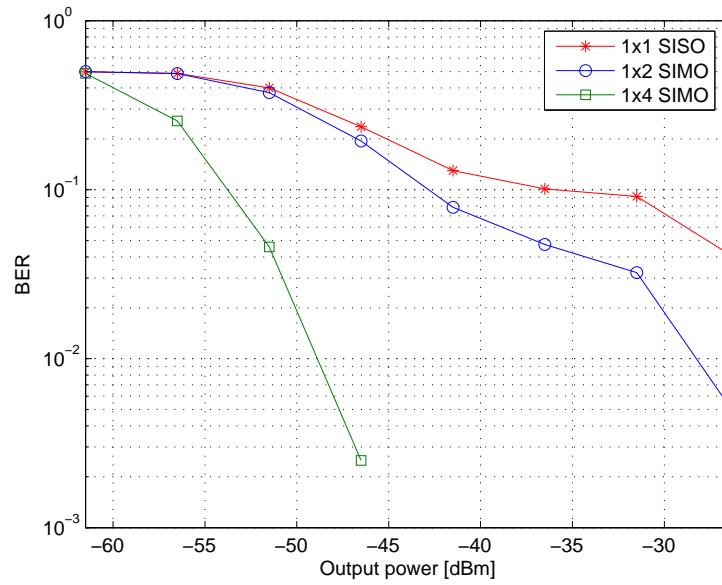


Figure 3.3: Receive diversity (QPSK, uncoded)

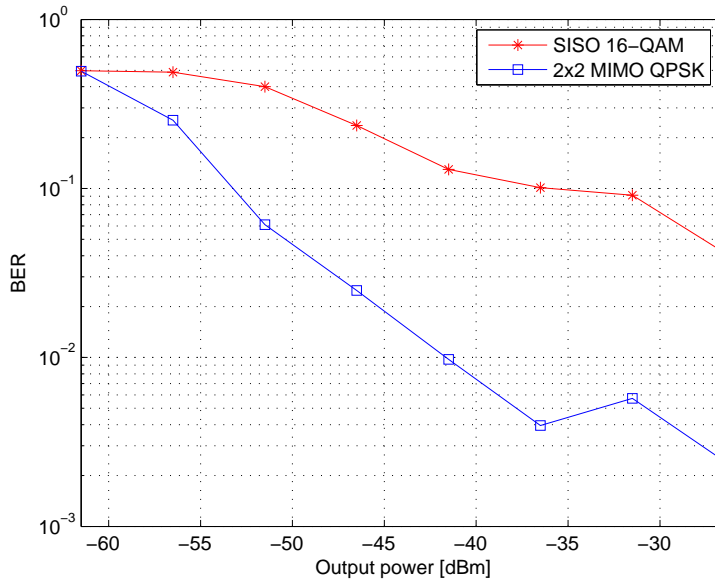


Figure 3.4: BER performance for coded 2×2 MIMO compared to SISO

A well-known result stating that it is more efficient to use more degrees of freedom than to pack many bits in few dimensions [14] can be observed by comparing 16-QAM SISO to QPSK 2×2 MIMO transmissions. Fig. 3.4 shows the significantly better performance of the MIMO system using a sphere decoder. The doubled slope indicates that the diversity gain can be exploited.

3.2.2 Space-Time Block Codes

Alamouti Scheme

The STBC coding scheme introduced by Alamouti [1] allows to obtain transmit diversity without increasing the decoder complexity. Fig. 3.5 shows a comparison of 2×1 Alamouti with a simple SISO system. The transmit power is individually adjusted such that both schemes consume the same amount of power. The Alamouti coded signal shows better performance with a gain of about 5 dB compared to SISO.

It is possible, to some extent, to use the Alamouti scheme for MU system users. Each user applies the Alamouti code to its two antennas. The resulting code is not orthogonal anymore, which renders MMSE detection suboptimal. Fig. 3.6 shows the BER for two users, one using QPSK and the other one using BPSK modulation.

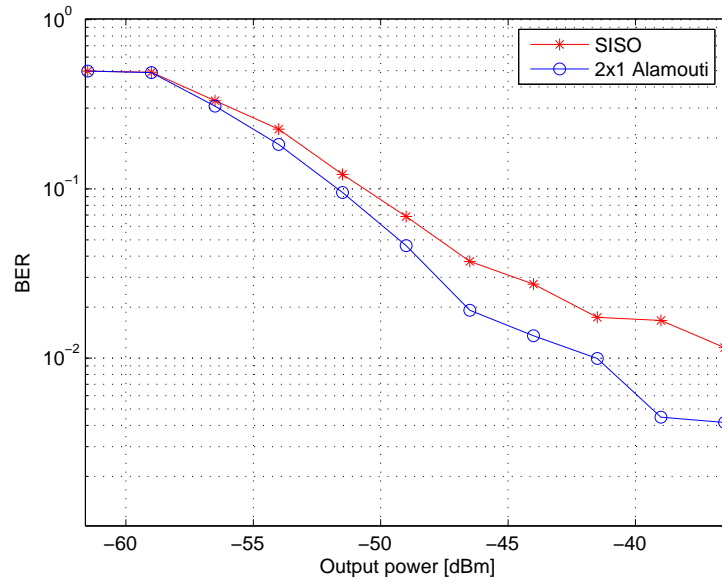


Figure 3.5: Comparison of Alamouti code with SISO (uncoded, QPSK)

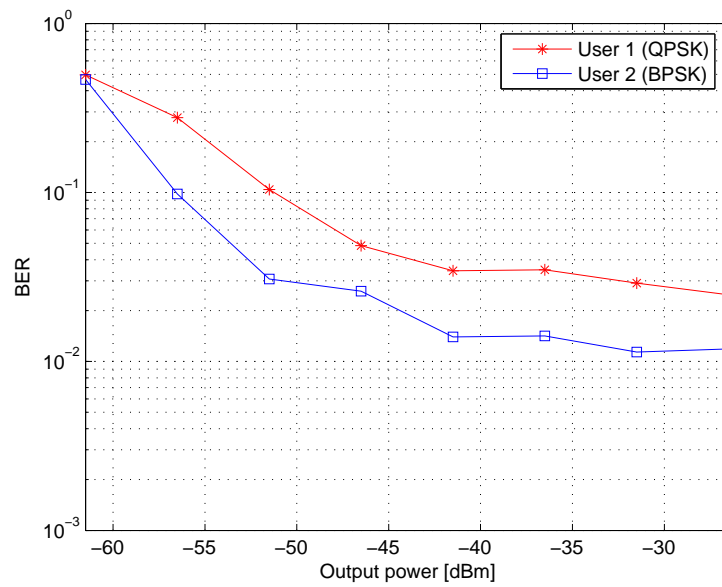


Figure 3.6: Two user coded Alamouti scheme (MMSE detector)

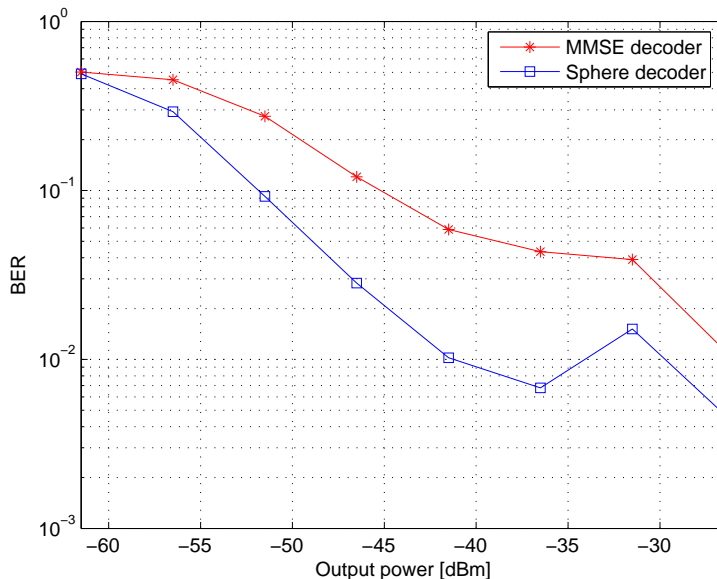


Figure 3.7: Golden code (QPSK, coded)

Golden Code

The Golden code is a full-rate STBC for a 2×2 system [2]. Unlike the Alamouti code, the Golden code requires an ML decoder to achieve its full performance. This is shown in Fig. 3.7 where a MMSE detector is compared to a sphere decoder.

3.2.3 Transmit Noise

Transmit noise was found to be a major impairment in MIMO systems, but has not yet obtained much attention in the literature. This noise is generated in the RF chain of the transmitter. The system model must be adapted in the following way to include transmit noise \mathbf{n}_t as well as receive noise \mathbf{n}_r .

$$\mathbf{y} = \mathbf{H}(\mathbf{s} + \mathbf{n}_t) + \mathbf{n}_r \quad (3.1)$$

As a first approximation, \mathbf{n}_t can be modeled as zero-mean i.i.d. complex Gaussian noise $\mathbf{n}_t \sim \mathcal{CN}(0, \mathbf{I}_t \sigma_t^2)$ and $\mathbf{n}_r \sim \mathcal{CN}(0, \mathbf{I}_r \sigma_r^2)$. The transmit noise SNR of our RF chain was estimated to be around 27 dB.

In contrast to the receive noise, transmit noise is not white anymore at the receiver. This leads to a significant performance degradation of systems that rely heavily on the white noise assumption.

In order to mitigate the effects of colored noise, a whitening filter can be used at the receiver. The received noise is $\mathbf{n} = \mathbf{H}\mathbf{n}_t + \mathbf{n}_r$ with mean $\mathcal{E}\{\mathbf{n}\} = 0$ and covariance matrix

$$\mathcal{E}\{\mathbf{n}\mathbf{n}^H\} = \mathbf{K} = \mathbf{H}\mathbf{H}^H\mathbf{I}_t\sigma_t + \mathbf{I}_r\sigma_r.$$

Using a Cholesky decomposition, an upper triangular matrix \mathbf{R} can be computed such that

$$\mathbf{R}^H\mathbf{R} = \mathbf{K}.$$

By multiplying the channel matrix and the received vector with the inverse of \mathbf{R}^H , we get

$$(\mathbf{R}^H)^{-1}\mathbf{y} = (\mathbf{R}^H)^{-1}\mathbf{H}(\mathbf{s} + \mathbf{n}_t) + (\mathbf{R}^H)^{-1}\mathbf{n}_r$$

with white noise

$$\mathbf{R}\mathbf{n}_t + (\mathbf{R}^H)^{-1}\mathbf{n}_r \sim \mathcal{CN}(0, \mathbf{I}).$$

3.2.4 Detector Comparison

Fig. 3.8 shows the Matlab frame error rate (FER) simulation results of SIC, soft-MMSE and hard sphere decoding with and without 27.5 dB transmit noise. It can be observed that the sphere decoder suffers significantly from transmit noise, while the MMSE detector is much less sensitive on transmit noise.

Fig. 3.9 shows measurements of the frame error rates of MMSE detectors and sphere decoders with hard and soft output. Each decoder is measured with and without noise whitening filter. In accordance with the Matlab simulations in Fig. 3.8, the sphere decoder is shown to be much more sensitive on transmit noise.

Real-time testbed detection performance Fig. 3.10 shows the detection performance measured on the real-time testbed between sphere decoding and SIC for different modulation schemes. This measurement was carried out on the TGn-C channel model using the channel emulator. The measurement setup and the MAC layer were configured such that 1000 frames were transmitted for each channel realization. In total, 200 different channel realizations were used. The output power of the channel emulator was swept in order to get different SNR values. It can be observed, that sphere decoding clearly performs better than SIC detection. Furthermore, 64-QAM modulation does not achieve acceptable FER performance. It is presumed, that this is mostly due to the analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC). The ADC and DAC are only 10 bits wide. In order to be able

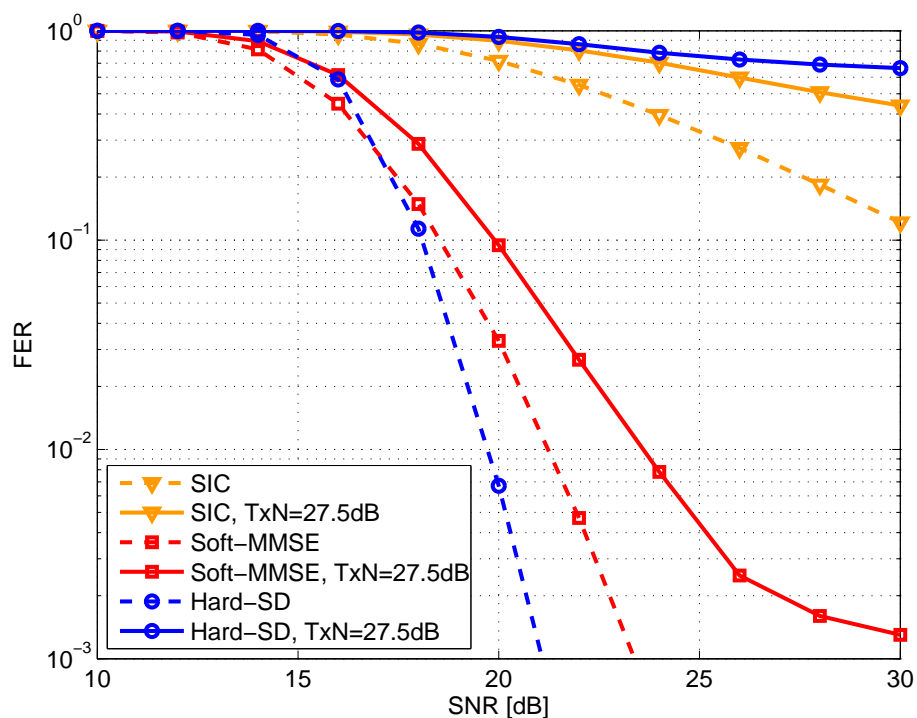


Figure 3.8: Simulation comparison of the detection performance with 27.5 dB transmit noise and without transmit noise (16-QAM, rate 1/2 coded).

to transmit 64-QAM with a 10-bit ADC, automatic gain control needs to be very accurate. The necessary number of bits in the ADC and more accurate algorithms for the AGC in order to optimally control the received signal need to be further evaluated.

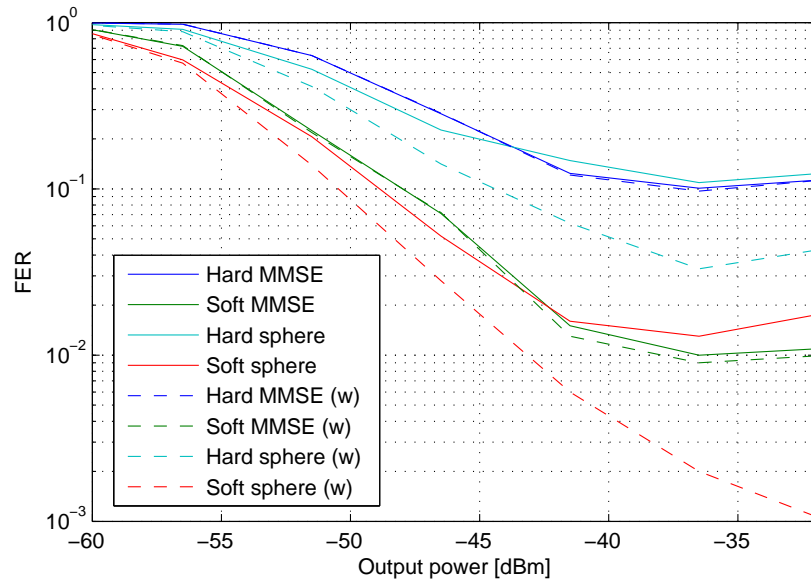


Figure 3.9: Comparison of MIMO detectors for 4×4 system, with and without whitening filter (QPSK, rate 1/2 coded).

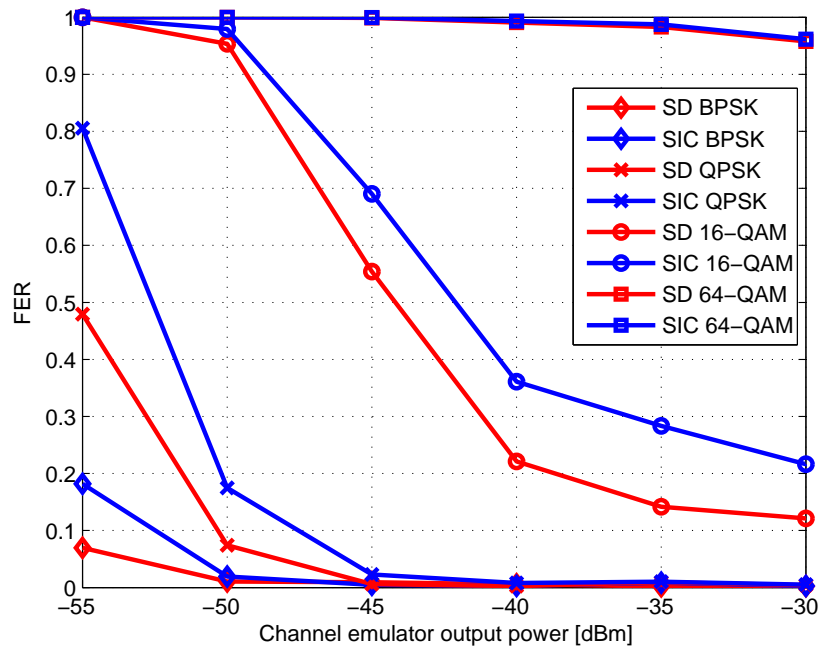


Figure 3.10: Comparison of SIC and sphere decoding on the real-time testbed.

3.2.5 Over-the-Air Measurements

For over-the-air measurements, the channel emulator of the previous setup is replaced by antennas. The measurements are carried out in an office environment by sweeping the transmit power.

Fig. 3.11 shows a 4×4 MIMO transmission with a rate $3/4$ convolutional code and compares it to a 2×2 MIMO transmission. It can be seen that the MIMO gain could be exploited in this channel.

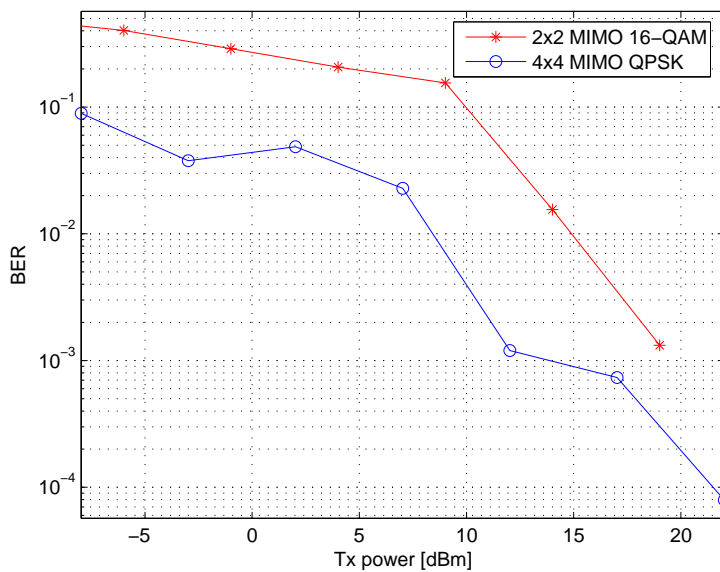


Figure 3.11: Over-the-air measurement results (QPSK, coded, MMSE detector)

Multi-User MIMO

The following MU-MIMO setup is measured here: Two users send simultaneously to one base station. The two users have a common clock source and can synchronize the start of their transmission by a cable (cf. Fig. 2.6). This ensures that there is only one frequency offset between users and base station, which can be compensated at the base station. The two users transmit with two antennas each and four antennas are required at the receiver.

This setup is compared to a time division multiple access (TDMA) 2×4 scheme, where the two users are transmitting alternately. To achieve the same data rate, the collision-based and the TDMA schemes are operated with QPSK and 16-QAM, respectively.

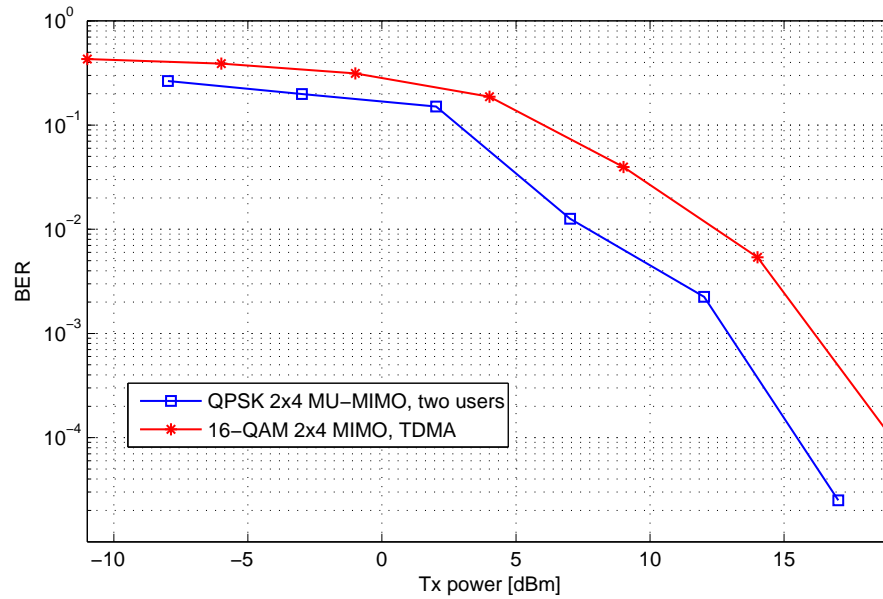


Figure 3.12: MU-MIMO measurement over the air (rate 1/2 coded)

The measurements in Fig. 3.12 show that the collision-based approach achieves a smaller BER and thus a higher throughput is possible. The transmit power was normalized across the entire system, i.e., the simultaneously transmitting users can send at only half the power of TDMA users. The better performance of the collision-based uplink stems from the MIMO gain achieved by doubling the number of transmit antennas.

Chapter 4

Conclusion

This report presented three major contributions derived from the ETHZ testbed activities during the last year of the MASCOT project.

First, the hardware and software setup of the ETHZ MIMO-OFDM offline testbed and its operating principle have been described. The offline testbed represents a convenient solution for experimenting with new ideas and algorithms, while still incorporating real-world wireless signal propagation effects. All signal processing tasks are performed in Matlab with floating-point precision. Only the completely prepared time-domain transmit signals are sent to the hardware and transmitted over the air or across the channel emulator. At the receiving side, the received time-domain signals are first stored in a hardware buffer in real-time and – after complete reception – are read out from Matlab. All receive signal processing is then carried out in Matlab. The offline testbed can be used for both, point-to-point and multi-user experiments.

Second, the testbed integration of a sorted QR decomposition (SQRD) ASIC and the implementation of QR-based SIC detection and sphere decoding in the real-time testbed have been described. The challenges of the testbed deployment of the SQRD ASIC have been addressed, as well as the architectural aspects and system requirements of the QR-based MIMO detection blocks. The integration of QR-based MIMO preprocessing in the real-time testbed allows for comparing different MIMO detection schemes in a real-time environment with real-world signal propagation. Furthermore, the implementation aspects for simultaneous multi-user MIMO uplink communication have been described, as well as the corresponding implemented hardware extensions for the real-time testbed.

Third, several measurement results acquired from both, offline testbed and real-time testbed, have been presented. The measurement results are in accordance with theoretical findings. Unfortunately, the sphere decoding

algorithm seems to be very sensitive on systematic transmit-side signalling impairments. These impairments can be modelled as transmit noise in the MIMO system equation. In the presence of transmit noise, the detection performance of the sphere decoder degrades significantly. This effect has been confirmed by dedicated Matlab simulations. Essentially, we became aware of the importance of the commonly neglected transmit noise through real-world performance assessment of the sphere decoder in the ETHZ real-time testbed.

Bibliography

- [1] Siavash Alamouti. A simple transmit diversity technique for wireless communications. *IEEE J. Sel. Areas Commun.*, 16(8):1451–1458, October 1998. [31](#)
- [2] Jean-Claude Belfiore, Ghaya Rekaya, and Emanuele Viterbo. The Golden code: A 2x2 full-rate space-time code with nonvanishing determinants. *IEEE Trans. Inf. Theory*, 51(4):1432–1436, April 2005. [33](#)
- [3] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei. VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE Journal of Solid-State Circuits*, 40(7):1566–1577, July 2005. [8](#)
- [4] A. Burg, P. Luethi, and M. Wenk. Deliverable D2.3.1: Hardware extension and MAC upgrade for MIMO testbed, April 2007. IST-026905 MASCOT. [5](#)
- [5] A. Burg, M. Wenk, and W. Fichtner. VLSI Implementation of Pipelined Sphere Decoding with Early Termination. In *Proceedings of the European Signal Processing Conf.*, September 2006. [26](#)
- [6] V. Erceg and et al. *TGn channel models*. IEEE 802.11-03/940r4, May 2004. [28](#)
- [7] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471, April 1985. [24](#)
- [8] D. Gesbert, M. Kountouris, R.W. Heath, Chan-Byoung Chae, and T. Salzer. Shifting the MIMO paradigm. *Signal Processing Magazine, IEEE*, 24(5):36–46, Sept. 2007. [19](#)
- [9] C. Hess, M. Wenk, A. P. Burg, P. Luethi, C. Studer, N. Felber, and W. Fichtner. Reduced-complexity MIMO detector with close-to ML

- error rate performance. In *Proc. Great Lakes Symposium on VLSI*, pages 200–203, March 2007. [14](#)
- [10] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner. VLSI implementation of a high-speed iterative sorted MMSE QR decomposition. In *IEEE Int. Symp. on Circuits and Systems*, pages 1421–1424, New Orleans, May 2007. [12](#), [14](#), [15](#), [17](#)
- [11] P. Luethi, C. Studer, S. Duetsch, E. Zraggen, H. Kaeslin, N. Felber, and W. Fichtner. Gram-Schmidt-based QR decomposition for MIMO detection: VLSI implementation and comparison. In *IEEE Asia Pacific Conf. on Circuits and Systems*, pages 830–833, Macao, China, November 2008. [12](#)
- [12] P. Luethi, M. Wenk, and D. Wagner. Deliverable D2.3.2a: Report on MIMO MAC extensions, December 2007. IST-026905 MASCOT. [13](#)
- [13] M. Morelli, C.-C.J. Kuo, and M.-O. Pun. Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review. *Proceedings of the IEEE*, 95(7):1394–1427, July 2007. [20](#)
- [14] D. Tse and P. Viswanath. *Fundamentals of wireless communication*. Cambridge University Press, 2005. [31](#)