Communicating Process Architectures – 2003
Jan F. Broenink and Gerald H. Hilderink (Eds.)
IOS Press, 2003

A CSP-based Processing Architecture for a Flexible MIMO-OFDM Testbed

H.S. CRONIE, F.W. HOEKSEMA and C.H. SLUMP
University of Twente, Department of Electrical Engineering, Mathematics and Computer Science (EEMCS), Signals and Systems Group, P.O. box 217 - 7500 AE Enschede - The Netherlands
email: h.s.cronie@student.utwente.nl, {f.w.hoeksema,c.h.slump}@utwente.nl

Abstract. Future wireless communication systems require novel techniques to increase the bitrate, coverage and mobility. One of these techniques is spatial multiplexing and we have investigated the use of a CSP-based kernel in the implementation of a spatial multiplexing testbed. It turns out that the use of the CSP-based kernel not only provides a good way of system modeling, but also provides a very scalable software architecture for the testbed. In future, we can change several system parameters without changing the software architecture. With the testbed we were able to verify the concept of spatial multiplexing in an office environment.

1 Introduction

Future wireless applications, such as wireless LANs, demand faster and more reliable communication links. Wireless LANs are typically used in an indoor environment. The indoor radio channel suffers from multipath propagation, which is caused by objects between the transmitter and receiver. Multiple components of the transmitted signal are received. The situation is illustrated in figure 1. These components interfere with each other and the in-

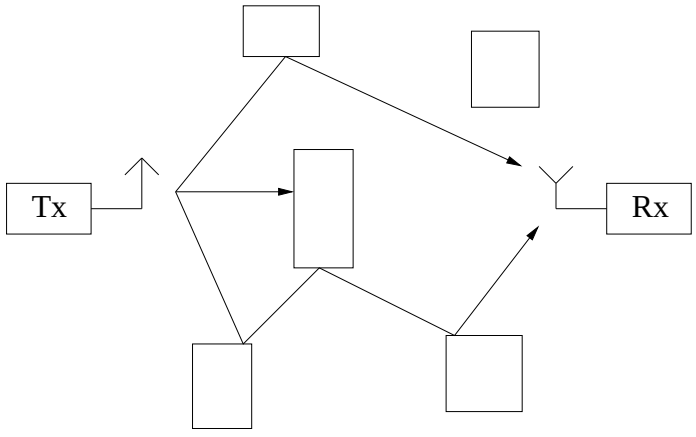

Figure 1: Multipath propagation between transmitter and receiver

terference can be constructive or destructive. The case of destructive interference is called a signal fade and its effect is that it causes a reduction in received signal power. The channel also varies with time[1] and the communication system has to be designed properly to cope

[1]The coherence time of the indoor channel at 2.4 GHz is around 20 ms.

with the negative effects of multipath fading. Recent research has shown that it is also possible to use the negative effect of signal fading in our advantage [1], [2]. For this, multiple transmit antennas are used at the transmitter and multiple receive antennas at the receiver. The idea is that fades become uncorrelated[2] for antenna separation $> \lambda/2$, in which λ is the wavelength of the radio signal. We transmit an independent datastream from each transmit antenna. At each receive antenna a linear combination of the signals transmitted is received. The input-output relation of the *Multiple Input Multiple Output* (MIMO) indoor fading channel corrupted by noise, can be written as:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

in which $\boldsymbol{n}$ is a complex gaussian noise vector added by the channel. The complex vector $\boldsymbol{x}$ contains the symbols transmitted. These symbols are drawn from a complex constellation. At the receiver, we estimate the channel transfer matrix $\boldsymbol{H}$ and detect the data by a process called *space-time decoding*. The effect is that we are able to increase the amount of data that can be transmitted and communication becomes more reliable, while keeping the bandwidth and total transmit power constant.

The datarate achievable on a communication link is bounded by its information theoretic capacity. The use of a MIMO communication system results in an increased channel capacity. Figure 2 shows the behavior of the capacity if both the number of transmit antennas N_t and the number of receive antennas N_r is increased. As can be seen from the figure, the possible

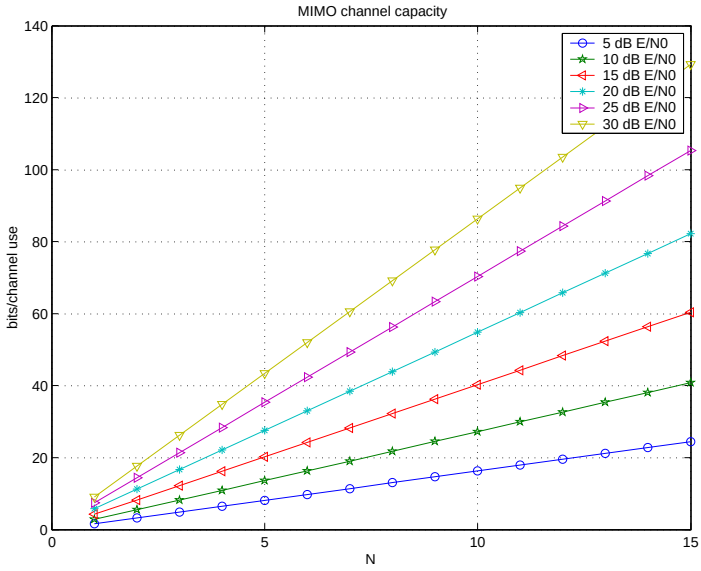

Figure 2: The capacity as function of the amount of antennas used ($N = N_t = N_r$)

gains are enormous. The capacity grows linear with the amount of transmit and receive antennas used. The received data however, has to be detected. For this, the channel transfer matrix $\boldsymbol{H}$ has to be known and a detection algorithm has to be developed which uses this matrix. To transmit information, a modulation method is needed and research has shown that *Orthogonal Frequency Division Multiplexing* (OFDM) is a suitable method in a fading environment [7]. To investigate the practical behavior of a MIMO communication system, we have developed a MIMO-OFDM testbed.

[2]Fades do not necessary have to be uncorrelated, but the possible gains are higher for uncorrelated fades.

2 Functional Testbed Architecture

In our research we address the question whether the use of a MIMO channel lives up to its promises. Claims, often based on models, have to be tested in real-world environments. We want to investigate under what conditions the channel model is valid. Moreover, other points of interest are the required processing capability for the different algorithms and the design of the computational platform for this type of systems (including HW/SW partitioning). We have build a testbed to answer these questions. Figure 3 shows an overview of the testbed. Both the transmitter and receiver use multiple antennas. The interesting signal processing

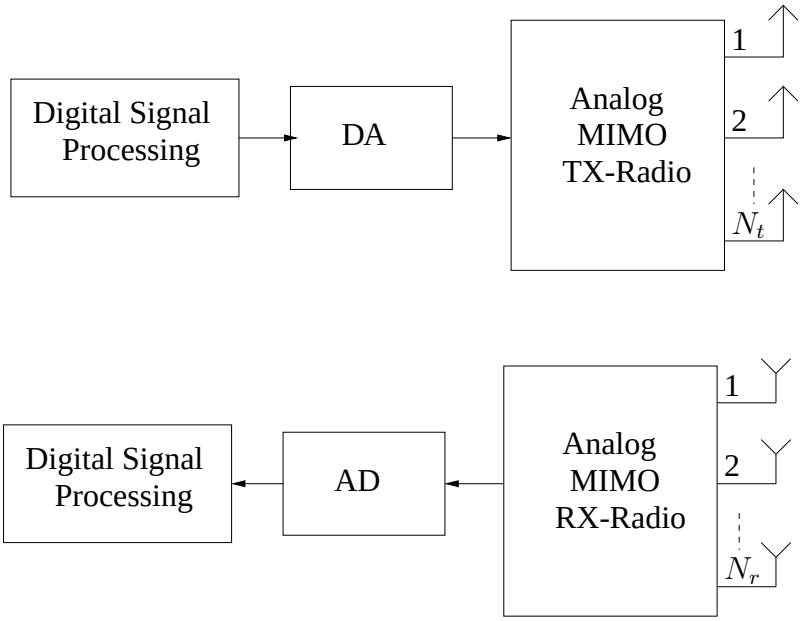

Figure 3: The MIMO-OFDM testbed

takes place at the receiver. First, analog processing converts the RF signal to a baseband signal. The baseband signal is sampled and the subsequent digital signal processing functions have to be implemented at the receiver:

- *OFDM synchronization:* The received signal has to be symbol-synchronized and some other OFDM signal processing has to be performed.
- *OFDM demodulation:* The synchronized data has to be demodulated.
- *Frame decoding:* The demodulated datastream contains symbols for the channel estimator, so called *pilot symbols* and symbols for the space-time decoder, so called *payload symbols*. These two have to be separated.
- *Channel estimation:* The MIMO detection algorithms require channel knowledge and the channel matrix $\boldsymbol{H}$ has to be estimated using the pilot symbols.
- *Space-Time decoding:* The received payload symbols have to be decoded.

The focus of our research is in developing MIMO algorithms and to investigate the feasibility of MIMO in a real-world office environment. So, we want to implement the algorithms into a real-time real-world system while not being fully knowledgeable of the computational requirements and hardware-software partitioning of the system. On one hand we want to

investigate new functions and algorithms and on the other hand, we have to be prepared to change the computational platform depending on the computational requirements of these algorithms, as we constrain ourselves in that the system has to operate in real-time.

Our MIMO system consists of different functions (tasks) that exchange data with one another. A function is implemented by some algorithm. However, we want to map these functions on a processing platform (hardware and/or software) in order to operate in real-time. We expect the functions and their interconnection structure (dataflow between the functions) to be more or less constant, while the algorithms and their computational complexity may vary. We want a way of system modeling which captures these communication aspects of the system and that allows mapping of an individual function to a particular processing platform independently of the other functions.

The theory of *Communicating Sequential Processes* (CSP) can be used to model our MIMO system. CSP is a notation for describing concurrent systems [3] and these systems interact with each other by means of communications. By using CSP, we can model all relevant aspects of our system. Moreover, we are able to focus on the functions and algorithms, real-time constraints and change computational platforms *while using a single system model only*.

3 CSP System Modeling

The required MIMO functions which are identified in the previous section, map directly to a CSP system model. The model for the receiver is given in figure 4. The CSP model consists

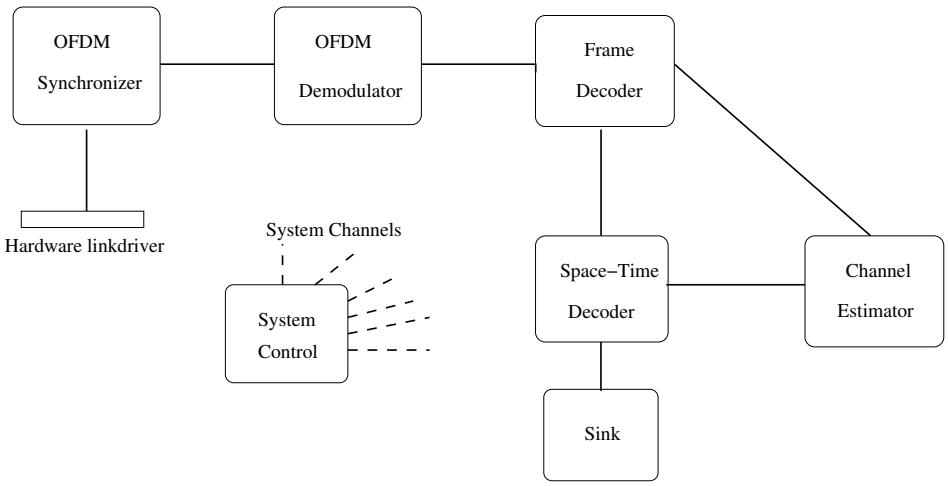

Figure 4: The CTCPP receiver architecture.

of five key processes:

- *OFDM Synchronizer:* the process acquires symbol synchronization and it communicates with the `OFDM demodulator` process. Synchronized symbols are written to the `OFDM demodulator` process.
- *OFDM Demodulator:* the process demodulates the OFDM symbols. Demodulated symbols are written to the `Frame Decoder` process.
- *Frame Decoder:* the `Frame Decoder` process extracts pilot symbols from the demodulated datastream. Pilot symbols are written to the `Channel Estimator` process and payload symbols are written to the `Space-Time decoder` process.

- *Channel Estimator:* the process performs channel estimation and channel estimates are written to the Space-Time Decoder.
- *Space-Time Decoder:* the process implements a MIMO detection algorithm.

The `System Control` process controls the other processes and status information is exchanged between these processes. Detected data is sent to a `Sink` process. The hardware linkdriver provides the sampled data from the analog radio. The communication in our system is not sample-based. The `OFDM Synchronizer` process delivers a group of OFDM symbols. A typical OFDM symbol consists of 32 samples and around 20 symbols are sent to the `OFDM Demodulator` process each time a message exchange takes place.

4 CTC++ Real-time kernel

To implement the CSP based architecture, we have used a real-time CSP kernel developed at the Control Engineering Group of our department [5], [6]. The C++ version of the kernel is used which provides us with `Process` classes and `Channel` classes. The kernel includes a scheduler and different priorities can be assigned to each process. The CTC++ kernel provides a novel hardware abstraction layer as is shown schematically in figure 5. Communication takes place over channels and each channel can have a linkdriver. The

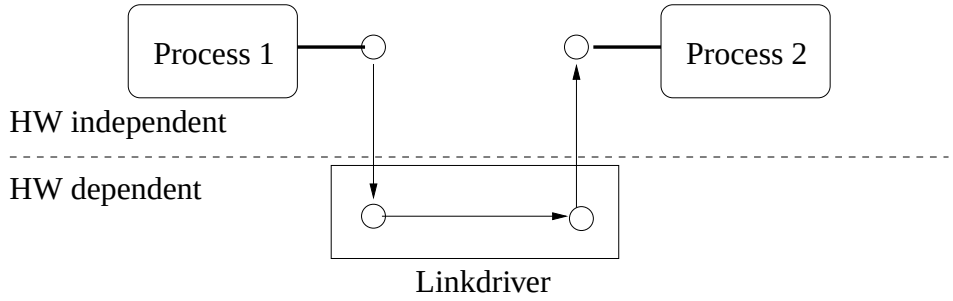

Figure 5: The hardware abstraction provided by CTC++.

linkdriver contains all the hardware dependent code. If the processing platform is changed, the linkdrivers are the only elements which require modifications. The different processes are implemented independently of the hardware they run on. Our processing platform is likely to change, however the receiver software itself does not have to be altered. Figure 6 shows the deployment diagram of our current testbed. The figure also shows a possible deployment diagram for a multi-processor platform.

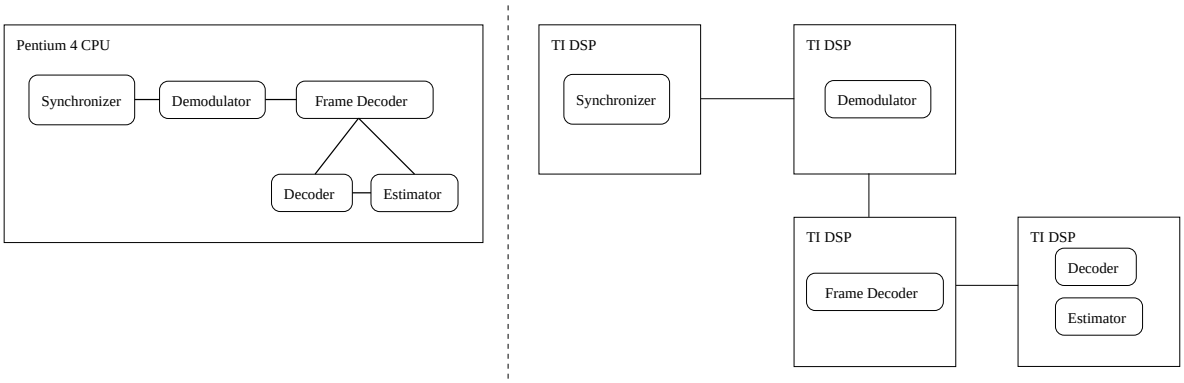

Figure 6: Two possible deployment diagrams, the first one is implemented.

4.1 Process communication

Whenever communication between two processes takes place, the kernel will provide the synchronization. The current implementation of the CTC++ kernel is more or less sample-based. Each communication instance consists of e.g. writing an integer on a channel. The channel itself makes a copy and a reading process is provided with this copy. In our system we would like to communicate on another scale. A buffer with OFDM symbols is written on the channel. This whole buffer would be copied by the channel and this introduces a performance loss. A solution is to write a reference to the buffer on the channel. This is illustrated in figure 7 for the `OFDM Synchronizer` process and the `OFDM Demodulator` process. The figure shows the two processes and the data structure which has to be communicated.

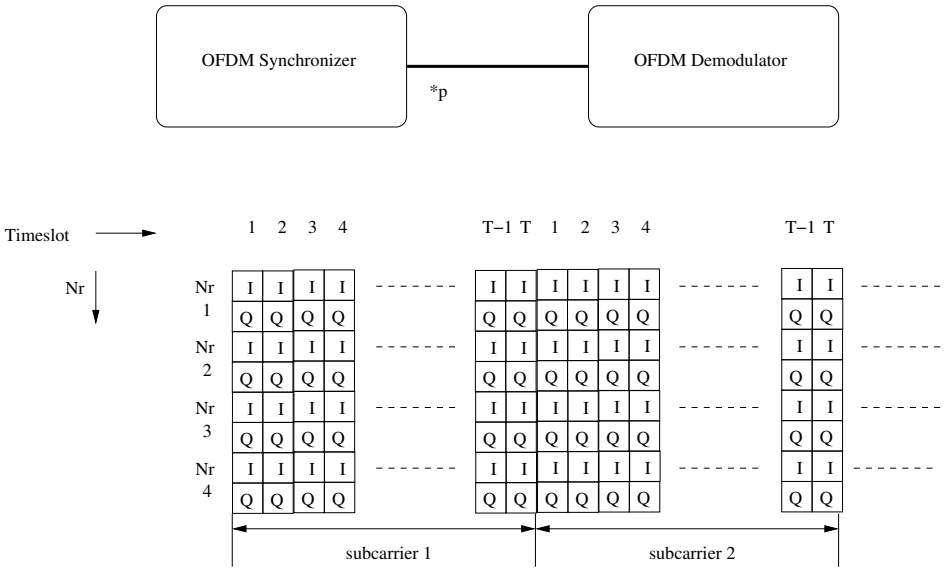

Figure 7: The communication on a channel.

The structure contains around 1 kB of data. A reference to this structure (*p) is written on the channel instead of the entire structure itself. However, this introduces some other difficulties with ownership. Only one process can make use of a data structure at a time. We have not solved these issues, but e.g. semaphores can be used to lock and unlock data structures.

4.2 Process priorities

In CTC++, we can also assign different priorities to the different receiver processes. The CTC++ does not require to assign priorities in the form of numbers. Priorities are assigned on the base of inter-process relations, so only a priority ranking is given. Some processes run on a higher priority as other processes. The priorities of the receiver processes are ranked as follows:

1. The highest priority is assigned to the `OFDM Synchronizer` process. If the synchronization is lost, synchronization has to be acquired again.
2. The second highest priority is reserved for the `OFDM Demodulator` process.
3. The third highest priority is reserved for the `Frame Decoder` process.
4. The fourth highest priority is reserved for the `Space-Time Decoder` process.
5. The fifth highest priority is reserved for the `Channel Estimator` process. Some channel estimates can be missed, because the Space-Time Decoder can also decode with older channel estimates. However, the system performance will degrade.

5 Testbed as an Experimental Platform

The testbed has been implemented and some experiments were made [4]. The receiver software runs on a Pentium 4 2.4 GHz machine and initial tests indicate that the system is able to run in real-time. Figure 8 shows a figure of the experimental setup in our laboratory. The

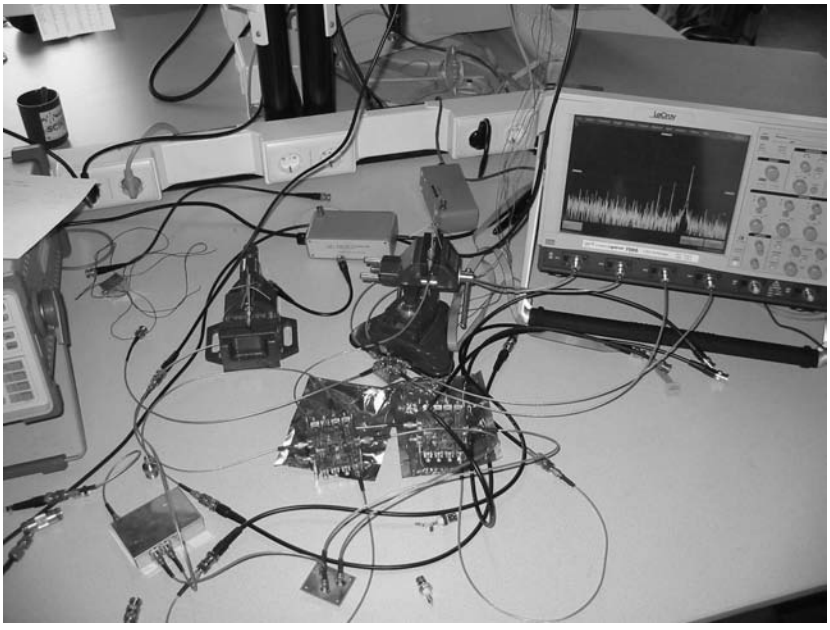

Figure 8: The experimental setup.

transmitter and receiver are separated a few meters and there is no line of sight component between the transmitter and receiver. Initial experiments have been performed and verify the concept of spatial multiplexing. Two independent datastreams have been transmitted from

two antennas and these two have been successfully separated at the receiver. For the experiment, the `Sink` process compares the received data with the data transmitted. The resulting bit error rate performance of the system is shown in figure 9.

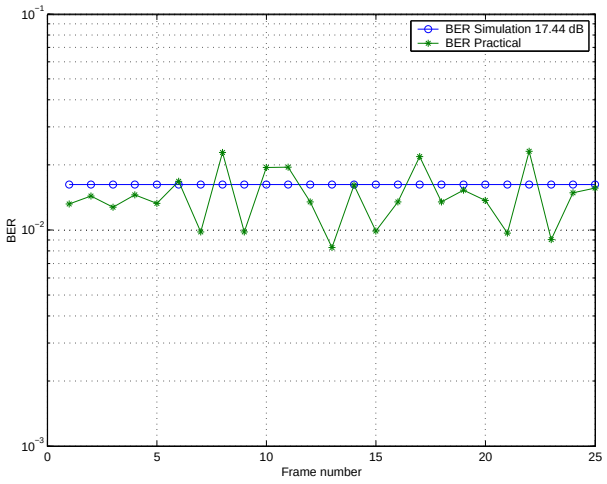

Figure 9: The raw bit error rate performance of the testbed.

In future research we plan to enlarge the channel bandwidth used, so the required amount of processing power increases. A multi-processor platform will be required to satisfy the need for computational power. The inter-process communication has to change and for this linkdrivers can be written.

6 Conclusions and Recommendations

In developing the testbed, the transition from traditional functional modeling to a CSP model is not large. The required functions map easily to CSP processes and the interaction between the processes is easily identified. As an example, we have described the receiver architecture for our testbed. The receiver software has been implemented with the CTC++ kernel and it seems to work functionally correct and testbed has been used to verify the concept of spatial multiplexing.

The use of the CSP and the CTC++ kernel is not only a nice way of system modeling, but also provides us with a scalable system (scalable with respect to the amount of antennas used and the transmission bandwidth). The software written has a proper hardware abstraction layer, which make it highly portable to other platforms. In future, a multi-processor system can be used and only the channel linkdrivers have to be rewritten.

For our type of application a design pattern could be developed and incorporated as an extension to the CTC++ kernel. The design pattern should include facilities for communicating references over channels and ownership management. This functionality could be incorporated into a special kind of channel. Also some additional questions have to be answered. For multi-processor platform e.g., the use of communicating references over channels implies a shared memory architecture. The alternatives to implement such a system, which is a combination of hardware and software design, can be investigated.

The CSP model also gives us the opportunity to mathematically verify the functional correctness of different receiver architectures and this can be done in future.

References

[1] I.E. Telatar, “Capacity of Multi-antenna Gaussian Channels”, IEEE Trans. Commun., vol. 42, No. 2/3/4, pp. 1617-1627, November-December 1999.

[2] D. Gesbert, H. Bölcskei, D. A. Gore, and A. J. Paulraj, “MIMO wireless channels: Capacity and performance prediction”, IEEE Globecom 2000, San Francisco, CA, pp. 1083-1088, Nov. 2000.

[3] A.W.Roscoe, “The Theory and Practice of Concurrency”, Prenctice Hall, 1998, ISBN 0-13-674409-5

[4] H.S.Cronie, “The Design and Implementation of an OFDM Spatial Multiplexing Testbed”, Ms.S. assignment, Signals and Systems, July 2003

[5] G.Hilderink, J.Broenink, W.Vervoort and A.Bakkers, “Communicating Java Threads”, www.ce.utwente.nl/javappp

[6] G.Hilderink, J.Broenink and A.Bakkers, “Communicating threads for Java”, www.ce.utwente.nl/javappp

[7] O.Evers, M.Sandell, J.Van de Beek “An introduction to orthogonal frequency-division multiplexing”, September 1996.