Technische Universität München

Lehrstuhl für Theoretische Informationstechnik

Search Based Software Design for Communication Receivers

Andreas Ibing

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieur (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender :	Univ.-Prof. Dr. techn. Josef A. Nossek
Prüfer der Dissertation :	
	1. Univ.-Prof. Dr.-Ing. Dr. rer. nat. Holger Boche
	2. Univ.-Prof. Dr.-Ing. Gerd Ascheid,
	Rheinisch-Westfälische Technische Hochschule Aachen

Die Dissertation wurde am 12.04.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 16.09.2011 angenommen.

Zusammenfassung

Die vorliegende Dissertation bearbeitet das Problem des Designs von Software der Bitübertragungsschicht für auf einer homogenen Mehrkern-Prozessorarchitektur basierende Kommunikationsempfänger. Dies beinhaltet die Auswahl von Komponentenalgorithmen sowie eines Berechnungs-Schedules, unter Beachtung von Möglichkeiten der iterativen Verarbeitung und Parallelisierung. Die Arbeit verfolgt einen Ansatz zur automatischen suchbasierten Optimierung unter Ausnutzung der speziellen Problemstruktur. Das Kriterium der Empfängeroptimierung ist dabei flexibel und kann als Utility-Funktion der Performanz-Parameter Übertragungsmodus, Signal-zu-Störverhältnis, Bitfehlerrate, Komplexität, Verzögerung sowie Durchsatz angegeben werden. Die Arbeit beginnt mit einer Betrachtung von Standardempfängerarchitektur und -algorithmen für Turbo-kodierte MIMO-OFDMA Übertragung. Danach wird in Abhängigkeit des Übertragungsmodus die generische Empfängerarchitektur aus bedingten Unabhängigkeiten der Übertragungsvariablen hergeleitet. Die Zerlegung in Empfängerkomponenten folgt aus der mathematischen Problemstruktur der stochastischen Inferenz. Zur praktischen Implementierung werden approximative Algorithmen zur Realisierung der verschiedenen Empfängerkomponenten betrachtet. Bei iterativer Verarbeitung wird jeweils Information aus den anderen Empfängerkomponenten als a priori Wissen verwendet. Die Zusammensetzung eines kompletten Empfängers aus Komponenten wird formal beschrieben und im Hinblick auf die Performanz-Parameter untersucht. Eine Möglichkeit der schnellen Prädiktion wird aufgezeigt, wobei eine Monte-Carlo Simulation des gesamten Empfängers nicht nötig ist. Darauf aufbauend folgt die automatische Suche im Empfänger-Entwurfsraum mittels Graphenalgorithmen. Im Vergleich zur Standardempfängerarchitektur ermöglichen die optimierten Empfänger einen Empfang bei signifikant geringerem SNR bzw. eine Komplexitätsreduktion im niedrigen SNR Bereich. Weiter wird die Wechselwirkung zwischen der Leistungsfähigkeit der physikalischen Schicht und der Ressourcenzuteilung betrachtet und eine Empfängerbewertung auf Netzwerkschicht in Abhängigkeit von Protokollkriterien ermöglicht. Abschliessend wird beispielhaft die Implementierung eines SDR Testbetts beschrieben, wobei Testbett-Parameter zur Veranschaulichung des Optimierungsverfahrens genutzt werden. Die vorliegende Arbeit basiert auf den auf Seite 191 folgende aufgelisteten Publikationen.

Abstract

The dissertation at hand deals with the problem of designing physical layer software for communication receivers based on a homogeneous multicore architecture. This comprises the choice of component algorithms and computation schedule, considering iterative processing and possibilities for parallelization. The approach which is used is an automatic search-based optimization exploiting the special problem structure. The criterion for receiver optimization is flexible and can be specified as utility function of the performance parameters transmission mode, signal to noise ratio, bit error rate, complexity, delay and throughput. The dissertation begins with a review of the standard receiver architecture and standard receiver algorithms for Turbo coded MIMO-OFDMA transmission. Then the generic receiver architecture is derived from conditional independencies of transmission variables, in dependence on the transmission mode. The receiver split-up into components follows the mathematical problem structure of stochastic inference. For practical implementation, approximative algorithms to implement the receiver components are treated in detail. In iterative processing, information from the other components is used as a priori knowledge. The composition of a complete receiver is formally specified and analysed regarding the performance parameters. A method for fast prediction is derived, which does not need Monte-Carlo simulation of the complete receiver. Based on this, automatic search in the receiver design space using graph algorithms is described. Compared to the standard architecture, the optimized receivers enable reception at significantly lower SNR or complexity reduction in the low SNR range respectively. The interrelation between performance of the physical layer and ressource allocation is looked at, and receiver evaluation on network layer in dependence on protocol criteria is enabled. Concluding, example implemenation in an SDR testbed is described, where testbed parameters are used to illustrate the optimization method. The document at hand is based on the publications listed on page 191 and the following.

Table of Contents

1 Introduction **1**

1.1 Aims and Structure of the Text . . . 1

1.2 Turbo-Coded MIMO-OFDMA Transmission . . . 2

1.3 3D WSSUS Channel Model . . . 3

1.4 Standard Receiver Architecture . . . 6

1.4.1 Downlink . . . 6

1.4.2 Uplink . . . 14

1.5 Noise, Interference and Countermeasures . . . 17

1.6 Software Defined Radio Baseband Hardware Model . . . 20

1.7 Search Based Software Engineering . . . 20

2 Generic Receiver Architecture **23**

2.1 Bayesian Inference . . . 24

2.2 Factorizing Joint Probability Density Function . . . 25

2.3 Receiver Components . . . 27

3 Component Algorithms Performing Approximate APP Computation **31**

3.1 Channel Estimator . . . 31

3.1.1 Channel Statistics Estimation and Tracking . . . 31

3.1.2 Approximate ML for First Iteration . . . 36

3.1.3 Approximate APP: Soft Data-Aided Channel estimation . . . 41

3.1.4 Accuracy Increase with A Priori Information . . . 53

3.2 MIMO Demapper . . . 54

3.2.1 Preprocessing Transformations . . . 55

3.2.2 Log-APP Ratio Generation . . . 56

3.2.3 Demapper Synthesis . . . 59

3.2.4 Accuracy Increase with A Priori Information . . . 68

3.3 Constituent Decoder . . . 74

4 Receiver Description Language **77**

4.1 Receiver Architecture Notation: Directed Bipartite Graph . . . 77

4.2 Component Algorithm Notation . . . 78

4.3 Serial Computation Schedule Notation: Regular Expression . . . 79

4.4 Parallel Schedule Notation: Parallelization on Factor Node Level . . . 80

5 Receiver Performance Prediction 83
5.1 Complexity . . . 83
5.2 Processing Delay . . . 85
5.3 Throughput . . . 85
5.4 Accuracy: BER and MI in dependence on SNR . . . 86
5.4.1 Parametric Tracking of LLR Density Evolution . . . 87
5.4.2 2-Parametric Gaussian Model and Offset Compensation . . . 90
5.4.3 Mutual Information Prediction Accuracy for Different Schedules . . . 93
5.4.4 Verifying BER and Threshold Prediction . . . 94

6 Automatic Receiver Optimization 97
6.1 Optimization Criteria . . . 98
6.2 5D Pareto Efficiency: Searching in the Decision Tree . . . 99
6.3 Example Search Results . . . 100
6.3.1 Operational SNR versus Receiver Complexity . . . 100
6.3.2 Processing Delay versus Number of Processing Elements . . . 103
6.3.3 Receiver Complexity versus Processing Delay . . . 103

7 Medium Access Control Aspects 107
7.1 Scheduling Criteria: Fairness vs. Efficiency . . . 108
7.1.1 Game Theoretic Cooperative Bargaining Solutions and Axiomatic Classification 109
7.1.2 Application as Schedulers . . . 111
7.2 Joint Scheduling and Link Adaptation . . . 115
7.2.1 Optimization Problem . . . 115
7.2.2 Approximative Scheduling Algorithms . . . 117
7.3 Gains and Limits of Adaptation: Imperfect Channel Knowledge . . . 124
7.3.1 Benefit of Channel Correlation Knowledge . . . 126
7.3.2 Signalling and Tracking . . . 132
7.4 Adapting Receiver Algorithm to Scheduling Parameters . . . 135
7.4.1 Motivation: Experimental Example Relations of Computation, Rate and SNR . . 136
7.4.2 Theoretical Explanation of Potential Gain . . . 137
7.4.3 Exploiting the Gain in Practical Scenarios . . . 141
7.5 Protocol Extensions for Multihop Relaying . . . 144
7.5.1 Frame Structure Extensions . . . 145
7.5.2 Aggregated Control Signalling . . . 145
7.5.3 Distributed Fair Scheduling . . . 146
7.5.4 Multi-Layer Routing . . . 146

8 SDR Testbed 149
8.1 Design Decisions . . . 149
8.2 Hardware Platform . . . 150
8.2.1 PCIe Backplane . . . 150
8.2.2 Cell Processor . . . 150
8.2.3 IXP Network Processor . . . 152

8.2.4 RF Frontends 153
8.2.5 Busses 155
8.3 Software Platform 157
8.3.1 Linux' on Host, Cell PPE and NPU XScale 157
8.3.2 Cell SPE Programming 158
8.3.3 NPU Microengine Programming 160
8.3.4 TCL Scripting 161
8.4 Implemented Functionality 162
8.4.1 Modem 162
8.4.2 Error Correction 165
8.4.3 Hybrid Iterative Reception 173
8.4.4 Medium Access Control and Radio Link Control 174
8.4.5 Testing and Visualization 180
8.5 Suitability for Further Developments 183
8.5.1 Distributed Physical Layer 183

Abbreviations **189**

Publication List **191**

Bibliography **195**

List of Figures

1.1 MIMO-OFDM transmitter using Turbo code. 3
1.2 Channel spreading functions. 3
1.3 Standard receiver architecture for downlink. 6
1.4 Downlink frequency synchonization accuracy. 8
1.5 Channel estimation accuracy. 11
1.6 Turbo decoder structure. 14
1.7 Standard receiver architecture for uplink. 14
1.8 Random access channel. 15
1.9 Synchronization preamble detection. 15
1.10 Influence of velocity, synchronization offsets and transmission bandwidth. 16
1.11 BER and PER for different modulation and velocity. 16
1.12 Interference randomization. 18
1.13 Interference cancellation. 19

2.1 Factor graph of joint pdf of transmission variables. 26
2.2 Factor graph for multi-user MIMO. 27
2.3 Message flow consisting of LLR vectors. 29

3.1 Time direction channel autocorrelation for different velocities. 32
3.2 4-parametric model of 2D power spectral density. 33
3.3 Channel statistics tracking for adaptive Wiener interpolation filtering. 33
3.4 Channel statistics parameter tracking example. 35
3.5 Example pilot lattice. 36
3.6 2D channel estimation filter kernel magnitude. 37
3.7 2D application of overlap-save method. 37
3.8 Complexity comparison of channel estimation methods. 40
3.9 Complexity break-down of convolution and upsampling based estimation. 41
3.10 MSE of adaptive WIF channel estimation. 41
3.11 3D ML channel estimation. 42
3.12 Iterative channel estimation setup. 43
3.13 Effective noise in APP channel estimation. 45
3.14 Expected normalized correlation. 45
3.15 APP channel estimation accuracy. 48
3.16 APP channel estimation application example. 49

3.17 APP channel estimation accuracy for MIMO-OFDM 52
3.18 Setup for iterative MIMO demapping-decoding. 54
3.19 MIMO demapper ’tool box’. 55
3.20 Max-Log approximation and exact bit probabilities. 58
3.21 MIMO MMSE stream separation based on Greville algorithm. 60
3.22 MIMO stream separation complexity. 63
3.23 MIMO demapping using QRD-M tree search. 64
3.24 Modulation set partitioning. 64
3.25 QRD-M MIMO demapping BER. 65
3.26 MIMO stream separation partial BER in dependence on LLR magnitude. 66
3.27 Hybrid uMMSE / subspace Max-Log-APP demapper complexity. 67
3.28 Hybrid uMMSE / subspace Max-Log-APP demapper performance. 68
3.29 A priori information and demapper accuracy increase. 69
3.30 APP MIMO demapper accuracy increase towards SIMO MRC accuracy. 73
3.31 Potential MIMO demapper accuracy gain with a priori information. 73
3.32 Turbo decoder EXIT chart. 75

4.1 Parallel schedule example. 81
4.2 Factor states. 81
4.3 Synchronous data flow graph. 82

5.1 Mutual information and AWGN variance for SISO BPSK. 89
5.2 EXIT chart prediction error for MIMO fading channel. 89
5.3 Mean, variance and mutual information of conditional Gaussian distribution. 90
5.4 LLR Density evolution example in iterative demapping-decoding. 90
5.5 MI surface in dependence on Gaussian parameters. 91
5.6 MI prediction with 2-parametric model. 92
5.7 MI offset compensation. 93
5.8 MI prediction accuracy. 94
5.9 BER prediction accuracy. 95

6.1 Example decision tree. 99
6.2 Cartesian product. 100
6.3 Automatic receiver optimization. 102
6.4 Minimum delay in dependence on number of processing elements. 103
6.5 Complexity and delay for different number of PEs. 104

7.1 Comparison of scheduling criteria. 108
7.2 Disagreement point and prioritization. 110
7.3 OFDMA Scheduler comparison. 113
7.4 Scaling with the number of users. 114
7.5 Approximative OFDMA scheduling algorithm for equal power allocation. 121
7.6 Approximative OFDMA scheduling algorithm for adaptive power allocation. 123
7.7 Channel correlation and scheduling granularity. 125

7.8 Packet error rates for different transmission modes. 126
7.9 Frequency selective scheduling and control lag. 127
7.10 3D selective scheduling gain and control lag. 128
7.11 2D CQI prediction error. 129
7.12 Mean and standard deviation of adaptation gain in dependence on control lag. 130
7.13 PER increase of adaptive transmission with control lag. 131
7.14 Throughput degradation of adaptive transmission with control lag. 131
7.15 Expected throughput in dependence on control lag. 132
7.16 Expected throughput for different signalling schemes. 134
7.17 Experimentally obtained performance-complexity-rate tradeoffs. 136
7.18 Rate-SNR-Computation tradeoff model. 137
7.19 Joint transmitter and receiver adaptation. 141
7.20 Receiver computation as variable. 144
7.21 Hierarchical scheduling and global fairness. 146

8.1 SDR demonstrator. 151
8.2 System overview with main components. 152
8.3 Cell CPU and development boards. 153
8.4 IXP NPU and development board. 154
8.5 Virtex5 FPGA and PCIe development board. 155
8.6 Bus topology and available raw data bandwidths. 156
8.7 Bus throughput measurements. 157
8.8 Reducing stalled cycles on Cell SPE. 159
8.9 Processor utilization and matrix data types. 160
8.10 NPU read access times for different memory types. 161
8.11 MIMO stream separation benchmarks on Cell SPE. 162
8.12 List-QRD-M MIMO demapper benchmarks on Cell SPE. 164
8.13 Convolutional encoder. 166
8.14 Trellis terminology. 167
8.15 Example decoding trellis. 168
8.16 ACS butterfly implementation with SPE vector intrinsics. 168
8.17 Throughput of truncated Viterbi in dependence on trace back memory. 171
8.18 BER after decoding. 171
8.19 ACS butterfly on SPE for 8-state trellis. 173
8.20 Accuracy of hybrid iterative receiver. 175
8.21 Protocol processing. 178
8.22 Microengine thread processing times. 180
8.23 Microengine utilization and memory bus load. 180
8.24 Base station protocol test in one host. 181
8.25 Generation of Gaussian distributed random numbers. 182
8.26 AWGN channel emulation. 183
8.27 Linear feedback shift register for pseudo-random binary sequence. 183
8.28 Network MIMO with propagation delay differences. 184

8.29 ISI caused by delay differences and OFDM processing. 185
8.30 Static base station cooperation areas. 186
8.31 Processing units and static cooperation areas. 187
8.32 Base station grid and cooperation areas. 187

List of Tables

3.1 Complexity of position-dependent APP channel estimation. 45
3.2 APP channel estimation complexity for MIMO-OFDM 52
3.3 Operation counts on vector processor for Greville-based MIMO equalization. 61
3.4 MAC operation comparison for different LMMSE MIMO equalizer implementations. . . 62
3.5 Complexity of Max-Log BCJR decoder with LTE parameters. 75

4.1 Factor node designation. 78
4.2 Component algorithm designation. 79

5.1 Hardware-dependent complexity measure of software implementation. 85

6.1 Enumeration of factor combinations. 100

7.1 Scenarios for scheduler comparison. 112
7.2 Scenario for adaptive power allocation. 122

8.1 Local store utilization for channel estimation and equalization. 165
8.2 Modem benchmark. 165
8.3 Local store utilization for Viterbi decoder. 170
8.4 Receiver component benchmarks on SPE. 174
8.5 Data plane thread partitioning and worst case line rate. 176

Chapter 1

Introduction

1.1 Aims and Structure of the Text

This work has two aims: first, to find the optimal software design and implementation of a software defined radio (SDR) physical layer, and second, to develop a realtime demonstrator for novel mobile radio systems.

Conditions for the first aim (software design and implementation) are a given set of transmission parameters and performance requirements for the receiver. The software can run on a fixed number of homogeneous processor cores with given instruction set.
Since the receiver's function of reconstructing a transmitted packet from distortion by the radio channel is a mathematical problem, chapter 2 looks into the structure of this mathematical problem to explore the possibilities for the software design. The generic receiver structure is derived, from which any specific receiver structure can be generated by parameter choice. The generic structure is a breakdown of receiver functionality into components. Each component solves a mathematical problem, but computation of the correct solution is far too complex.
To find ways to make the components computable with a limited number of instructions and without degrading their accuracy too much, chapter 3 looks into approximative algorithms to implement the components. Different algorithms can be found, showing a tradeoff between the number of instructions and accuracy for each component: with respect to the complete receiver, Pareto efficient component implementations can be found.
To yield a complete receiver, component implementations are combined in chapter 4. Since there are different component implementations available, can be done in many ways. And according to the generic receiver structure, the receiver's components can be recomputed with updated input from neighbouring

components, if the overall accuracy is not yet sufficient. Further, sometimes several components can be computed at the same time if multiple processing elements (processor cores) are available.
To enable a quick comparison of different receiver candidates, chapter 5 deals with fast determination of a receiver's performance parameters. A method is derived to quickly predict the accuracy of the concatenated processing without actually executing it. Other predicted performance parameters are the sum number of instructions necessary to reconstruct a packet and the necessary processing time.
Chapter 6 searches the optimal software as selection of component algorithms, computation order and mapping to processor elements. This combinatorial problem is represented as a tree, and the search uses tree traversal with branch and bound, where the search target may e.g. be minimal sum number of instructions, minimal packet processing time or maximum throughput under BER and SNR constraints.

One conditions for the second aim of the work (development of realtime demonstrator) is to minimize development effort: the testbed intention is to demonstrate functionality, not to be prototype for a commercial product. Another condition is to meet the performance requirement of executing modern signal processing and coding for 20MHz 2x2 MIMO.
The demonstrator needs higher layer protocol functionality, which is dealt with in chapter 7.
The demonstrator follows the SDR approach and is described in detail in chapter 8.

There are several relations between the two aims of the work. The demonstrator hardware is used as example target for the automatic optimization method, i.e. parameters like the demonstrator instruction set are used in the chapters 3 to 6. Relations between physical layer optimization and medium access control criteria are described in chapter 7.

1.2 Turbo-Coded MIMO-OFDMA Transmission

Modern radio communication systems which face multipath propagation normally use OFDM transmission – e.g. the standards 3GPP LTE, mobile WiMax, DVB-T(2) and WLAN 802.11a/g/n. It is often combined with multiple antenna signal processing schemes (MIMO) to enhance spectral efficiency. One driving reason for this is the possibility of low-complexity equalization, which can be performed independently per subcarrier (also for MIMO). Modern systems further employ strong channel codes like parallel concatenated convolutional codes (Turbo codes) or LDPC codes – both of which are fit for iterative decoding.

A block diagram of a transmitter for Turbo-coded MIMO-OFDMA transmission is shown in Fig. 1.1- An information word u is Turbo encoded with the desired code rate (puncturing, rate matching, e.g.

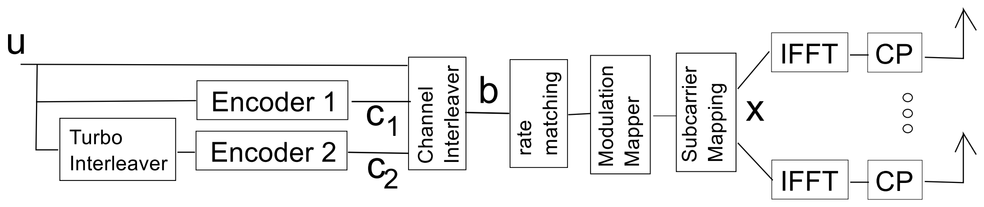


Figure 1.1: MIMO-OFDM transmitter using Turbo code.

[CNB+08]) into code word b. After the QAM modulation mapper, it becomes the vector of complex symbols x in the frequency domain. The diagram depicts joint encoding of different data streams (over different transmit antennas), which is possible for single-user MIMO (SU-MIMO). For MU-MIMO, joint encoding of all data streams to one terminal is also possible. The other case would be separate encoding of each MIMO data stream (e.g. in LTE). After mapping the baseband symbols to subcarriers and antennas, IFFTs are performed for the symbol vectors of each transmit antenna independently, before cyclic prefices (CP) are added to guard against multipath propagation by creating periodicity for receiver FFT processing.

1.3 3D WSSUS Channel Model

Channel variations over time are classified as large scale fading on the one hand and small-scale fading on the other. Large scale fading means changes which occur over distances much larger than one wavelength, especially path loss and shadowing (e.g. by buildings). Small scale fading describes the effect of superposition of incident wave fronts taking into account their phase differences – small scale thus means the effects in the distance order of one wavelength. An incident wave front at the receive antenna position p can be described by path delay τ, direction of arrival $\Omega = \{\phi, \theta\}$, and Doppler frequency shift ν. For antenna arrays at transmitter and receiver, the antenna element positions are denoted with the vectors $\mathbf{p}_T$ and $\mathbf{p}_R$. The spreading function for propagation with discrete number K of

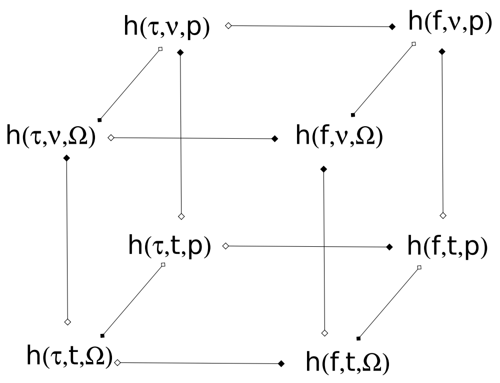


Figure 1.2: Channel spreading functions according to [Gal07].

rays only can therefore be written as [Fle00, Gal07]:

$$h(\tau, \nu, \Omega) = \sum_{k=1}^{K} \alpha_k \cdot \delta(\tau - \tau_k) \cdot \delta(\nu - \nu_k) \cdot \delta(\Omega - \Omega_k) \tag{1.1}$$

A transmit signal $s(t)$ is received at receiver position p as [Gal07]:

$$x(t, p) = \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{S^2} h(\tau, \nu, \Omega) s(t - \tau) e^{-(jk\Omega^T p)} e^{j2\pi\nu t} d\Omega d\tau d\nu \tag{1.2}$$

This description was obtained in [Fle00] as generalization from the Bello system equations [Bel63] by Fourier transform along the spatial dimension at the receiver side. In [Gal07] the model is extended again to include also the spatial dimension at the transmitter (direction of departure DOD, and direction of arrival DOA). The spreading function then becomes $h(\tau, \nu, \Omega_T, \Omega_R)$. The 3-dimensional selectivity is represented by $h(f, t, p)$ (frequency-selective, time-selective and antenna-selective fading). The extended system equations and their Fourier transform relationships are illustrated in Fig. 1.2 (if the diagram would include DOD and DOA separately, it would be 4-dimensional and include 16 equations [Gal07]). For transformation along the angular domain (2D transform, azimuth and elevation), the distance vectors are normalized with the wavelength λ to $q = \frac{p}{\lambda}$, yielding a 'modified' Fourier transform [Gal07]:

$$h(\tau, \nu, p) = \int_{S^2} h(\tau, \nu, \Omega) e^{-j\frac{2\pi}{\lambda}\Omega^T p} d\Omega \tag{1.3}$$

$$h(\tau, \nu, \Omega) = \frac{1}{\lambda} \int_{\mathcal{R}^3} h(\tau, \nu, p) e^{j\frac{2\pi}{\lambda}\Omega^T p} dp \tag{1.4}$$

The multidimensional channel autocorrelation in this form is given as (frequency-time-space correlation function [Fle00, Gal07]):

$$R(\Delta f, \Delta t, \Delta q_T, \Delta q_R) = \mathbb{E}[h(f, t, q_T, q_R) \cdot h^*(f + \Delta f, t + \Delta t, q_T + \Delta q_T, q_R + \Delta q_R)] \tag{1.5}$$

The autocorrelation function in frequency domain only is [Gal07]:

$$\rho(\Delta f) = \int_{\mathbb{R}} \int_{S^2} R(\Delta f, \nu, \Delta q_T, \Omega_R)|_{\Delta q_T = 0} \, d\Omega_R d\nu \tag{1.6}$$

The autocorrelation in time domain is

$$\rho(\Delta t) = \int_{\mathbb{R}} \int_{S^2} R(\tau, \Delta t, \Delta q_T, \Omega_R)|_{\Delta q_T=0}\ d\Omega_R d\tau \tag{1.7}$$

and the autocorrelation in spatial domain:

$$\rho(\Delta q_t) = \int_{\mathbb{R}} \int_{S^2} R(\tau, \nu, \Delta q_t, \Delta q_R)|_{\Delta q_R=0}\ d\nu d\tau \tag{1.8}$$

According to the Wiener-Khinchin theorem, the power spectral density of a wide-sense-stationary random process is the Fourier transform of the corresponding autocorrelation function. The power delay profile is therefore [Gal07]:

$$P(\tau) = \int_{\mathbb{R}} \int_{S^2} R(\tau, \nu, \Delta q_T, \Omega_R)|_{\Delta q_T=0}\ d\Omega d\nu = \mathbb{E}[|h(\tau)|^2] \tag{1.9}$$

The power Doppler profile is:

$$P(\nu) = \int_{\mathbb{R}} \int_{S^2} R(\tau, \nu, \Delta q_T, \Omega_R)|_{\Delta q_T=0}\ d\Omega d\tau = \mathbb{E}[|h(\nu)|^2] \tag{1.10}$$

And the power angular profile:

$$P(\Omega_R) = \int_{\mathbb{R}} \int_{\mathbb{R}} R(\tau, \nu, \Delta q_T, \Omega_R)|_{\Delta q_T=0}\ d\tau d\nu = \mathbb{E}[|h(\Omega_R)|^2] \tag{1.11}$$

The physical radio channel is sampled by the digital communication system: in space it is sampled at the antenna positions, with N_T transmit and N_R receive antennas. It is sampled in time with period Δt at a number of sampling frequencies with distance ΔF. The sampled channel at time t_n and frequency F_k (on the sampling grid) is denoted as matrix:

$$\mathbf{H}(t_n, F_k)\ \in\ \mathbb{C}^{N_R \times N_T} \tag{1.12}$$

In the following the base band channel model is used for convenience. Channel correlation is especially relevant for channel estimation in chapter 3 and scheduling and link adaptation in chapter 7.

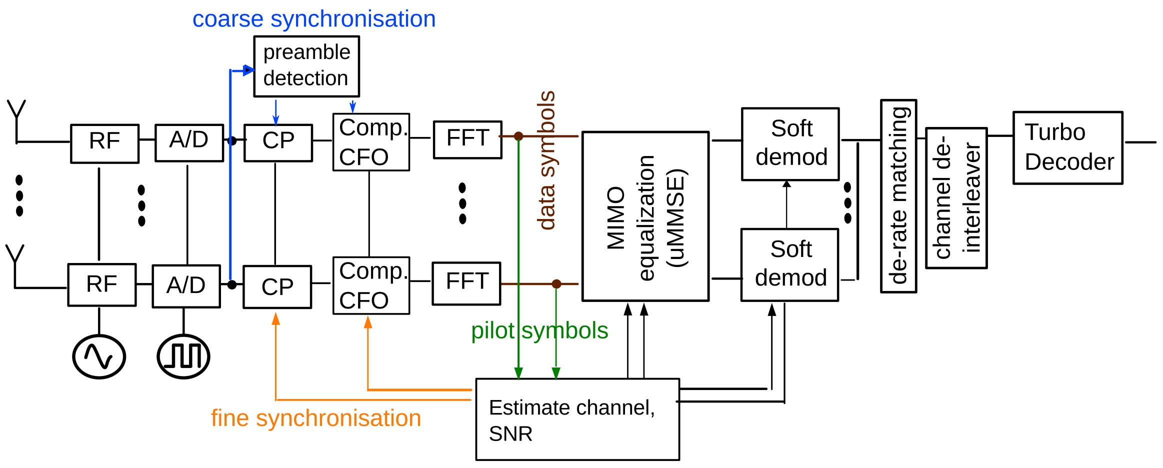


Figure 1.3: Standard receiver architecture for downlink (compare [MIJ08]).

1.4 Standard Receiver Architecture

In this section the standard receiver architectures for MIMO-OFDMA downlink and uplink are shortly reviewed and their differences are highlighted.

1.4.1 Downlink

A block diagram of the downlink receiver (terminal receiver) is shown in Fig. 1.3. In the following, the main signal processing functions are described.

Coarse synchronisation: A mobile receiver needs to synchronize its time, frequency and (to a lesser extend) sampling clock to the transmitter. Carrier frequency offset (CFO) is caused by oscillator mismatch and relative movement, and is for OFDM divided into integer CFO (integer number of subcarrier distances) and fractional CFO. Initial information about time and frequency offset compared to the base station is obtained by correlation using knowledge about a synchronisation preamble which is contained in the downlink signal. In the obtained correlation profile, the existence and position of a preamble is determined by peak-to-average detection using thresholding.

Timing offset Preamble detection uses either cross-correlation with the known preamble sequence or (with less complexity) autocorrelation of the received values exploiting adequate structure of the sequence. A 2-fold periodic preamble in time domain with period L is autocorrelated in [SC97]:

$$P(d) = \sum_{m=0}^{L-1} (r^*_{d+m} r_{d+m+L}) \tag{1.13}$$

Computation can be implemented as moving sum. For a constant channel, sample pairs of the two preamble halves add up with the same phase (phase increase between two symbol halves). The energy (used for thresholding later) is computed [SC97]:

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2 \tag{1.14}$$

And the correlation profile yielded as:

$$M(d) = \frac{|P(d)|^2}{R(d)^2} \tag{1.15}$$

The threshold value is a tradeoff between preamble miss probability and false detection probability, which both depend on SNR. If several receive antennas are available, maximum ratio combining (MRC) can be employed to benefit from receive diversity. In a similar scheme [SBB95], autocorrelation has been applied to the OFDM cyclic prefix – which finds the start of OFDM symbols, not a frame start marked by a preamble.

Fractional frequency offset After timing detection using the peak-to-average threshold, the angle of the peak

$$\hat{\Phi} = \text{angle}\{P(\hat{d})\} \tag{1.16}$$

gives the estimation of fractional CFO [SC97]:

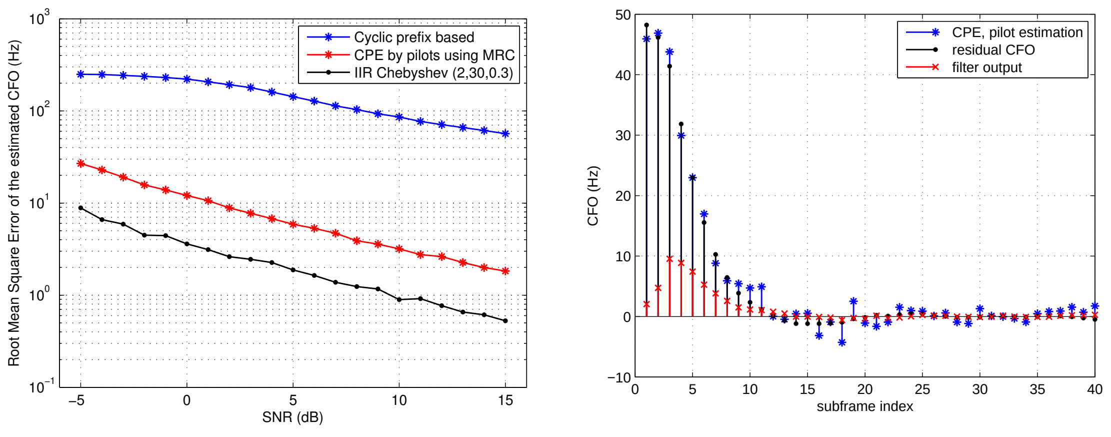
$$\hat{\Delta f} = \frac{\hat{\Phi}}{\pi T} \tag{1.17}$$

This fractional CFO is then compensated by multiplication with a complex oscillation.

Integer frequency offset [SC97] uses a second preamble OFDM symbol, which compared to the first one is differentionaly modulated with sequence v_k. The integer CFO (shift of 2g positions) can be determined again by correlation:

$$\hat{g} = \underset{g}{argmax} \frac{|\sum_{k\in X} x^*_{1,k+2g} v^*_k x_{2,k+2g}|^2}{2(\sum_{k\in X} |x_{2,k}|^2)^2} \tag{1.18}$$

where $x_{2,k}$ is the modulation symbol on subcarrier k in the second preamble OFDM symbol x_2. The integer CFO is then also compensated.



(a) Example CFO estimation error for coarse synchronization (autocorrelation using CP), fine synchronization and tracking (with filtering) [MIJ08].

(b) Example evolution of remaining CFO from coarse estimation to tracking [MIJ08].

Figure 1.4: Downlink frequency synchonization accuracy.

Transmit diversity for synchronization preambles To also exploit transmit diversity in case of MIMO, [SZ03] uses a cyclically shifted Zadoff-Chu sequence per transmit antenna. Perfect cross-correlation is maintained for a shift larger than CIR. With preamble period N_p, the shift can be N_p/N_T, for N_T transmit antennas. Joint correlation at the receiver for CFO estimation is [SZ03]:

$$\lambda(m) = \sum_{k=m}^{m+N_p-1} r^H(k)r(k+N_p) \tag{1.19}$$

Time and Frequency Offset Precompensation CFO causes inter-carrier interference (ICI) due to receiver FFT processing (subcarriers loose orthogonality). Time offset causes phase rotation after receiver FFT and possibly inter-symbol interference (ISI, if outside CP). To avoid both effects, estimated time and frequency offset are precompensated before receiver FFT [SFFM01]. Time offset is compensated by selecting the start of the FFT symbol vector, frequency offset is compensated by multiplication with complex oscillation. The two precompensation parameters are fed initially by coarse synchronisation, later by fine synchronisation.

Channel estimation: Channel estimation is normally enabled by transmission of pilot symbols which are known to the receiver, so that the channel is sampled at the pilot positions. The receiver needs to reconstruct the channel at the positions of (unknown) data symbols from the received pilot values, which are corrupted by noise. The Wiener filter [Wie49] is the optimal noise reduction filter for a noisy signal (in the minimum mean square error sense) if autocorrelation and noise variance are known. It is used for one-dimensional signals like speech as well as multidimensional signals like images (2D) and video (3D) [Woo06]. The combination of Wiener filtering and interpolation to solve the channel estimation problem is known as Wiener Interpolation Filter and has been described for OFDM channel estimation in [HKR97]. It is normally applied to the time-variant channel transfer function. After removing cyclic prefix and performing N-point FFT, the signal in frequency domain is given as

$$Y[n,k] = X[n,k]H[n,k] + Z[n,k] \tag{1.20}$$

where $Y[n,k]$ is the received symbol on subcarrier n in OFDM symbol k. As initial step (least squares method), the receiver obtains noisy samples of the time-variant channel transfer function at the locations of the known pilot symbols:

$$\tilde{H}_p[n,k] = \frac{Y_p[n,k]}{X_p[n,k]} = H_p[n,k] + \frac{Z_p[n,k]}{X_p[n,k]} \tag{1.21}$$

Noise suppression filtering based on the initial least squares estimates can be written as

$$\hat{H}_p[n,k] = \sum_{n_0,k_0} \boldsymbol{c}^*[n,k]\tilde{H}_p[n-n_0,k-k_0] \tag{1.22}$$

where $\boldsymbol{c}[n,k]$ are filter coefficients. The estimation Mean Square Error

$$MSE(\boldsymbol{c}) = \mathbb{E}\{\hat{H}_p[n,k] - H_p[n,k]\} \tag{1.23}$$

becomes minimal for the coefficients satisfying the Wiener-Hopf-Equation (follows from principle of orthogonality [Hay01, HKR97]):

$$\boldsymbol{c}[n,k]|_{opt} = (\boldsymbol{R}[n,k] + \frac{\boldsymbol{I}}{SNR})^{-1}\boldsymbol{r}[n,k] \tag{1.24}$$

where $\boldsymbol{r}[n,k]$ is the crosscorrelation between the channel transfer value $H[n,k]$ at the position to be computed and the pilots within filter range (values written as vector). $\boldsymbol{R}[n,k]$ is the autocorrelation (values written as matrix) between $H[n,k]$ at all the pilots within filter range. The OFDM channel transfer function's 2D correlation function $r_H(\Delta t, \Delta f) = \mathbb{E}\{H(t1, f1)H^*(t2, f2)\}$ can be separated into time and frequency parts:

$$r_H(\Delta t, \Delta f) = r_H(\Delta t)r_H(\Delta f) \tag{1.25}$$

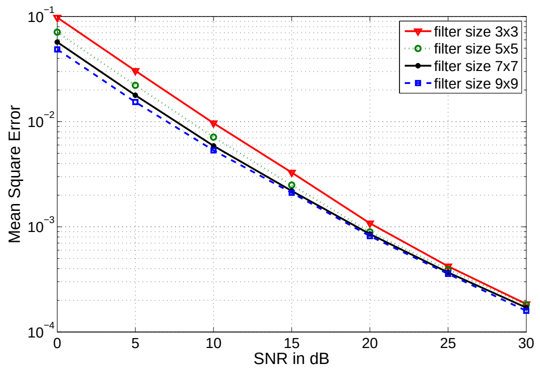
Receivers use either a (less complex) static Wiener filter or an adaptive Wiener filter. For a static Wiener filter the autocorrelation is assumed to be known apriori and normally predefined according to 'worst-case' assumptions (like maximum delay spread and maximum Doppler spread). This means assuming less correlation between pilots than there will be typically, and the approach is referred to as 'robust' filter design – robust against filter mismatch [SHFS04]. Static 'robust' filtering normally assumes a uniform multidimensional power density spectrum (for MIMO-OFDM with the dimensions delay spread, Doppler spread and angular spread [SHFS04, Aue09]). From the power spectrum assumption, the channel autocorrelation follows by IFFT [OS09]. For the uniform spectrum case the correlation in time direction becomes:

$$r_{H,unif}(\Delta t) = sinc(2\pi f_D \Delta t) \tag{1.26}$$

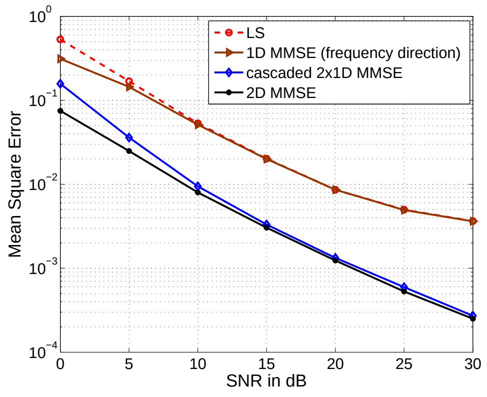
with $sinc(x) = sin(x)/x$, and f_D being the maximum Doppler shift. $\Delta t = \Delta k t_s(L+N)$ with t_s being the sampling period and $t_s(L+N)$ one OFDM symbol duration. The correlation in frequency direction:

$$r_{H,unif}(\Delta f) = sinc(\pi T_m \Delta f) e^{j2\pi\tau_{shift\Delta f}} \tag{1.27}$$

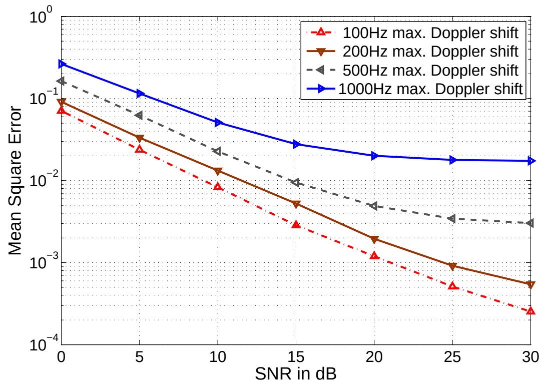
with $T_m = Lt_s$ as the maximum delay, $\tau_{shift} = T_m/2$, $\Delta f = \Delta F \Delta l$ with ΔF being the subcarrier distance. With the distances $[\Delta k \ \Delta l]$ from pilots inside filter range, the coefficients can be computed from (1.25) and (1.24). SNR estimation can be done by singular value decomposition based subspace methods (projection onto noise subspace, normally static filter) [ESB+98], or with IFFT from CFR – signal and noise variance are then contained in the CIR area of an OFDM symbol, and outside only noise variance [JL07]. To enable channel estimation for MIMO, pilot positions are exclusive per transmit antenna (other transmit antennas do not transmit on these positions, zero symbol). The described Wiener interpolation filtering can then be performed without crosstalk per transmit-receive antenna pair. E.g. [LI10b] illustrates filter performance simulatively by measuring channel estimation mean square error for different channel autocorrelation and noise parameters. The assumed underlying channel model is the 'urban macro' type from [3GP06]. The gain of increasing filter size is illustrated in Fig. 1.5a, where the resulting MSE is shown over channel SNR. Fig. 1.5b shows a comparison of a 2D filter with with different filter types, namely a 1D filter in frequency direction and a cascaded 2x 1D filter (separable, applied first in frequency direction and then in time direction). All three filters use the same amount of pilots for Wiener filtering. The effect of degrading accuracy with increasing terminal velocitiy (different max. Doppler shift) is illustrated in Fig. 1.5c.



(a) Channel estimation accuracy for varying 2D filter size (urban macro channel, Jakes Doppler spectrum with 72Hz max. Doppler shift) [LI10b].



(b) Accuracy of different filter types. All filters except LS have the same number of 49 filter coefficients (urban macro channel, max. Doppler shift 100Hz) [LI10b].



(c) Influence of different Doppler spread (Jakes spectrum) on accuracy of 2D filter with 7×7 coefficients [LI10b].

Figure 1.5: Channel estimation accuracy.

Fine synchronisation, tracking: Based on channel estimation, time and frequency offsets can be re-estimated with higher accuracy and fed back to pre-compensation before the FFT.

CFO is described as time domain multiplication with the complex oscillation $e^{j\frac{2\pi\Delta\phi n}{N}}$, where the phase increases $\Delta\phi$ for neighbouring samples. Here it is assumed that the frequency dispersive channel is constant for two neighbouring OFDM symbols. A received symbol in time domain (nth sample in mth OFDM symbol) is:

$$y_{m,n} = \frac{1}{N}\left\{\sum_{k=0}^{N-1} X_{m,k}H_k e^{2\pi\frac{kn}{N}}\right\} \underbrace{\cdot\, e^{j\frac{2\pi\Delta\phi n}{N}}}_{\text{CFO}} + n_n \tag{1.28}$$

where $X_{m,k}$ is the transmit symbol on the kth subcarrier. After FFT the received symbol on the lth subcarrier is:

$$Y_{m,l} = \frac{1}{N}\sum_{n=0}^{N-1}\left\{\sum_{n=0}^{N-1} X_{m,k}H_k e^{j2\pi\frac{kn}{N}}\right\} \cdot e^{j\frac{2\pi\Delta\phi n}{N}} \cdot e^{-j2\pi\frac{ln}{N}} + N_{m,l} \tag{1.29}$$

With the approximation $e^{j\frac{2\pi\Delta\phi n}{N}} \approx 1 + j\frac{2\pi\Delta\phi n}{N}$, this can be rewritten as [AW02]:

$$Y_{m,l} \approx H_l X_{m,l} \underbrace{\frac{1}{N}\sum_{n=0}^{N-1}(1 + j\frac{2\pi\Delta\phi n}{N})}_{\approx e^{j\psi}\text{, common phase error}} + \underbrace{\frac{1}{N}\sum_{k=0,k\neq l}^{N-1} H_k X_{m,k} \sum_{n=0}^{N-1}(1 + j\frac{2\pi\Delta\phi n}{N})e^{-j2\pi\frac{(l-k)n}{N}}}_{\text{intercarrier interference}} + N_{m,l} \tag{1.30}$$

i.e. the received symbols undergo a common phase error (same for all subcarriers) and intercarrier interference. Since the channel is estimated including the common phase error, CFO and 'clean' channel are not separable in one OFDM symbol. But between neighbouring OFDM symbols it is:

$$\mathbb{E}(\hat{H}_{m.l} - \hat{H}_{m-1,l}) = e^{j2\pi\Delta\phi} \tag{1.31}$$

The CFO can thus be estimated by maximum ratio combining over subcarriers (and receive antennas if several are available):

$$\Delta\hat{\phi} = \frac{1}{2\pi}\angle\sum_{l=0}^{N-1}\hat{H}_{m.k}\hat{H}^*_{m-1,l} \tag{1.32}$$

Results for a sequence of OFDM symbols can be filtered. Accuracy increase from inital coarse CFO synchronization to CFO tracking is illustrated in Fig. 1.4b. Precompensating CFO before receiver FFT avoids it causing ICI.

Delay: Time offset is tracked to fully use the CP against delay spread, and to avoid possible filter (phase) mismatch of a static Wiener channel estimation filter. Timing offsets, as long as not exceeding the CP, cause only a circular shift of the samples of an OFDM symbol in the time domain. According to the Fourier transform shift theorem

$$\mathcal{F}(y_{n-m}) = Y_k \cdot e^{\frac{-2\pi j}{N}km} \tag{1.33}$$

this means a phase rotation which linearly grows over subcarriers. To estimate the time offset, either a regression over the phase of the channel samples can be performed [MIJ08] – which is biased by the channel's own CIR (decreasing phase over subcarriers). Or preferably it can be estimated using an IFFT transform from CFR to CIR, followed by a sliding window search for the CP position which contains maximum CIR energy [SFFM01].

MIMO symbol demapping, LLR generation After channel estimation, the MIMO streams are separated according to the MMSE criterion, to enable per-stream demapping. With noise variance σ^2, the

MMSE receiver filter matrix for separation of the streams is given by:

$$G_{\text{bias}} = (H^H H + \sigma^2 I)^{-1} H^H$$

The unbiased version is $G_{\text{unb}} = S\,G_{\text{bias}}$, where S is the diagonal matrix which removes the bias introduced by the MMSE criterion [Zim07]:

$$S = \text{diag}\Big(\frac{1}{(G_{\text{bias}}H)_{1,1}}, \dots, \frac{1}{(G_{\text{bias}}H)_{N_T,N_T}}\Big)$$

The equalized symbol vector is:

$$\hat{\mathbf{x}} = G_{\text{unb}}\mathbf{y}$$

The receiver then computes a log-likelihood ratio (LLR) for each transmit bit c:

$$L(c) = \ln \frac{P(c = +1)}{P(c = -1)} \tag{1.34}$$

For ease of notation (and implementation), a bit c can take the values ± 1 instead of 1 and 0. The LLR for transmit antenna i and bit position j is (under assumption of Gaussian noise and Max-Log approximation per stream):

$$L(c_{i,j}) = -\frac{1}{2\sigma_{eq}^2}\Big(\min_{x_i \in \mathcal{X}^1_{j=1}} |\hat{x}_i - x_i|^2 - \min_{x_i \in \mathcal{X}^1_{j=-1}} |\hat{x}_i - x_i|^2\Big)$$

where σ_{eq}^2 is the noise variance on the stream after filtering. $x_i \in \mathcal{X}^1_{j=1}$ means the set of symbols where the bit whose LLR is to be computed has the value 1, $\mathcal{X}^1_{j=-1}$ is the complement set. The applied max-log approximation considers only the Euclidean distances to the closest two symbol candidates (with positive and negative bit, respectively). For separable modulation sets (like used e.g. in LTE), implementation reduces to independent one-dimensional table lookups for inphase and quadrature bits.

Channel decoder: After de-ratematching (depuncturing and de-interleaving), the Turbo decoder is run. A Turbo decoder consists of two constitutional decoders (fitting to the two encoders), and Turbo interleaver and Turbo de-interleaver. The constitutional soft-input soft-output convolutional decoders use the BCJR algorithm [BCJR74]. Normally systematic constitutional codes are used, so that the two decoders only need to exchange extrinsic information for the information bits (and not also the parity bits). A block diagram of the Turbo decoder is shown in Fig. 1.6. The constitutional codes can be identical (like e.g. the 8-state decoders in LTE [3GP09]). As example of a pseudo-random Turbo interleaver, quadratic

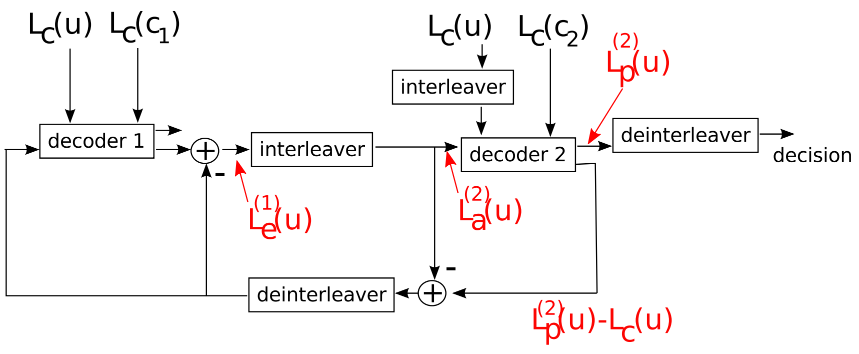


Figure 1.6: Turbo decoder structure (compare [LC04]).

polynomial permutation (QPP) is used in LTE [Nim08]. Normally 3-11 Turbo decoder iterations are run, accuracy of the LLR signs improves over iterations. The information bits are obtained by quantizing the information bit LLRs to their sign.

1.4.2 Uplink

The structure of an uplink OFDMA receiver is illustrated in Fig. 1.7.

Coarse synchronization: Random Access Channel For entering the system by non-synchronized transmission, a random access channel (RACH) is used (its position in time and frequency is broadcast in the downlink). To guard against a long round-trip delay, the RACH is much longer in time than one OFDM symbol (which is only guarded against delay spread, not round-trip delay). To allow for reduced-complexity computation of correlation using FFT-based fast (cyclic) correlation, a RACH preamble contains an own (extra long, for round-trip delay) cyclic prefix. RACH and RACH preamble are illustrated in Fig. 1.8. Detection of RACH preambles is done on the time domain sample stream before OFDM cyclic prefix removal and the receiver FFTs. A large FFT (fitting the preamble sequence length) of the time domain samples in the RACH window is computed, followed by elementwise multiplication with

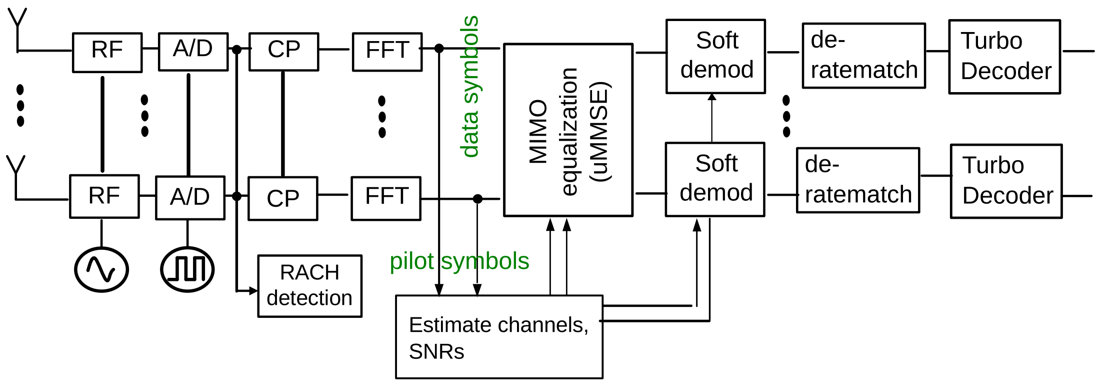
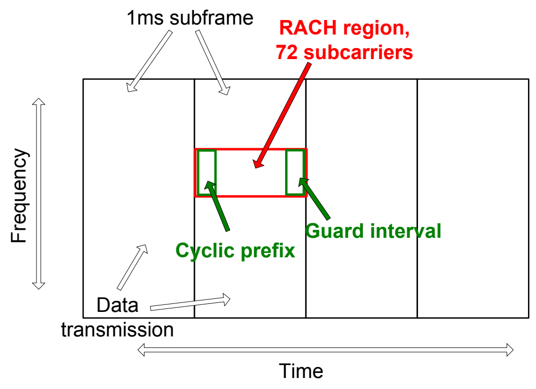
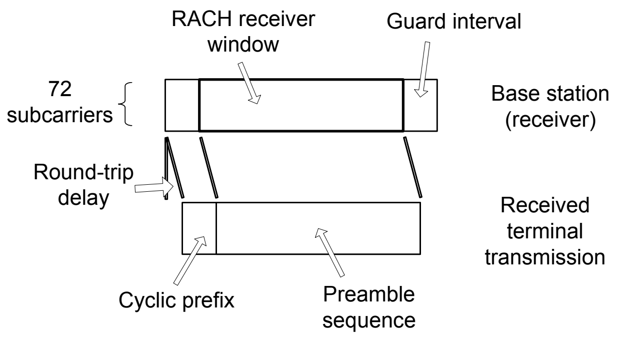


Figure 1.7: Standard receiver architecture for uplink (compare [ZI09]).

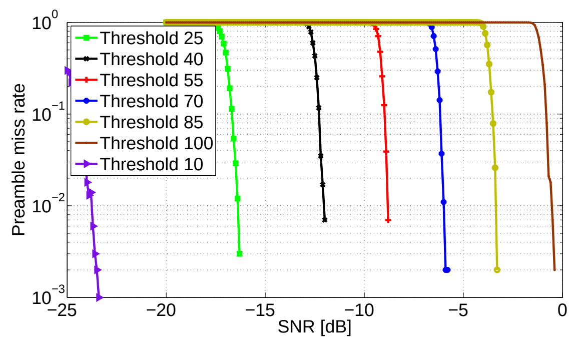


(a) Position of LTE random access channel in time and frequency. A random access burst is much longer than an OFDM symbol, to allow for several kilometers of round-trip delay [DPSB07, ZI09].

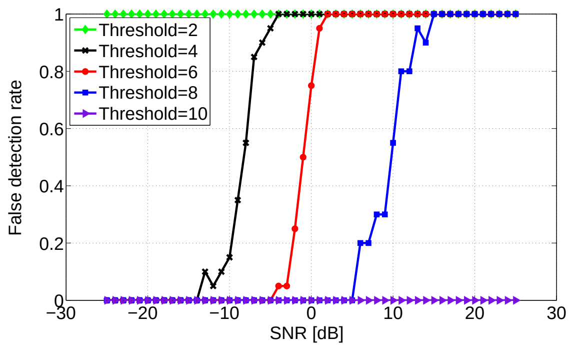


(b) Unsynchronized random access burst in LTE [DPSB07, ZI09].

Figure 1.8: Uplink random access channel (RACH).



(a) Preamble miss rate in dependence on threshold (AWGN channel). For negligible miss rate in the operating SNR region the threshold should be smaller than around 40.

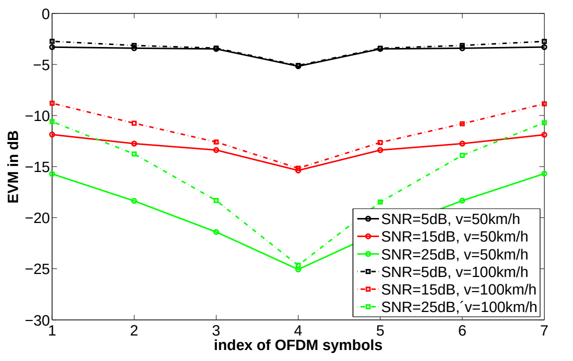


(b) Preamble false detection rate in dependence on threshold (AWGN channel). For negligible false detection rate the threshold shoud be larger than 10.

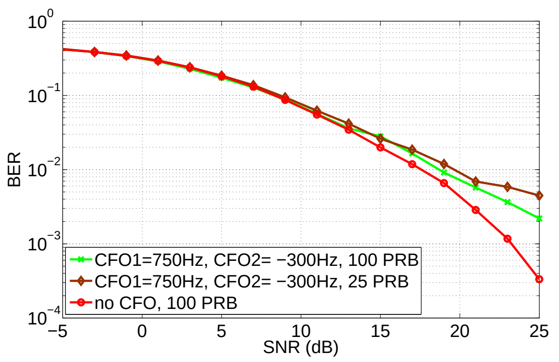
Figure 1.9: Peak-to-average threshold for preamble detection [ZI09].

the stored FFT of a preamble [DPSB07]. The resulting vector is IFFT transformed, followed by peak to average detection (comparison to a threshold value) [WHZ06]. If a peak is detected, the round-trip delay estimate is given by the peak position and phase. The peak-to-average threshold value should both avoid preamble misses and false detections in the receiver operating SNR region (illustrated in Fig. 1.9a and 1.9b). After detection of an initial ranging preamble, the base station protocol stack instructs the terminal to adjust its clock to compensate the round-trip delay (timing advance) – since the base station is/has reference time and frequency.

Channel estimation uses Wiener filtering like in the downlink; pilots are transmitted in the resources allocated to a terminal. To provide uplink channel knowledge to the base station also on other subbands, the base station can command transmission of a sounding signal. The Wiener filter is applicable also to the sounding signal. The error vector magnitude (average squared distance of symbols from their ideal



(a) Error vector magnitude for equalization of one resource block for different noise levels and terminal velocities (at 2.6GHz). Pilot symbols are transmitted in the fourth of 7 OFDM symbols (middle OFDM symbol) [ZI09].



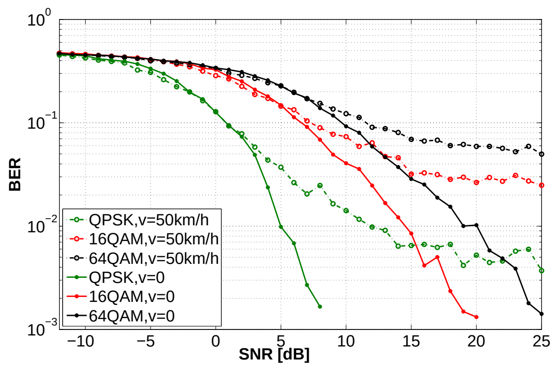
(b) Uncoded bit error rates without CFO and with compensation of different CFOs (two terminals) for different transmission bandwidths and DFT-S OFDMA [ZI09].

Figure 1.10: Influence of velocity, synchronization offsets and transmission bandwidth.

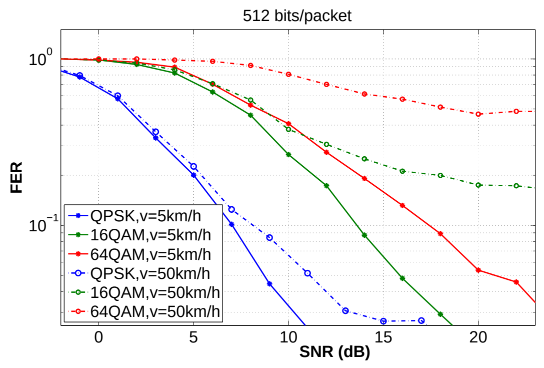
locations in the complex plane) after equalization is illustrated in Fig. 1.10a for different noise levels and velocities (Jakes Doppler Spectrum). Uncoded bit error rates without CFO and with two different CFOs (two terminals) and compensation are illustrated in Fig. 1.10b for different bandwidths.

MIMO demapping and Turbo Decoding use the same algorithms as described for the downlink. Fig. 1.11a illustrates resulting bit error rates (BER) after turbo decoding for different modulation levels for small packet size and different terminal velocities [ZI09]. Fig. 1.11b illustrates frame error rates (FER) for intermediate packet size.

Differences to downlink processing In OFDMA uplink, transmission of a terminal is non-continuous. Considering the transmission gaps (in time and frequency), tracking of time and frequency offsets can still be applied [IZK08]. Since the remaining offsets are individual per terminal, receiver-based pre-



(a) Bit error rates for the worst case of 40 information bit packet (smallest packet size), for different terminal velocities, urban micro channel and DFT-S OFDMA [ZI09].



(b) Frame error rates for different modulations and terminal velocities, packet size 512 information bits, DFT-S OFDMA [ZI09].

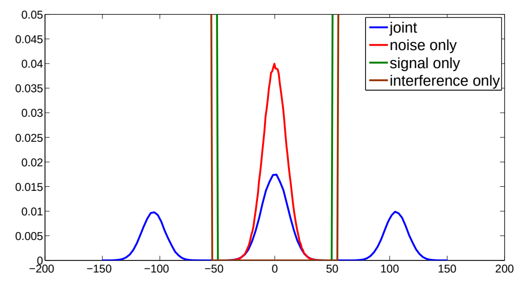
Figure 1.11: BER and PER for different modulation and velocity.

compensation before FFT is not possible (and also not wanted, since the BS is reference). After receiving a timing advance instruction, the terminal sets a timer to switch it to 'unsynchronized' mode – uplink transmission then requires transmission of a new RACH preamble. The BS can command an adjustment of timing advance with a certain granularity and reset the terminal's synchronization timer [3GP07]. Similar to timing advance there would also be the possibility to command a frequency advance with certain granularity (to reduce ICI).

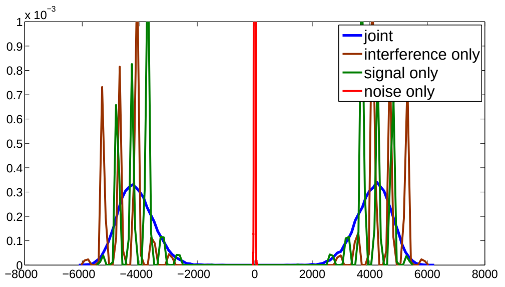
1.5 Noise, Interference and Countermeasures

Noise can be countered by increasing packet size and using a strong code (apart from increasing transmission power). The difference between thermal noise and interference is that interference is structured: it has non-Dirac shaped autocorrelation and also has code structure.

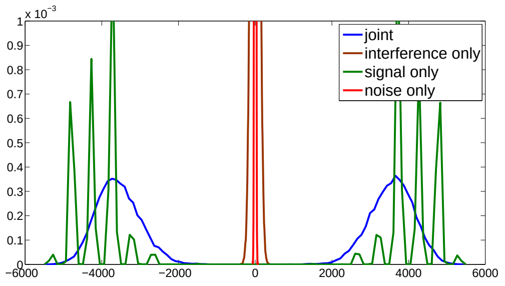
Interference Randomization A (pseudo-)randomization is used by interfering transmitters to avoid interference with same structure. The receiver treats interference as noise. The decoder effects coding gain for input with the adequate structure – and an effect of interference randomization is to avoid applying this gain also to interference. Current cellular systems use interference randomization in the form of terminal-specific scrambling (LTE) or (pseudo-)randomized OFDM subcarrier mapping (Wimax). Fig. 1.12 shows an example (assuming same constant channel and QPSK modulation), where randomization enables communication at negative SIR.



(a) LLRs after demapper, strong interference of -0.83 dB SIR, SNR 14dB



(b) LLRs after decoder. Interference with (same) code structure is amplified by decoder. Decoding is unsuccessfull (interference information word is output).



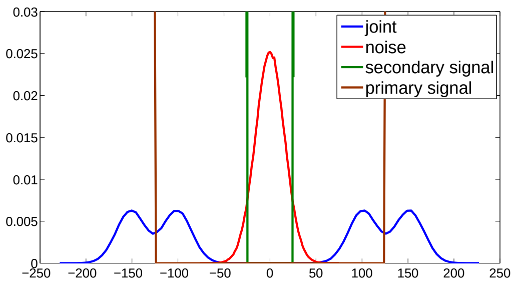
(c) LLRs after decoder. Unstructured (random) interference is not amplified by decoder. Decoding is successfull.

Figure 1.12: Interference randomization [ipa10a].

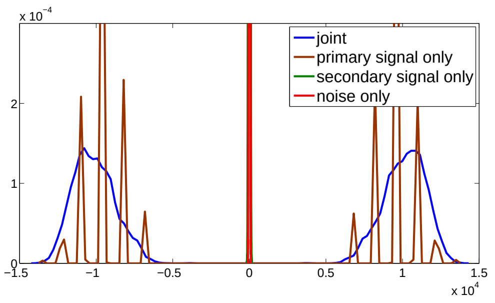
Interference Suppression The receiver treats (white) noise and interference together as coloured noise, and applies filtering to improve SINR. This approach needs an estimate of both noise and interference statistics (or their joint statistics). An example is MIMO stream separation with known interference directions and powers (sometimes called ’interference rejection combining’).

Interference Coordination and Interference Alignment A signalling protocol is used to avoid or reduce interference. One example is LTE, where cell edge users from neighbouring cells are scheduled in different subbands, and the schedulers semi-statically negotiate subband transmit power levels [3GP10b]. A generalization from this would be multi-cell scheduling. Another example is cognitive radio using spectrum sensing. The aim is to only transmit when not causing (too much) interference. The ’full-blown’ case is Interference Alignment [CJ08], where all transmitters have full channel state information and jointly precode to orthogonalize signal subspace and interference subspace at all receivers.

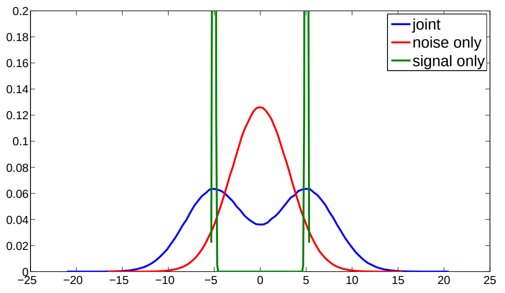
Interference Cancellation The approach is to decode and subtract interference at the receiver. Interference is treated as separate signal. Interference cancellation at the transmitter according to dirty paper coding [Cos83] needs channel state information (which renders it somewhat unpractical). Interference cancellation at the receiver needs channel estimation also for the interfence channel, and knowledge



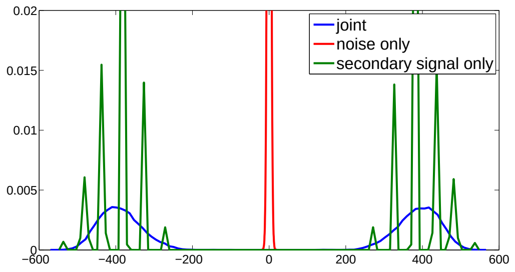
(a) LLR density after demapper.



(b) LLR density after decoder for primary signal.



(c) After demapper, after subtracting decoded and re-distorted primary signal.



(d) After decoder for secondary signal.

Figure 1.13: Interference cancellation [ipa10b].

about the interference signal structure. An example is again MIMO detection – successive detection of separately encoded MIMO streams. The principle can also be applied to successive detection of different transmission systems using overlapping channel ressources. An implementation obstacle is that the interference reduces the available ADC dynamics. An illustration of interference cancellation in the LLR domain is given by Fig. 1.13. A special case of interference cancellation is the concept of antenna cancellation [CJS$^+$10], which can be described as echo cancellation in the analog domain (before LNA and ADC) to allow for full-duplex transmission/reception at the same frequency — using multiple transmit antennas whose waves cancel out at receive antenna positions.

Joint signal and interference detection While interference cancellation uses successive detection, this more complex approach detects using the joint model of signal and interference. Examples are MIMO maximum likelihood detection, or the concept of multi-cell joint detection (sometimes called ’coordinated multi-point’).

1.6 Software Defined Radio Baseband Hardware Model

There are different definitions of ’Software Defined Radio’. A common one is that baseband modulation and demodulation are performed in software. Recent multicore processors for baseband processing in the terminal are the EVM (embedded vector processor [BHM+05]), the MUSIC chip [Ram07] and the Sandbridge processor [GIM+06]. For high-end terminals an SDR implementation offers a cost advantage due to smaller (cheaper) silicon area: in case of many functions which are not all needed at the same time, only a small increase in code memory is necessary [Ram07]. Recent multicore DSPs for base station base band processing are TCI6488 [Tex07] (C64x+ cores) and MSC8144 [Fre07] (Starcores). Channel decoding is always coprocessor accelerated, also possibly FFT processing. Some related multicore examples from high performance computing are the Cell processor [KDH+05], Tilera [Til09], Larrabee [SCS+08] and Tesla [NVI07]. Physical layer processing in the testbed (chapter 8) is implemented on the Cell. The main topics of this thesis are receiver algorithms, their optimization and software implementation. Hardware criteria like area and power consumption are not considered. A homogeneous multicore processor architecture is assumed, allowing to flexibly distribute (or dynamically allocate) computational power between receiver functions.

1.7 Search Based Software Engineering

Search based software engineering means the approach of applying search algorithms for optimization to software engineering problems. It has been applied to problems in software testing, requirements analysis, software design, software development and software maintenance. A list of example applications is given in [Har07].

[KWA+09] describes a design flow for co-design of hardware and software for SDR. [WDS10] also relates to the problem of joint design of hardware and software for embedded systems, and formulates it as two coupled optimization problems (’hardware knapsack problem’ and ’software knapsack problem’; the hardware solution must provide sufficient ressources for the software solution). After partitioning of system functionality into hardware and software, different configuration options remain both in hardware (e.g. clock speed) and in software (e.g. image resolution), which are to be jointly optimized.

The document at hand deals with communication receivers whose functionality is mainly realized in software. The desired result is an optimization of the software implementation of receiver algorithms (software part of the two coupled problems), in dependence on the underlying hardware (instruction set etc.). The mathematical problem structure and signal processing and decoding algorithms of communication receivers offer a wide tradeoff between accuracy, complexity and processing delay. Software complexity can be measured as processor cycles on target processor core(s). The special communication receiver problem structure allows to quantify accuracy using an information theoretic measure (namely mutual information), even for intermediate processing results. The communication receiver software design problem therefore proves especially suited for search based optimization.

Chapter 2

Generic Receiver Architecture

In this chapter, stochastic inference frameworks are shortly treated, namely (loopy) Bayesian belief propagation, Kullback-Leibler (KL) divergence minimization and Bethe free energy. The corresponding graphical models are shortly reviewed, especially factor graphs. The joint probability density function of receiver random variables is factorized exploiting conditional independencies between variables, and the resulting generic receiver structure is illustrated as graphical model for single-user and multi-user reception and concatenated channel codes. This chapter is a breakdown of the receiver into components (and their connections) which perform a posteriori probability (APP) computation on variable subsets, to iteratively approximate the joint a posteriori probability. The components themselves are described in detailed in chapter 3.

Stochastic inference and the related graphical models (illustrating conditional dependence and independence of variables) are described in [Pea88, Bea03]. Bayesian belief propagation is derived in [Pea88] as propagation of local ’beliefs’, which corresponds to sequential updating of probability distributions according to Bayes’ theorem. It has originally only been intended for graphs without loops, i.e. trees [Pea88]. Propagation of any initially known probability densities (’evidence’) through the tree computes the correct a posteriori probabilities. The algorithm is generalized as ’summary propagation’; the sum-product algorithm and corresponding factor graph model for illustration are described in [KFL01]. Bayesian belief propagation yields the correct joint a posteriori probability density function when the variable dependency structure is a tree (no loops) — for arbitrary topology it can be used as approximation [Pea88, Mac04] (iterative approximation, as local beliefs then propagate in loops). Loopy belief propagation is also discussed in [JN02, RU08], its convergence for graphs is discussed in [WF01]. While Belief propagation works on directed graphs, the Bethe free energy inference works on undirected graphs (Markov random fields). Different stochastic inference frameworks are described and compared in [YFW01, YFW05, Min05, AS06]. The relation between free energy and belief propagation is described in [YFW01], the correspondence of iterative maximization of a posteriori probabilities versus minimization of KL divergence is given in [AS06].

Inference frameworks have been applied to understand as well as to generate algorithms for receivers. In [MMC98] it is shown that the standard Turbo decoding algorithm is an instance of loopy Belief propagation. Factor graphs have been applied to (iterative) receivers including symbol demapping in [WS01, Wym07]. KL divergence minimization has been used for CDMA reception in [HLR+08] and for OFDM reception in [MKF+09]. The Bethe free energy approach has been applied to iterative receivers in [LL09].

2.1 Bayesian Inference

Bayesian inference for trees A Bayesian network is a directed acyclic graph (DAG), where random variables are represented as nodes. Conditional independencies between nodes are given by the d-separation property of the DAG [Bis06]. Probabilistic inference is done by local message-passing in the tree (i.e. only between neighbouring nodes). To avoid counting information twice during inference, the aposteriori probability of each variable is split into two parts: the ’causal’ support π (for ’probability’) and the ’diagnostic’ support λ (for ’likelihood’). The ’belief’ (aposteriori probability) of a variable b is computed as:

$$\mathrm{BEL}(b) = \alpha\lambda(b)\pi(b),$$

where the ’causal’ support $\pi(b)$ is the probability density of the variable b, conditioned on the evidence $\mathbf{e}^+$ in the graph propagated to the node through its parents:

$$\pi(b) = \mathbb{P}(b|\mathbf{e}^+).$$

The ’diagnostic’ support $\lambda(b)$ is the probability of the evidence $\mathbf{e}^-$ propagated to node b through its children, conditioned on b:

$$\lambda(b) = \mathbb{P}(\mathbf{e}^-|b)$$

α always denotes a constant to normalize a probability density. Incoming messages from several children $a_1 \dots a_J$ are combined as:

$$\lambda(b) = \prod_{j=1}^{J} \lambda_{a_j}(b)$$

The combination of incoming messages from several parents $d_1 \dots d_K$ is [Pea88]:

$$\pi(b) = \sum_{d_1,\dots,d_K} \mathbb{P}(b|d_1,\dots,d_K) \prod_{i=1}^{K} \pi_b(d_i)$$

The message sent to parent d_k is computed as [Pea88]:

$$\lambda_b(d_k) = \alpha \sum_b \lambda(b) \sum_{d_i:\ i\neq k} \mathbb{P}(b|d_1,\dots,d_K) \prod_{n\neq k} \pi_b(d_n)$$

The message sent to child a_j is:

$$\pi_{a_j}(b) = \alpha \frac{\mathrm{BEL}(b)}{\lambda_{a_j}(b)}$$

Each node only has to ’know’ its probability conditioned to that of its immediate parents.

For graphs (with loops) For exact marginalization, the graph can be transformed into a tree using the junction tree algorithm [LS88], where cycles are eliminated by clustering them into single nodes. An example would be to implement a Turbo decoder by applying the Viterbi algorithm to the supertrellis (which hugely increases complexity).

A low-complexity approximation alternative is to just apply belief propagation to the graph (loopy belief propagation, LBP) [Mac04]. For the case of more than two factor nodes (more than one cycle), the sequence of factor node updates (local message propagation) may influence convergence behaviour.

2.2 Factorizing Joint Probability Density Function

MIMO transmission at time instance t over the channel matrix $\mathbf{H}^{(j,t)}$ of the jth OFDM subcarrier is denoted as

$$\mathbf{y}^{(j,t)} = \mathbf{H}^{(j,t)} \cdot \mathbf{x}^{(j,t)}(\mathbf{b}^{(j,t)}) + \mathbf{n}^{(j,t)}. \tag{2.1}$$

We assume that the channel does not have memory, which can also be considered as subcarrier model in MIMO-OFDM transmission. $\mathbf{b}^{(j,t)}$ is a vector of transmit bits as part of the complete codeword $\mathbf{b}$, $\mathbf{x}^{(j,t)}$ is the corresponding vector of modulated symbols. The complete set of received symbol values of the message (all time instances) is denoted $\mathbf{y}$. The transmitter uses Turbo coding, so that the code word $\mathbf{b}$ consists of the information bits $\mathbf{u}$, parity bits $\mathbf{c}_1$ of the first constituent encoder and parity bits $\mathbf{c}_2$ of the second constituent encoder. For a single bit of the bit vector $\mathbf{b}$ at position i it is written b_i. Fig. 1.1 on page 3 illustrates the encoding and modulation signal flow at the transmitter.

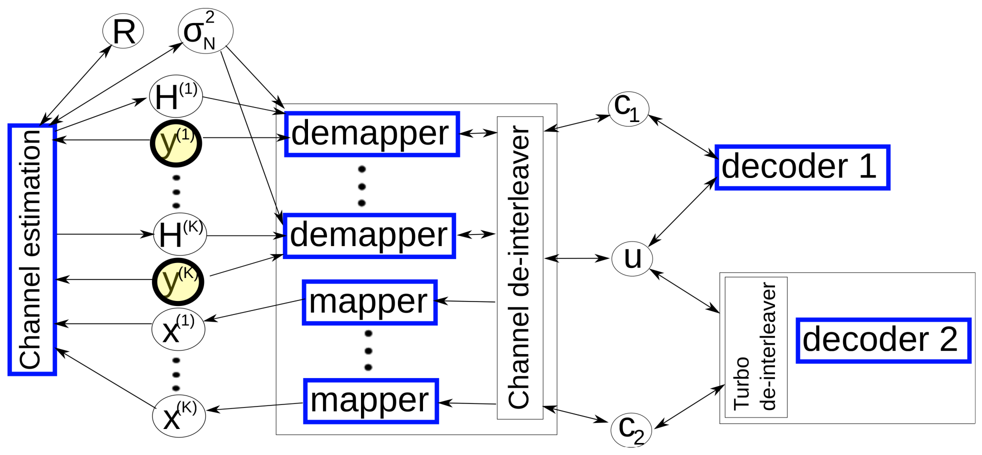


Figure 2.1: Factor graph of joint probability density. Variable nodes are circles, factor nodes are squares. ’Evidence’ $\mathbf{y}$ is shaded. All variable nodes are vectors, $\mathbf{H}^{(t)}$ are matrices [ipa10c]

Maximum receiver accuracy would be reached if computing the maximum likelihood solution on codeword basis:

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} \mathbb{P}(\mathbf{u}|\mathbf{y})$$

As this is practically infeasible, the practical approach is an iterative local approximation of the information bit APPs with subsequent binary quantization. The joint probability density can be factorized:

$$\mathbb{P}(\mathbf{u},\mathbf{y}) = \Big(\prod_{j,t} f_{Dem}(\mathbf{b}^{(j,t)},\mathbf{y}^{(j,t)},\mathbf{H}^{(j,t)})\Big)\cdot f_{Dec1}(\mathbf{u},\mathbf{c_1})\cdot f_{Dec2}(\mathbf{u},\mathbf{c_2})\cdot f_{CE}(\mathbf{y},\mathbf{x})\cdot\Big(\prod_{j,t} f_{Map}(\mathbf{b}^{(j,t)})\Big)$$

which corresponds to a demapper for each different time instance, the two constituent decoders, a soft symbol mapper for each time instance and channel estimation (for all symbol positions of the code word). The received vectors $\mathbf{y}$ are ’evidence’. The factorization is illustrated in Fig. 2.1 (using a factor graph notation similar to [Bis06]). The factor graph for multi-user MIMO (MU-MIMO) is illustrated in Fig. 2.2 for two users: demapping is performed jointly, decoding separately.

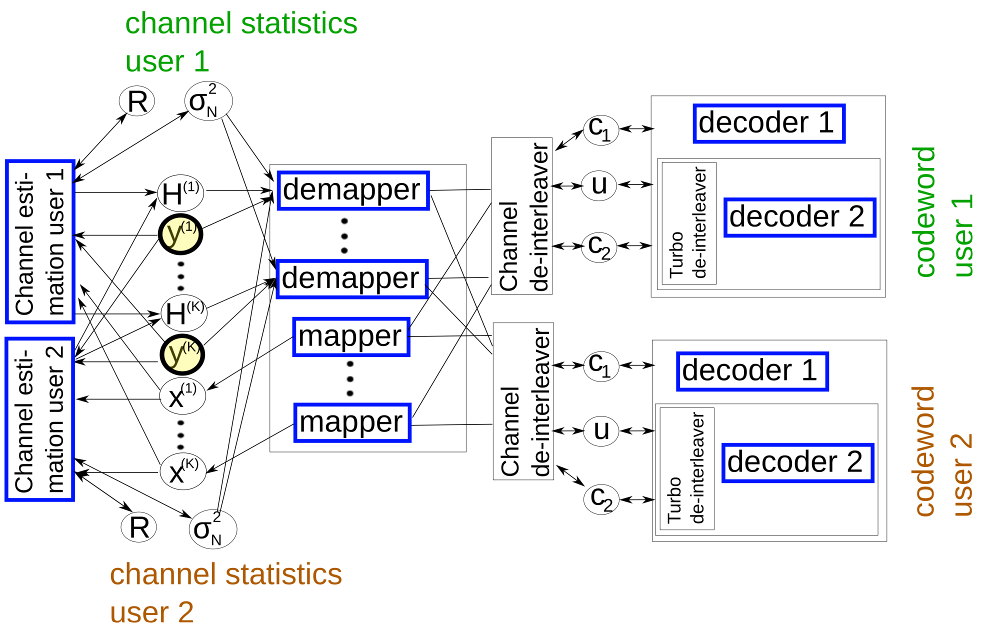


Figure 2.2: MU-MIMO: joint MIMO demapping, separate decoding. [ipa10c]

2.3 Receiver Components

This section describes the update of *factor nodes* and *variable nodes* from the generic receiver architecture (Fig. 2.1) and specifies the messages passed between them.

Factor nodes perform APP computation for the neighbouring variable nodes. APP computation is an improvement over likelihood computation, if a priori information is available (apart from the non-informative prior). Expressed in terms of the maxima of the respective densities, maximum a posteriori (MAP) is at least as accurate as maximum likelihood (ML). To avoid premature quantization, the complete densities are to be computed and exchanged. For complexity reduction in implementation, densitites are normally represented by one or a few parameters, e.g. mean APP or max. APP (Dirac), or mean and variance (Gaussian) etc. For bit variables, a complete probability density can be expressed in one number as LLR.

After the last node update, parameter estimates are then obtained by quantization: taking the mean value of the APP or its maximum (block MAP instead of symbol MAP).

MIMO Demapper Digital transmission has a discrete modulation alphabet, the i possible MIMO vectors are denoted $\mathbf{x}_{Qi}$. Although not necessary in general, for implementation the estimated density of a channel sample is assumed to be a Dirac distribution (i.e. quantized to one value). The demapper for time instance t and subcarrier j then computes and outputs for each bit b_i of the codeword fragment $\mathbf{b}^{(j,t)}$:

$$\mathbb{P}(b_i) = \sum_{\mathbf{x}_{Qi}} \mathbb{P}(b_i|\mathbf{x}_{Qi}, \mathbf{H}^{(j,t)}, \mathbf{y}^{(j,t)})\mathbb{P}(\mathbf{x}_{Qi}) = \sum_{\mathbf{x}_{Qi}} \mathbb{P}(b_i|\mathbf{x}_{Qi}, \mathbf{H}^{(j,t)}, \mathbf{y}^{(j,t)}) \prod \mathbb{P}(b_i) \tag{2.2}$$

The bit density is conditioned on all valid modulation vectors, where a priori probability information about the modulation vectors (from a priori bit probabilities) is used. The non-informative prior would be the uniform distribution. There are M^n possible transmit vectors when transmitting with modulation constellation size M on n antennas ($n \log_2 M$ bits in $\mathbf{b}^{(j,t)}$).

Constituent Decoder The decoder uses a priori codeword probabilities (from a priori bit probabilities) and conditions on all valid codewords v from the codebook V. Decoder 1 computes and outputs for each bit b_i of $\mathbf{u}$ and $\mathbf{c_1}$:

$$\mathbb{P}(b_i) = \sum_{v \in V} \mathbb{P}(b_i|v)\mathbb{P}(v) = \sum_{v \in V} \mathbb{P}(b_i|v) \prod \mathbb{P}(b_i)$$

There are 2^K valid codewords for a transmission with K information bits, where K is the length of $\mathbf{u}$. The computation can be reformulated as summation over states of the code trellis and efficiently implemented using the BCJR algorithm [BCJR74]. Decoder 2 makes a corresponding computation involving a summation over the valid codewords of constituent code 2.

Soft Mapper The soft mapper computes the probability density of a transmit symbol (vector) from bit probabilities. Conditioning is therefore on all possible bit combinations of the length corresponding to one symbol (vector).

$$\mathbb{P}(x) = \sum_{\mathbf{b}} \mathbb{P}(\mathbf{x}|\mathbf{b})\mathbb{P}(\mathbf{b}) \tag{2.3}$$

For pilot positions soft mapping is not needed, pilot symbols have a priori known Dirac distributions.

Channel Estimator The channel estimator estimates the probability density of channel samples $\mathbf{H}^{(j,t)}$, using received values within a surrounding estimation area – e.g. covering the complete area where the codeword is transmitted – and also uses probabilities for transmit symbols within this area:

$$\mathbb{P}(\mathbf{H}^{(j,t)}|\mathbf{Y}_{\mathcal{A}}) = \int_{\mathbf{X}_{\mathcal{A}}} \mathbb{P}(\mathbf{H}^{(j,t)})|\mathbf{Y}_{\mathcal{A}}, \mathbf{X}_{\mathcal{A}})\mathbb{P}(\mathbf{X}_{\mathcal{A}}) \tag{2.4}$$

where $\mathcal{A}$ is the estimation area including neighbouring symbols from position $(j.t)$. $\mathbf{Y}_{\mathcal{A}}$ and $\mathbf{X}_{\mathcal{A}}$ are the random vectors of transmit symbols x and received values y inside $\mathcal{A}$. The integral is used in the formula to cope for the general case where the densities of transmit symbols are not modelled as Dirac distributed by the receiver. The model for the conditional probability normally uses channel autocorrelation R and noise variance σ_N^2, so that noise reduction Wiener filtering [Li00] can be performed.

Variable nodes For illustration of variable node update processing, Fig. 2.3 is used, which depicts message passing of LLR vectors for iterative demapping-decoding. An LLR $L(c)$ is equivalent to the probability distribution of the corresponding bit c:

$$L(c) = \ln \frac{\mathbb{P}(c = +1)}{\mathbb{P}(c = -1)}, \tag{2.5}$$

The inverse relations between LLR and the probability of the bit being positive or negative are:

$$\mathbb{P}(c = \pm 1) = \frac{e^{\pm L(c)/2}}{e^{+L(c)/2} + e^{-L(c)/2}}. \tag{2.6}$$

The nodes of vector variables $\mathbf{c_1}$ and $\mathbf{c_2}$ have links to two factor nodes and simply forward messages, e.g. $L_a^{(dec1)}(\mathbf{c_1}) = L_e^{(det)}(\mathbf{c_1})$. Node $\mathbf{u}$ has links to three factor nodes and therefore computes an outgoing message as (elementwise) multiplication of the incoming messages from the respective two other links. In the log-APP domain the multiplication is an addition: $L_a^{(det)}(\mathbf{u}) = L_e^{(dec1)}(\mathbf{u}) + L_e^{(dec2)}(\mathbf{u})$. When the iterative update of nodes is to be stopped, the decoded message is obtained as binary quantization of $L_p(\mathbf{u}) = L_e^{(det)}(\mathbf{u}) + L_e^{(dec1)}(\mathbf{u}) + L_e^{(dec2)}(\mathbf{u})$.

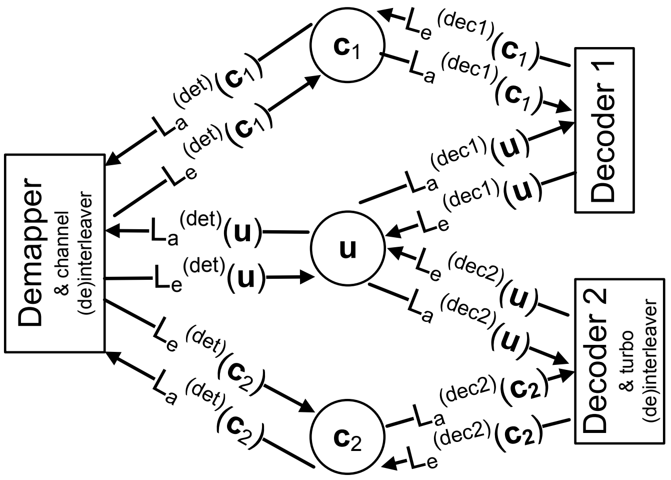


Figure 2.3: Message flow for iterative demapping-decoding consisting of LLRs [IB10].

Scheduling Factor Computation The order of node updates is arbitrary (although it does not make sense to update a node when there is no new message on an incident link - the output would remain the same). One possibility is to iteratively update all variable nodes at once and then all factor nodes at once, but for practical implementation a reduction of computational effort is wanted. The normal

MIMO receiver processing scheme for Turbo coded transmission is yielded as special case with following schedule: first update all demappers once and the variable nodes; then iteratively update decoder 1, $\mathbf{u}$, decoder 2, and $\mathbf{u}$. For iterative demapping-decoding, demapper updates can be mixed into the schedule. It is also possible to update e.g. only a subset of the demappers (not all time instances).

Chapter 3

Component Algorithms Performing Approximate APP Computation

This chapter discusses details of the receiver components (factor nodes of chapter 2), with a focus on complexity-reduced approximative APP computation. Several algorithms are described and compared to characterize the accuracy/complexity tradeoff of each component. A second focus lies on accuracy improvement with growing a priori information.

3.1 Channel Estimator

The algorithms in this section are based on Wiener filtering. Compared to the standard approach (Sec. 1.4) there are two differences: filtering is adaptive to changing channel statistics, and channel estimation is soft data-aided. For a non-informative prior of data symbols, channel estimation complexity can be considerably reduced – therefore section 3.1.2 describes ML channel estimation for the first iteration, and section 3.1.3 describes the APP version for later updates.

3.1.1 Channel Statistics Estimation and Tracking

For adaptive Wiener noise suppression filtering both in ML and APP channel estimation, channel and noise covariances need to be estimated and tracked. The channel statistics are estimated here from pilot positions. In this formulation, channel and noise statistics parameters are not updated by Bayesian inference (it is of course possible to extend the estimation to also use data positions). The adaptive filtering is based on parametric tracking of multidimensional correlation of the MIMO-OFDM channel transfer function and SNR (in uplink per user). Initialization assumes minimum (’worst-case’) pilot correlation (robust filtering as in 1.4).

Parametric multidimensional PSD tracking Motivation for adaptive filtering is to exploit as much correlation between pilots as possible. This means that the channel autocorrelation is to be estimated and tracked. Fig. 3.1 illustrates differences between available time direction correlation for different user velocities, assuming 2.1 GHz carrier frequency and maximum velocity of 400 km/h (worst case, e.g. high-speed train).

Channel correlation can be estimated directly or using the power spectral density (Sec. 1.3). Here the latter approach is chosen. In uplink, there are also remaining uncomponsated synchronization errors (per-user time and frequency offsets), which in difference to the downlink cannot be precompensated by the receiver before FFT. [JL07] uses a 2-parametric PSD model, assuming Gauss spectra for delay spread and Doppler spread. Adding the two synchronization offsets, here a 4-parametric PSD model is tracked. [YA08] points out that synchronization errors may bias spreading estimation, which is a motivation to estimate these parameters jointly or at least independently. The 2D model (for OFDM) is illustrated in Fig. 3.2, assuming uniform spectra. The width of the window in Fig. 3.2 indicates delay spread, the height indicates Doppler spread and the location of the point C shows time and frequency offset (shifting the spectrum). The number of parameters to describe the PSD can of course be increased (e.g. using distribution moments), which increases necessary estimation effort. There is a tradeoff between estimation accuracy and estimation time, which is essential for tracking a time-varying spectrum.

A block diagram of the resulting (multi-user) uplink adaptive channel estimation is shown in Fig. 3.3. For complexity reduction, synchroniation offset compensation (by phase rotations) and noise reduction filtering are separated, which allows for pre-compensation of a small set of filter kernels (the filtering approach is detailled in Sec. 3.1.2 and 3.1.3).

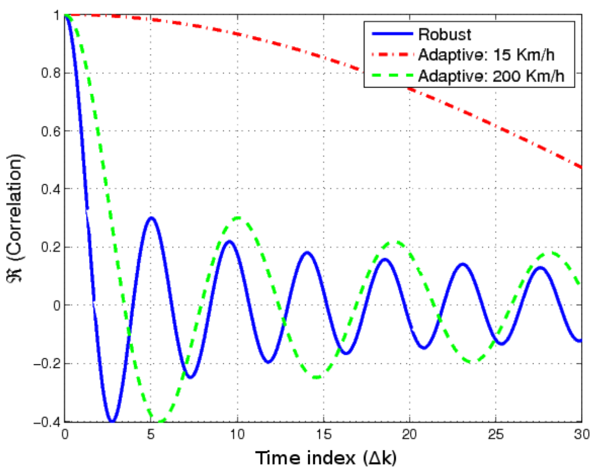


Figure 3.1: Time direction channel autocorrelation for different velocities [KID10].

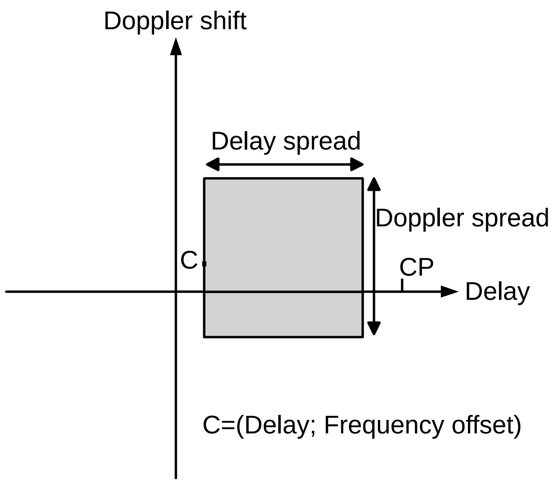
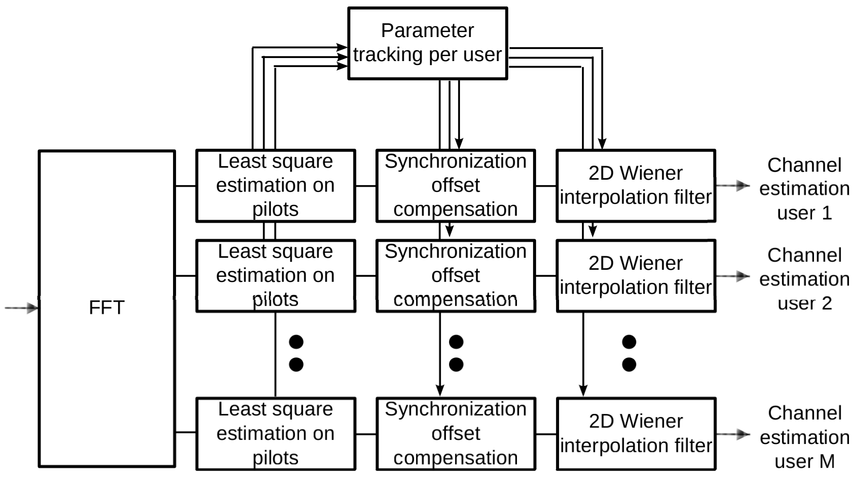


Figure 3.2: 4-parametric model of 2D power spectral density, assuming uniform distribution [KID10].



Parameter tracking per user
FFT
Least square estimation on pilots
Synchronization offset compensation
2D Wiener interpolation filter
Channel estimation user 1
Channel estimation user 2
Channel estimation user M

Figure 3.3: Uplink channel statistics tracking and adaptive Wiener interpolation filtering [KID10].

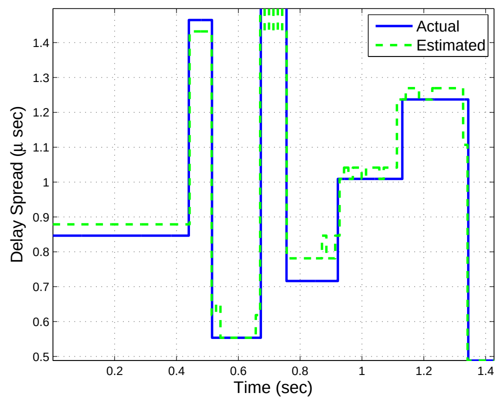
Delay and delay spread estimation There are different approaches for delay and delay spread estimation, e.g. [MZB00, YLCC00, AJ04, WKP01]. Here, the begin and end of delay spread are determined by thresholding in an estimate of the power delay profile [KID10]. Beginning with the LS channel estimation samples, windowing in frequency direction is applied to avoid leakage by Fourier transform from frequency domain to delay domain. The PDP estimate is averaged using a sliding window in time direction over several OFDM symbols.

Frequency offset and Doppler spread estimation References for Doppler spread estimation include [YA08, SL03, JL07]. The time-direction estimation window should be longer than the channel coherence time. Analogeous to delay and delay spread estimation, frequency offset and Doppler spread could be estimated by thresholding in the power Doppler profile. Here, the direct approach using the (absolute value of) time-direction autocorrelation (using LS channel estimation samples) is taken. [SL03] searches for the first zero crossing, [JL07] for the minimum extreme value. To also yield a result in the case of Gaussian correlation shape (Gaussian spectrum), here the 10% crossing has been chosen. Frequency offset estimation uses the maximum ratio combing formula as in Sec. 1.4 (Eq. 1.32).

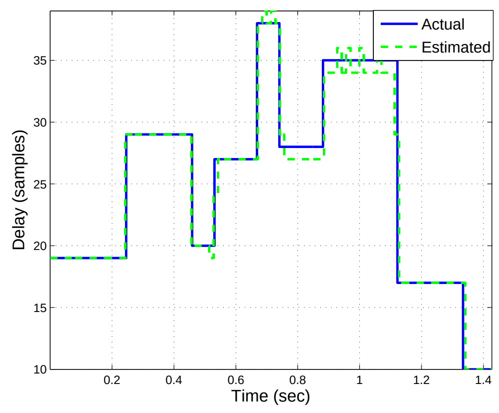
MIMO correlation estimation Either parameters of power angular spread are tracked (power angular spectrum, would fit nicely into the picture of multidimensional PSD tracking), or directly the correlation between elements of the MIMO matrix are estimated.

SNR Estimation SNR is estimated as in [JL07] from the power delay profile, with the assumption that signal energy is contained within a delay equal to cyclic prefix length, and noise energy is spread over the complete OFDM symbol duration.

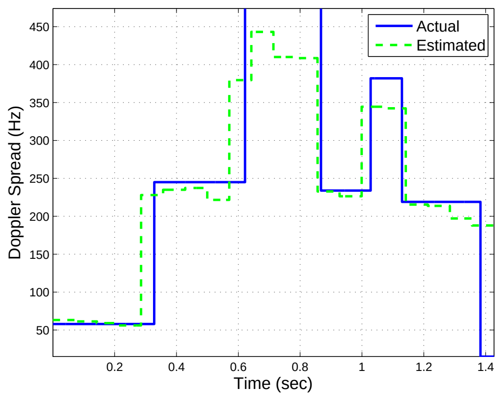
Illustration Tracking of time-varying OFDM channel statistics is illustrated in Fig. 3.4. Simulation parameters are 20 MHz bandwidth, 1200 used subcarriers of FFT length 2048, with subcarrier spacing 15 kHz. A 2D rectangular pilot grid with 4 resource elements spacing between pilots is assumed. Observation window length is 70 ms, only for Fig. 3.4e a smaller window of 10 ms is used. Fig. 3.4d and Fig. 3.4e illustrate the tradeoff between estimation variance for slowly varying channels (long window performs better) and filter adaptation delay for fast varying parameters (small window performs better).



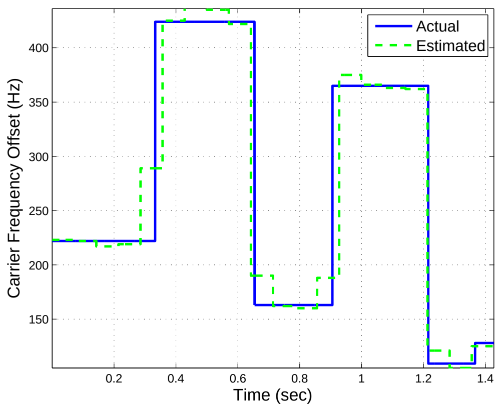
(a) Delay spread tracking.



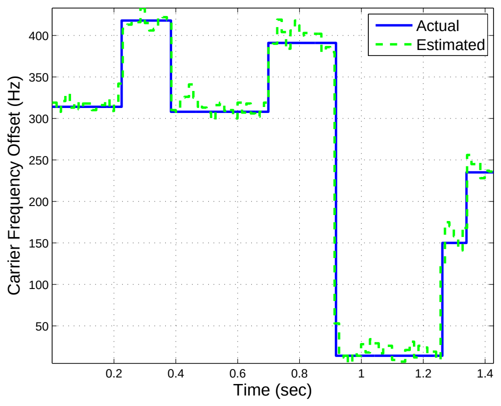
(b) Delay tracking.



(c) Doppler spread tracking (non-causal window, large processing delay).



(d) Carrier frequeny offset tracking (non-causal window, large processing delay).



(e) CFO tracking with reduced window length.

Figure 3.4: Channel statistics parameter tracking [KID10].

3.1.2 Approximate ML for First Iteration

This subsection describes a complexity-reduced shift-invariant implementation of the Wiener interpolation filter. Computational complexity of the implementation can be considerably reduced by exploiting two properties: first, multidimensional Wiener filtering is in general non-separable, while upsampling for interpolation is separable if the sample structure is a lattice - so it is beneficial to separate the two steps [LI10b]. Second, Wiener filtering can be implemented using spectral shaping of parially overlapping multidimensional blocks (fast convolution, overlap-add or overlap-save method) [LI10b]. Accuracy and complexity are scalable by choice of Wiener filter kernel size.

In many cases, filtering can be implemented as convolution. For sampled signals, this depends on the sampling grid (which for channel estimation is the pilot grid). [MS83] showed that convolution of a (multidimensional) sampled signal is possible if the sample grid is a lattice, i.e. if the *n*-dimensional sample grid is spanned by *n* basis vectors. In particular, there are lattices corresponding to nonrectangular sampling. For computationally efficient computation of convolution, fast algorithms based on Fourier transform can be applied. Fast linear convolution can be implemented based on block-wise Fourier transforms (circular convolution) using the ’overlap-save’ or ’overlap-add’ method [OS09].

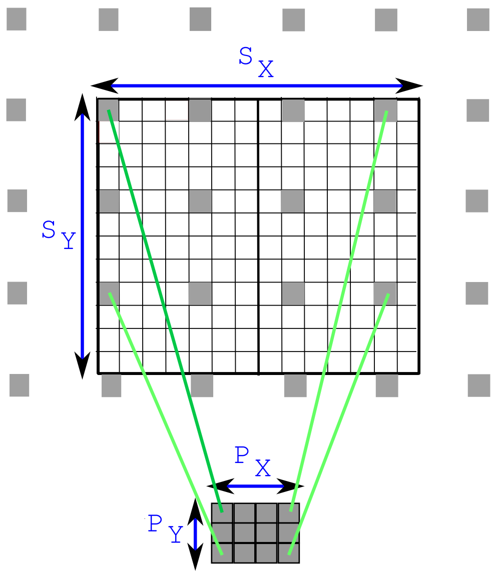


Figure 3.5: Example pilot lattice. To estimate an area of $S_x \times S_y$ values, noise reduction filtering is applied only on the $P_x \times P_y$ pilots, and upsampling afterwards [LI10b].

Published implementations of the Wiener Interpolation Filter jointly perform noise reduction and interpolation and are based on matrix multiplication [HKR97, ESB$^+$98, SJ06]. [HKR97] searches for each data symbol position for the *n* closest pilot locations based on Euclidean distance or a weighted distance measure (to select *n* pilots with large correlation values). The filter coefficients are not position independent (they differ depending on data symbol position). The estimator is hence shift-variant [HKR97]. For complexity reduction, the number *n* of pilots involved in filtering can be chosen small. For static filtering, coefficients can be precomputed (to avoid correlation matrix inversion). Also, the 2D estimator can be approximated by a concatenation of two 1D filters [HKR97] (this entails accuracy loss because while the channel autocorrelation function is separable according to model assumptions, the resulting filter is not). [ESB+98] considers 1D filtering (in frequency direction) and achieves complexity reduction (with accuracy loss) by 'Optimal Low Rank' filtering (OLR-MMSE), which is a dimension reduction by projection of pilot values onto a subspace found by singular value decomposition (SVD), before filtering is performed by matrix multiplication. For static filtering, the SVD can be precomputed. [SJ06] applies 1D Wiener Interpolation filtering (frequency direction) by matrix multiplication in a block-based way to reduce complexity (with accuracy loss): the matrix for one small block is precomputed (static filtering), and to reduce edge distortion the blocks of pilots overlap.

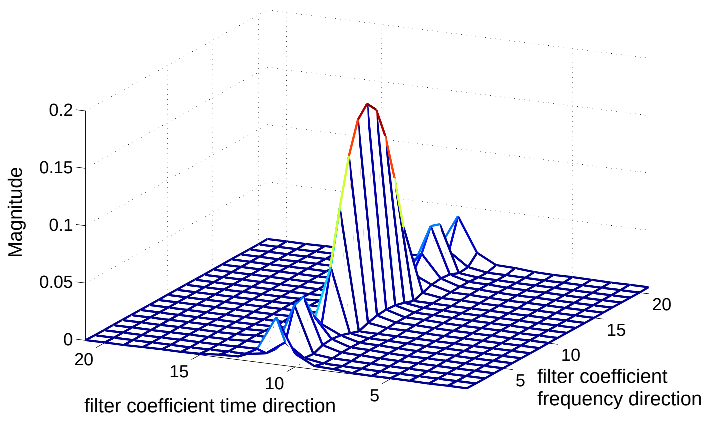


Figure 3.6: 2D convolution filter kernel magnitude for filter size 21 × 21, maximum Doppler shift 200Hz and 20dB SNR, urban macro channel model [3GP06]. High Doppler spread means little channel correlation in time direction [LI10b].

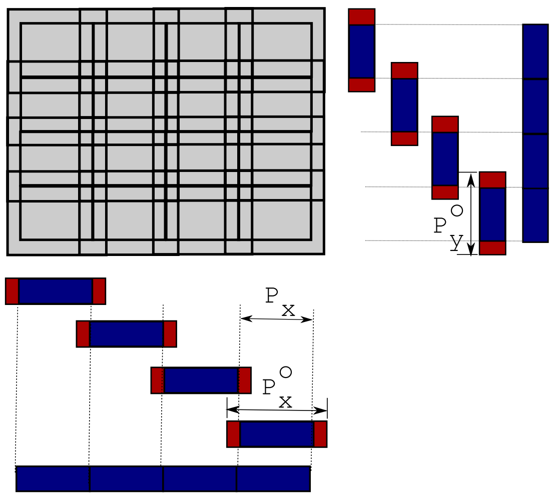


Figure 3.7: Illustration of 2D application of overlap-save method [LI10b].

Noise reduction filtering on pilots by 2D fast convolution

Here, noise reduction filtering and interpolation are separated and performed sequentially. Since the sample grid is assumed to be a lattice, multidimensional Wiener filtering only on pilot samples becomes shift-invariant and can be implemented by convolution (compare Fig. 3.5). Fast linear convolution by overlap-save (or overlap-add) method is applied, which is based on a block-wise application of the circular convolution theorem [OS09]:

$$b[n] * c[n] = \mathcal{F}^{-1}\{\mathcal{F}(b[n]) \cdot \mathcal{F}(c[n])\} \tag{3.1}$$

For computation of linear convolution by this method, zero-padding is used to guard against the overlap error compared to cyclic convolution. Two-dimensional application of the overlap-save method is illustrated in Fig. 3.7. Magnitude of a 2D complex Wiener filter kernel is illustrated in Fig. 3.6. For static filtering, the multidimensional FFT of the filter kernel can be precomputed. Adaptive filtering can be implemented by precomputing a set of filter kernels and adaptively choosing one of them.

Upsampling interpolation

In a second step, multidimensional upsampling interpolation is performed. This operation is separable [Woo06], so it can be performed sequentially for each dimension. Implementation consists of upsampling (zero stuffing) followed by 1D lowpass filtering, where again fast convolution and a precomputed FFT of the filter kernel are used. Upsampling is:

$$H_{up}(n) = \begin{cases} H(\frac{n}{L_{up}}) & \text{when } L_{up} \text{ divides } n \\ 0 & \text{else} \end{cases} \tag{3.2}$$

For an upsampling factor of L_{up} (in Fig. 3.5 L_{up} is 4), the cutoff frequency of the lowpass filter is π/L_{up}.

Complexity

Complexity of the proposed implementation for two dimensions is compared with an implementation following [HKR97, SJ06], where a precomputed filter matrix is used (static filter or adaptive selection of one of several precomputed filters). Block-wise filtering is assumed, where the channel coefficients are estimated for a block of $S_x \times S_y$ symbols, which contains $P_x \times P_y$ pilots. Compared are the number of complex multiplications necessary per estimated sample, because in signal processors additions tied to multiplications are most often computed 'for free' by means of *multiply-accumulate* instructions.

The normal implementation is a linear mapping (matrix multiplication) of the block's pilot values onto all symbols of the block. Following [SJ06], the mapping includes a few pilots from neighbouring blocks, to avoid poor accuracy at the edges. Thus, the effective 2D blocks overlap at the edges: instead of using only the $P_x \times P_y$ pilots, the matrix is a mapping from $P_x^o \times P_y^o$ pilots onto the $S_x \times S_y$ symbols and therefore needs

$$C_{\text{WIF}}^{\text{normal}} = P_x^o P_y^o \tag{3.3}$$

multiplications per estimated sample.

The proposed implementation filters on blocks of pilot symbols with the same block size $P_x^o \times P_y^o$ (including overlap, but not including data positions). Multidimensional Fourier transformation is separable [Woo06]. For a block area (with overlap) including $2^m \times 2^k$ pilots, the 2D-FFT of the block's pilots needs $P_y^o P_x^o log_2 P_x^o + P_x^o P_y^o log_2 P_y^o$ multiplications (Cooley-Tukey implementation [CT65]). If the block dimensions are not chosen as power-of-2 values, the prime factor FFT can be applied [Goo58]. The FFT of the filter kernel is assumed to be precomputed, where zeros were stuffed at the edges to fill the block size and to avoid wrap-around. After transformation, $P_x^o P_y^o$ multiplications are needed in Fourier domain. IFFT needs the same number of multiplications as FFT, so for pilot filtering complexity there is in sum:

$$C_{\text{filter}} = \frac{2\left(P_y^o P_x^o log_2 P_x^o + P_x^o P_y^o log_2 P_y^o\right) + P_x^o P_y^o}{S_x S_y} \tag{3.4}$$

Upsampling interpolation by zero stuffing and lowpass filtering is performed sequentially for the dimensions. The lowpass can be implemented by 1D fast convolution. Depending on the size of the lowpass filter kernel, the overlap can be different than for previous Wiener filtering. It is assumed that an area of $P_x^u \times P_y^u$ pilots (including overlap for convolution) is interpolated to a resolution of $S_x^u \times S_y^u$ samples (also including overlap), first in y-direction with

$$C_{\text{upsample}-y} = \frac{2P_x^u S_y^u \log_2 S_y^u + P_x^u S_y^u}{S_x S_y} \tag{3.5}$$

multiplications, then in x-direction with

$$C_{\text{upsample}-x} = \frac{2S_y S_x^u \log 2S_x^u + S_y S_x^u}{S_x S_y} \tag{3.6}$$

multiplications. For the proposed Wiener interpolation filter implementation the multiplications per output sample are yielded as:

$$C_{\text{WIF}}^{\text{proposed}} = C_{\text{filter}} + C_{\text{upsample}-y} + C_{\text{upsample}-x} \tag{3.7}$$

Using the same pilot grid as in Sec. 3.1.2 (2D rectangular, 4 resource elements pilot spacing), a comparison of the complexity of normal and proposed implementation for different filter sizes is shown in Fig. 3.8. Due to the different complexity growth orders, the gain of the proposed implementation quickly grows with larger filter size. A breakdown of the complexity of the proposed implementation into the three contributing parts from Eq. (3.7) is shown in Fig. 3.9 for different filter sizes. In the presented example, the complexity of the proposed WIF implementation in mainly determined by upsampling interpolation. This is due to the fact that an upsampling interpolation filter suited for general-purpose signal processors is chosen. In case of hardware acceleration by a specialized co-processor, further complexity reduction is possible by using a different lowpass filter. With a cascaded-integrator-comb (CIC) filter, upsampling interpolation is possible without any multiplications, only using additions [OS09].

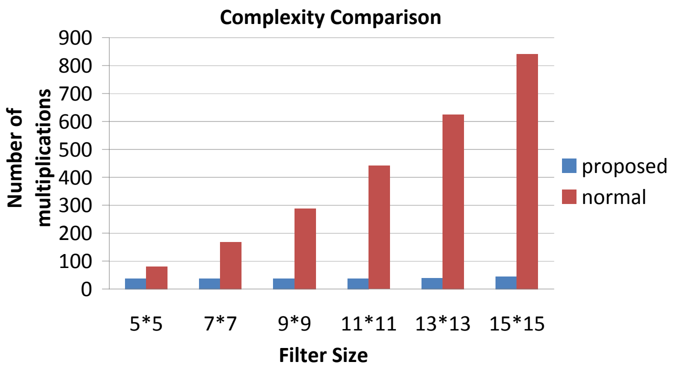


Figure 3.8: Complexity comparison for the example pilot grid (Fig. 3.5). The proposed implementation reduces absolute complexity as well as complexity growth order with respect to filter size [LI10b].

Accuracy illustration

The choice of block length in time direction needs to consider channel estimation delay (for both the normal and the proposed implementation). A combination of the proposed implementation with the matrix multiplication based one is also possible: edge blocks where there are no pilots from neighbouring blocks available, could be estimated using a precomputed matrix (at the edges of frequency band or at the edges of resource allocations in frequency direction for OFDMA uplink). The limitation of filter kernel size can employ an appropriately smooth windowing function (e.g. Hamming window [OS09]). The filter kernel size can be chosen adaptive to delay spread and Doppler spread (adaptive filtering, e.g. rectangular but non-quadratic filter kernel).

Fig. 3.10 illustrates the channel estimation MSE versus SNR curves for perfect adaptive LMMSE filtering (perfect parameter tracking), adaptation with some mismatch (errors in parameter tracking and differences between actual and modelled spectrum assumptions), robust filtering (assumed Doppler spread of 700 Hz) and least squares channel estimation.

In [Aue09] the spatial correlation is exploited for improved noise reduction by 3D Wiener filtering (robust static filtering).

Fig. 3.11 illustrates the maximum achievable channel estimation gain exploiting MIMO correlation. For full correlation (in the figure assumed for time, frequency and space), doubling the number of pilots covered by the filter are (into any of the directions) reduces the MSE by 3dB.

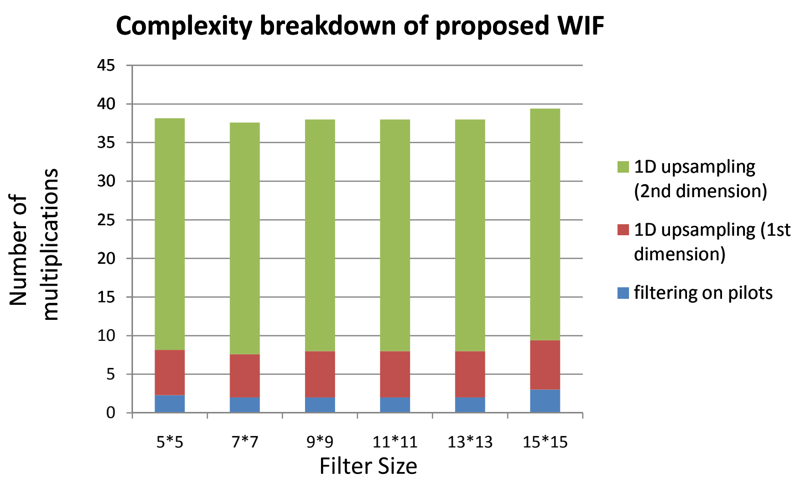


Figure 3.9: Complexity of the proposed implementation in the example is mainly determined by upsampling interpolation. Further complexity scaling is possible by choice of the 1D lowpass filter [LI10b].

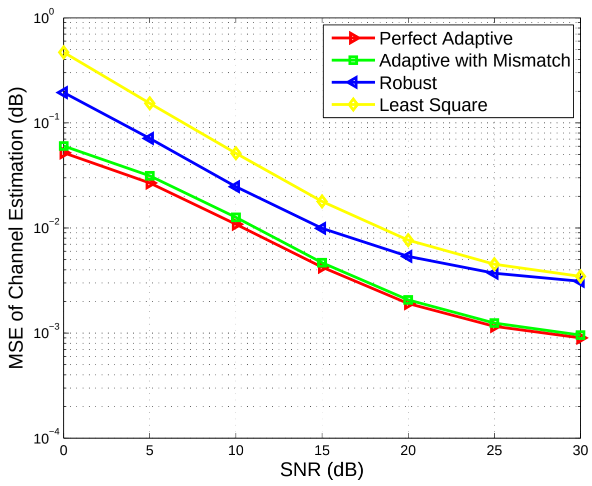


Figure 3.10: MSE comparison [KID10].

3.1.3 Approximate APP: Soft Data-Aided Channel estimation

APP channel estimation also uses data positions for improved estimation accuracy (also called ’semi-blind’ estimation). Data symbols are uncertain at the receiver, their probabilities are obtained from bit LLRs by a soft mapper. This subsection describes two APP channel estimation algorithms, which differ in the density model of uncertain data symbols. The first subsubsection describes an algorithm suited for OFDM APP channel estimation, which uses a Gaussian transmit symbol pdf model and includes ’soft symbol noise’ into a Wiener noise reduction filter. The second subsubsection describes an algorithm suited for MIMO-OFDM APP channel estimation, which also includes (MIMO-) interference. The algorithm uses discrete pdf models for data symbols and for interfering symbols. Gaussian parameters

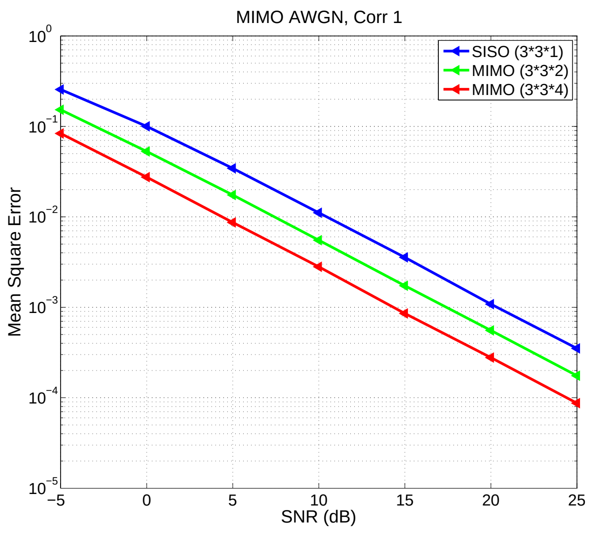


Figure 3.11: 3D ML channel estimation [Li10a].

(first two moments) of channel sample pdf estimates are then derived to enable Wiener noise reduction filtering.

Many references deal with APP channel estimation. Soft data aided channel tracking with a Kalman filter is described in [SK08]. APP channel estimation with a joint Wiener noise reduction filter for pilot and data positions is performed in [SJS03] and [SDU06], where they assume the same noise and correlation between samples, independent of whether the samples are on data or pilot positions. Usage of the expectation-maximization (EM) algorithm for channel estimation is considered in [ONSS08] for an uncoded system. EM-based joint channel estimation and detection in a formulation as Gaussian message passing is described for single-carrier transmission in [GH11]. Iterative blind channel estimation and detection based on EM for CDMA is employed in [WMK08]. Joint channel estimation and detection for OFDM with the space-alternating generalized EM (SAGE) algorithm and discrete cosine transform based dimension reduction is proposed in [PSP10]. A variational Bayesian SAGE algorithm with sparsity prior distributions is used in [SF11] to estimate the number of relevant multipath channel components and their parameters. Semi-blind channel estimation for MIMO-OFDM with a weighted linear prediction based blind criterion in the least squares (LS) approach is described in [WZS08]. Channel estimation with the EM algorithm is performed in [KB06] separately for pilot and data positions, and the estimates are combined afterwards. In [ORP05], probability based grouping of symbols and interpolation based on discrete Fourier transform is proposed. Thresholding of a-priori probabilities is proposed in [KBH06] to only use 'reliable' symbol positions, which can be seen as list-based processing.

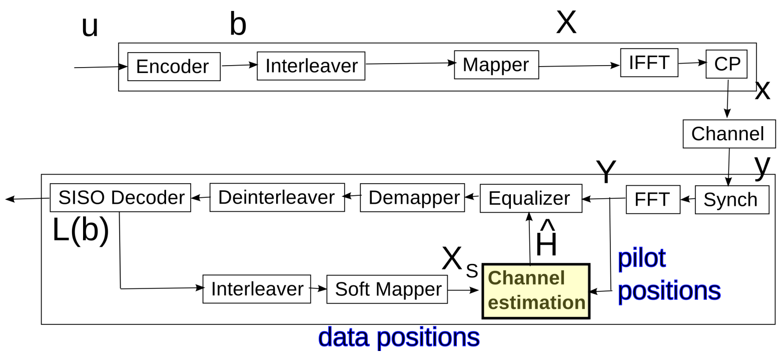


Figure 3.12: Application of iterative channel estimation in iterative receiver [LI].

APP Estimation for OFDMA: Gaussian Densitiy Model

In this subsubsection, the model used in [SJS03] and [SDU06] is extended to account for per-symbol noise enhancement due to uncertainty about transmit data symbols. From the soft mapper, normally only the mean of the transmit symbol density is used as 'soft symbol':

$$X_S(m,n) = \mathbb{E}[X(m,n)] = \int_X \mathbb{P}(X(m,n))dX \quad , \tag{3.8}$$

where an integral is used for the general case instead of a sum, to leave open the transmit symbol density model $\mathbb{P}(X(m,n))$ at this point. APP channel estimation in the general formulation computes

$$\mathbb{P}(H(m,n)|\mathbf{Y}_{\mathcal{A}}) = \int_{\mathbf{X}_{\mathcal{A}}} \mathbb{P}(H(m,n)|\mathbf{Y}_{\mathcal{A}},\mathbf{X}_{\mathcal{A}})\mathbb{P}(\mathbf{X}_{\mathcal{A}})d\mathbf{X}_{\mathcal{A}} \quad , \tag{3.9}$$

where $\mathcal{A}$ is an area around (m,n), including neighbouring symbols. $\mathbf{Y}_{\mathcal{A}}$ and $\mathbf{X}_{\mathcal{A}}$ contain the random variables $X(i,j)$ and $Y(k,l)$, with i,j and k,l inside $\mathcal{A}$. The frame for a channel estimation algorithm is defined by choosing the density models $\mathbb{P}(\mathbf{X}_{\mathcal{A}})$ and $\mathbb{P}(H(m,n)|\mathbf{Y}_{\mathcal{A}},\mathbf{X}_{\mathcal{A}})$.

Standard approach to APP channel estimation As standard algorithm it is referred to the work [SJS03] and [SDU06]. Initial LS estimation on data positions is performed using $X_S(m,n)$ as transmit symbol hypothesis:

$$H_{LS}(m,n) = \frac{Y(m,n)}{X_S(m,n)} = \frac{X(m,n)H(m,n)}{X_S(m,n)} + \frac{N(m,n)}{X_S(m,n)} \tag{3.10}$$

Joint filtering of the $\mathbf{H}_{LS}$ on pilot and data positions is:

$$\hat{H}(m,n) = \sum_{k,l\ \in\mathcal{A}} C^*(k,l)H_{LS}(m+k,n+l) \tag{3.11}$$

When written as a vector $\mathbf{c} = \text{vec}(\mathbf{C})$, the filter coefficients can be computed as [HKR97]

$$\boldsymbol{c} = (\boldsymbol{R}_{\mathcal{A}} + \sigma_N^2 \boldsymbol{I})^{-1} \boldsymbol{r}_{\mathcal{A}} \quad , \tag{3.12}$$

where $\boldsymbol{r}_{\mathcal{A}}$ is the autocorrelation vector of the channel transfer function between the position $H(m,n)$ to be filtered and the positions $H(m+k, n+l)$ in filter range $\mathcal{A}$. $\boldsymbol{R}_{\mathcal{A}}$ is the autocorrelation matrix of channel transfer function values $H(m+k, n+l)$ in filter range (noise and channel transfer function are uncorrelated). While the enumeration order of vec($\cdot$) for elements of $\mathcal{A}$ is arbitrary in Eq. (3.12), the same enumeration order needs to be applied for $\boldsymbol{c}$, $\boldsymbol{R}_{\mathcal{A}}$ and $\boldsymbol{r}_{\mathcal{A}}$. Accuracy of this algorithm is clearly suboptimal, because for data positions the noise variance is underestimated, or correspondingly the normalized correlation is overestimated. Data positions are given too much weight.

Practical infeasibility of using exact transmit symbol densities In this section, the exact transmit symbol density model is used and filtering with position-dependent coefficients is applied. $\mathbb{P}(\mathbf{X}_{\mathcal{A}})$ in Eq. (3.9) follows a discrete distribution for digital systems, which use e.g. quadrature amplidude modulation (QAM). The modulation set is denoted as Q, and its elements as $x_Q(i)$, $i = 1, \ldots, N_Q$. For pilots, the receiver is completely sure about the symbol value, i.e., the distribution is a shifted Dirac function. For a data position, the distribution is a weighted sum of Dirac impulses at modulation set positions $\mathbf{x}_Q$. The integral in Eq. (3.9) can therefore be replaced by a sum. $\mathbf{X}_{\mathcal{A}}$ denotes one realisation of the transmit symbol random variables in $\mathcal{A}$. With independent transmit data symbols, the probability of one such realisation is $\mathbb{P}(\mathbf{X}_{\mathcal{A}}) = \prod_{k,l \in \mathcal{A}} \mathbb{P}(X_{\mathcal{A}}(m+k, n+l))$. For given $\mathbf{X}_{\mathcal{A}}$, the mean conditional probability of a channel position to estimate can be determined as:

$$\mathbb{E}[H(m,n)|\mathbf{Y}_{\mathcal{A}}, \mathbf{X}_{\mathcal{A}}] = \sum_{k,l\ \in \mathcal{A}} \tilde{C}^*(m,n,k,l,\mathbf{X}_{\mathcal{A}}) H_{LS}(m+k, n+l, X_{\mathcal{A}}(m+k, n+l)) \tag{3.13}$$

The filter coefficients as well as the unfiltered channel estimates depend on the assumed realisation $\mathbf{X}_{\mathcal{A}}$ of transmit symbols. The filter has to be applied for each possible realisation of $\mathbf{X}_{\mathcal{A}}$, and the results have to be weighted with the realisation probability, yielding the mean APP for this channel position:

$$\mathbb{E}[H(m,n)|\mathbf{Y}_{\mathcal{A}}] = \sum_{\mathbf{X}_{\mathcal{A}} \in \mathcal{X}_A} \mathbb{E}[H(m,n)|\mathbf{Y}_{\mathcal{A}}, \mathbf{X}_{\mathcal{A}}] \cdot \mathbb{P}(\mathbf{X}_{\mathcal{A}}) \quad , \tag{3.14}$$

where $\mathcal{X}_A$ is the set of all $\mathbf{X}_{\mathcal{A}}$ with non-zero probability. For a filter area with N_A positions, $N_Q^{N_A}$ filters would be applied for one channel position to estimate, which clearly is too complex for practical

Computation	Type and Number of operations
H_{LS}	1 CDIV
filter matrix:	
add covariances	N_A RADD
inversion	$\frac{1}{2}N_A^3 + \frac{1}{2}N_A^2$ RMAC
multiply $\mathbf{r}$, $\mathbf{H}_{LS}$	N_A RMAC, N_A CMAC
X_S (only for proposed APP)	4 LU, 4 RMAC, 4 CMAC
σ_S^2 (only for proposed APP)	1 CMAC, 1 RADD
$\sigma_{N_{eff}}^2$ (only for proposed APP)	1 RDIV, 2 RMAC, 1 RADD, 1 CMAC

Table 3.1: Complexity per position to estimate, using QPSK and filter area with N_A entries.

implementation.

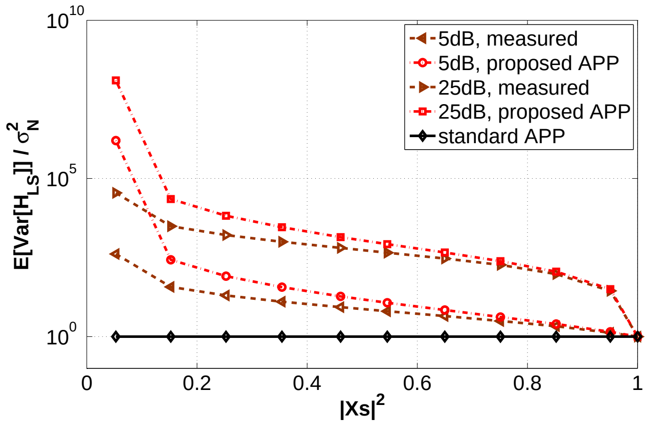


Figure 3.13: Expected noise increase factor due to uncertainty about transmit symbol, in dependence on mean soft symbol magnitude. For different channel SNRs.

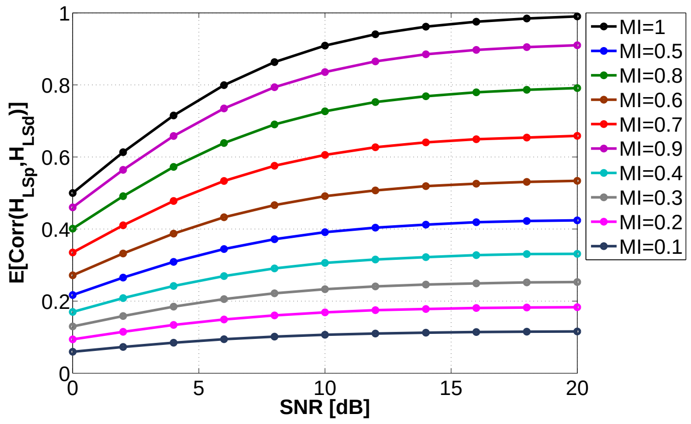


Figure 3.14: Expected normalized correlation between least squares estimates of a pilot and a data position for a flat channel, in dependence on channel noise and a priori information.

Proposed model and algorithm While [SJS03] and [SDU06] model $\mathbb{P}(X(m,n))$ as Dirac function at position $X_S(m,n)$, here a Gaussian model with mean $X_S(m,n)$ and variance $\sigma_S^2(m,n)$ is used. Both

$X_S(m,n)$ and $\sigma_S^2(m,n)$ are computed by the soft mapper from a-priori bit probabilities $\mathbf{b}_{X(m,n)}$:

$$\begin{aligned} X_S(m,n) &= \mathbb{E}[X(m,n)] = \sum_{i=1}^{N_Q} x_Q(i) \cdot \mathbb{P}(X(m,n) = x_Q(i)) && (3.15) \\ \sigma_S^2(m,n) &= \mathbb{V}[X(m,n)] = \sum_{i=1}^{N_Q} |x_Q(i) - X_S(m,n)|^2 \mathbb{P}(X(m,n) = x_Q(i)) && (3.16) \end{aligned}$$

For 4-QAM it is $\sigma_S^2(m,n) = 1 - |X_S(m,n)|^2$. Without any a-priori information, the mean soft symbol is zero and the soft symbol variance is one. Effective noise variance of least squares channel estimates on data positions is higher than on pilot positions. Soft symbol 'noise' is denoted as N_S:

$$X(m,n) = X_S(m,n) + N_S(m,n) \tag{3.17}$$

With the channel transfer on one OFDM subcarrier $Y(m,n) = H(m,n) \cdot X(m,n) + N(m,n)$, for the channel estimates before filtering it is:

$$\begin{aligned} H_{LS}(m,n) &= \frac{Y(m,n)}{X_S(m,n)} = H(m,n)\frac{X_S(m,n)}{X_S(m,n)} + H(m,n)\frac{N_S(m,n)}{X_S(m,n)} + \frac{N(m,n)}{X_S(m,n)} \\ &= H(m,n) + N_{eff}(m,n) && (3.18) \end{aligned}$$

$H_{LS}(m,n)$ is an unbiased estimate, i.e. $\mathbb{E}[H_{LS}(m,n)] = H(m,n)$. The effective noise variance before filtering is:

$$\begin{aligned} \sigma_{N_{eff}}^2(m,n) &= \mathbb{V}[H_{LS}(m,n)] = \mathbb{V}[N_{eff}(m,n)] = \frac{\sigma_N^2}{|X_S(m,n)|^2} + \frac{\sigma_S^2(m,n)}{|X_S(m,n)|^2}|H(m,n)|^2 \\ &\approx \frac{\sigma_N^2}{|X_S(m,n)|^2} + \frac{\sigma_S^2(m,n)}{|X_S(m,n)|^2}|H_{LS}(m,n)|^2 && (3.19) \end{aligned}$$

The effective noise variance depends on soft symbol magnitude, soft symbol variance and (approximately) the unfiltered channel estimate, and is therefore position-dependent. Computation of the position-dependent filter coefficients is

$$\boldsymbol{c}(m,n) = (\boldsymbol{R}_{\mathcal{A}} + \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathcal{A},N_{eff}}(m,n)))^{-1}\boldsymbol{r}_{\mathcal{A}} \quad , \tag{3.20}$$

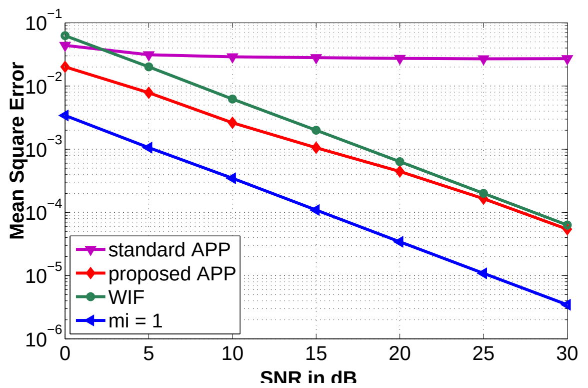
where $\mathrm{diag}(\boldsymbol{\sigma}^2_{\mathcal{A},N_{eff}}(m,n))$ is the diagonal matrix containing the effective noise variance vector $\boldsymbol{\sigma}^2_{\mathcal{A},N_{eff}}(m,n)$ of the positions in filter area $\mathcal{A}$.

Illustrations This paragraph illustrates effects of data symbol uncertainty and channel noise. Effective noise variance is illustrated in Fig. 3.13. The figure shows the ratio of the expected effective noise variance and channel noise variance only, in dependence on soft symbol magnitude. The figure contains simulation results as well as predictions according to Eq. (3.19) for different channel SNR values. The assumption of the standard APP model is also shown. While the standard model underestimates variance of channel estimates on data positions, the proposed model overestimates it. For very reliable soft symbols (magnitude close to 1, almost like pilots), the models converge. The reduction factor β of normalized correlation of LS estimates compared to that of the channel is considered:

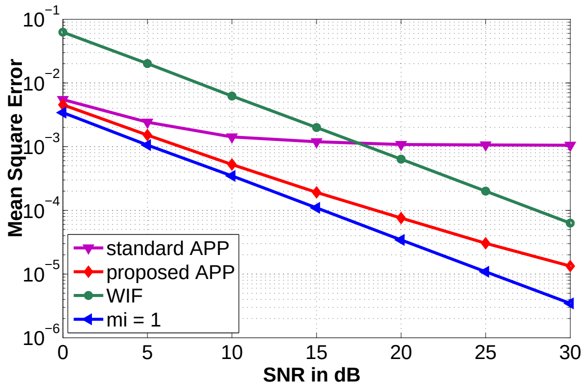
$$\beta(m_1, n_1, m_2, n_2) = \frac{\mathbb{E}[H_{LS}(m_1, n_1) H^*_{LS}(m_2, n_2)]/P_{H_{LS}}}{\mathbb{E}[H(m_1, n_1) H^*(m_2, n_2)]/P_H} , \tag{3.21}$$

where P_H is the average channel power. Apart from channel noise, the normalized correlation between $H_{LS}(m_1, n_1)$ and $H_{LS}(m_2, n_2)$ also depends on the magnitude of the two soft symbols $X_s(m_1, n_1)$ and $X_s(m_2, n_2)$, their soft symbol variances and the magnitudes of the noise-free channel samples themselves. Reduction with channel noise is illustrated in Fig. 3.14 for varying SNR and different a-priori MI values. Simulations use the common 1-parametric conditional Gaussian distribution assumption for a-priori LLRs from [ten01].

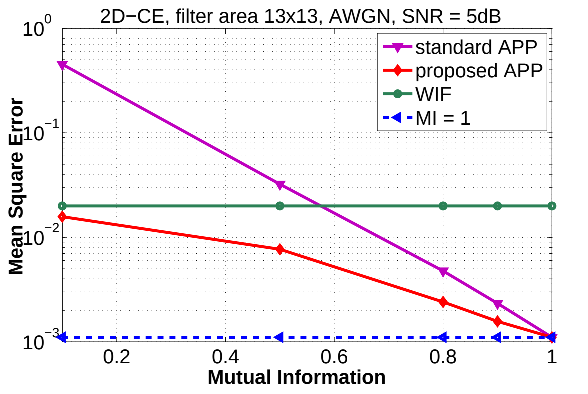
Estimation Accuracy Filter performance is evaluated simulatively by measuring channel estimation mean square error (MSE) for different values of channel noise and a-priori MI. Parameters of OFDM transmission are FFT length 2048, of which 1200 subcarriers are used, and 15kHz subcarrier distance (20MHz bandwidth). For the simulations, a rectangular filter area $\mathcal{A}$ of 13×13 symbols is used, including 3×3 pilots, and AWGN channel. Fig. 3.15a and Fig. 3.15b compare the MSE over SNR for WIF (using only pilots), the standard method and the proposed algorithm. As upper bound, also the hypothetical OFDM system where only pilots are transmitted (highest possible pilot density, no data symbols) is included, corresponding to MI $= 1$. While the standard APP algorithm outperforms the WIF only for high a-priori MI and low channel SNR, accuracy of the proposed algorithm is always equal to or better than the WIF. Due to the higher 'pilot' density, the hypothetical 'full MI' system compared to the WIF achieves an accuracy which is $10 \log \frac{13^2}{3^2} = 13$dB better for this example filter area, pilot grid and channel correlation. Fig. 3.15d shows the MSE of the different algorithms in dependence on a-priori MI. Here again it is evident that the standard APP algorithm needs a threshold a-priori MI to offer an improvement over the less complex WIF. The proposed algorithm outperforms the WIF even at very low a-priori MI, and shows better accuracy than the standard APP algorithm for the complete parameter range. Only for



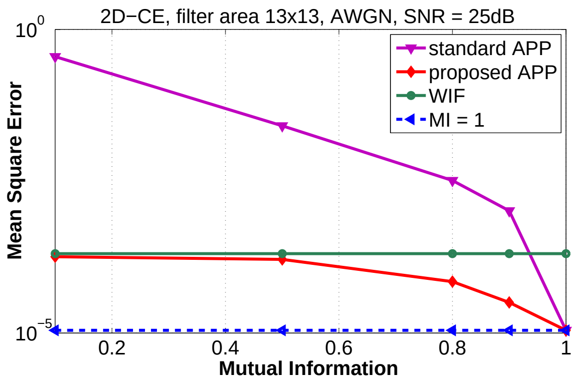
(a) Channel estimation accuracy with MI = 0.5 over varying channel SNR.



(b) Channel estimation accuracy with MI = 0.9 over varying channel SNR.



(c) Channel estimation accuracy with SNR = 5dB over varying a priori MI.

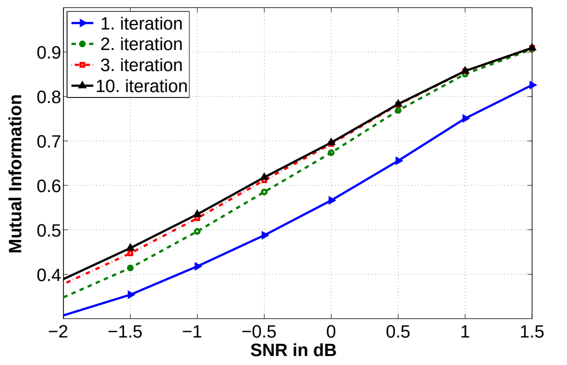


(d) Channel estimation accuracy with SNR = 25dB over varying a priori MI.

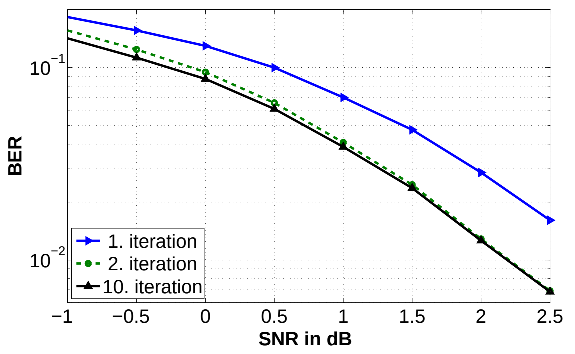
Figure 3.15: APP channel estimation accuracy [LI].

very high a-priori MI, the two APP algorithms converge to the same solution and accuracy ('full MI').

Complexity The complexity of the proposed method is compared to that of the 2-dimensional WIF. It is not compare to the standard APP method, as that is more complex and often less accurate than the WIF. If the channel is to be estimated for a small filter area within a larger resource allocation used for transmission, the WIF could be implemented with 2D fast convolution and separate upsampling [LI10b]. But on the boundaries of the resources allocated for transmission, convolution would introduce distortion. Therefore the complexity is compared to that of the general, i.e. shift-variant WIF implementation as in [HKR97]. Complexity is measured by the type and amount of elementary operations necessary to compute one filter output sample. The complexity break-down is listed in Tab. 3.2 for a filter area using N_A positions, which may be pilots and/or data symbols. Listed are real-valued and complex-valued multiply-accumulate instructions (RMAC and CMAC), divisions (RDIV, CDIV), table look-ups (LU) and additions (RADD). Tab. 3.2 assumes that matrix inversion is performed using the Bauer-Reinsch algorithm [BR70]. Any search effort for the positions to include in the WIF is neglected in the



(a) Mutual information of LLRs after decoder for different number of iterations.



(b) Bit error rate after decoder for different number of iterations.

Figure 3.16: Application example in iterative receiver [LI].

comparison. Both for the WIF and for the proposed APP method, complexity and accuracy can be scaled by chosing the size of the filter area. The WIF can be seen as the special case of APP where only pilots are used. For data positions the APP method needs an extra effort which consists of the soft mapper and computation of effective noise variance. For both estimators, the complexity is mainly defined by the matrix inversion. Considering e.g. a 4×4 filter area ($N_A = 16$ entries), filtering requires only 2% more operations if all entries are data symbols compared to when all 16 entries are pilots (counting 4 real-valued operations for a complex one). Rather than distinguishing between pilot positions and data positions, the used number of filter coefficients can be seen as limiting factor for channel estimation accuracy. The filter should use the positions with most normalized correlation between their LS channel estimates. Depending on pilot distance and channel fading behaviour, they may be only pilot or also data positions.

Application Example in Iterative Receiver The proposed estimation is evaluated in an example iterative receiver setup, together with the (weak) 8-state convolutional channel code with encoder transfer function $G(D) = [1; \frac{1+D^2+D^3}{1+D+D^3}]$, as depicted in Fig. 3.12. MI and bit error rate are evaluated after the channel decoder in dependence on channel SNR, for different number of iterations. Improvements over iterations are shown in Fig 3.16a and Fig. 3.16b. In this example, a channel estimation improvement of around 10dB translates into a system improvement of around 1dB in terms of MI and BER. For higher channel SNR only one iteration is enough to exploit the APP channel estimation, while for very low SNR the improvement needs several iterations. In the SNR range of interest, asymptotic accuracy is achieved with 10 iterations.

APP Estimation for MIMO-OFDMA: Discrete Density Model

APP channel estimation for MIMO-OFDM needs an accurate stochastic model in order to offer an improvement over the WIF. In addition to channel choise, data positions undergo MIMO stream interference and effective noise due to uncertainty about the transmit symbols. While for OFDM transmission good results can be obtained with a Gaussian density model for uncertain transmit symbols, this approach does not work for MIMO-OFDM. With a data symbol also in the denominator, the resulting distribution would contain a complicated Gaussian ratio distribution. A ratio distribution of independent Gaussian variables with zero mean e.g. would be a Cauchy distribution, i.e. not have an expectancy nor a variance.

Notation The notation for this subsubsection is as follows: transmit and receive antennas are enumerated as vector and matrix indices, while subcarriers and time are enumerated in brackets. For brevity, indices or brackets are sometimes omitted. The equation then holds independently for any index or bracket value. E.g. for 2x2 MIMO it is written:

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} + \begin{pmatrix} N_1 \\ N_2 \end{pmatrix} , \tag{3.22}$$

for any one time instance and subcarrier.

Algorithm The soft mapper computes

$$\mathbb{P}(X(m,n)) = \sum_{\mathbf{b}_{X(m,n)} \in \{0,1\}^{\log_2(M)}} \mathbb{P}(X(m,n)|\mathbf{b}_{X(m,n)})\mathbb{P}(\mathbf{b}_{X(m,n)}) \quad , \tag{3.23}$$

where M is the modulation alphabet size and $\mathbf{b}_{X(m,n)}$ are the bits from the codeword $\mathbf{b}$ that were mapped to symbol $X(m,n)$. LLRs are converted to probabilites according to $\mathbb{P}(b = \pm 1) = \frac{1}{1+e^{\mp L(b)}}$.
The first algorithm iteration computes the standard WIF, because there is yet no a-priori information available. The remaining noise after WIF is [HKR97]:

$$\sigma^2_{WIF} = \sigma^2_n \Big(1 - \boldsymbol{r}^T_{\mathcal{A}} (\boldsymbol{R}_{\mathcal{A}} + \sigma^2_N \boldsymbol{I})^{-1} \boldsymbol{r}_{\mathcal{A}}\Big) \tag{3.24}$$

In a later iteration after channel decoding, the LLRs are first mapped to transmit symbol distributions according to Eq. (3.23). Then LS channel estimation is performed on data positions, and expectancy and variance of channel samples at data positions are derived for later noise reduction filtering. A data transmit symbol X_j is nonzero, so the MIMO transmission equation can be solved for channel samples:

$$H_{ij} = \frac{Y_i - N_i - \sum_{k=1..N_T;k\neq j} H_{ik}X_k}{X_j} \tag{3.25}$$

The expectancy is:

$$\mathbb{E}[H_{ij}] = \mathbb{E}\Big[\frac{Y_i}{X_j}\Big] - \sum_{k=1..N_T;k\neq j} \mathbb{E}_{H_{ik}}[H_{ik}] \cdot \mathbb{E}_{X_j,X_k}\Big[\frac{X_k}{X_j}\Big] \tag{3.26}$$

The variance is approximated with the total variance formula:

$$\mathbb{V}[H_{ij}] \approx \mathbb{E}_{\mathbf{X}}\Big[\mathbb{V}[H_{ij}|\mathbf{X}]\Big] + \mathbb{V}_{\mathbf{X}}\Big[\mathbb{E}[H_{ij}|\mathbf{X}]\Big] \tag{3.27}$$

To illustrate, this is solved for H_{11} in 2x2 MIMO with QPSK modulation:

$$H_{11} = \frac{Y_1 - H_{12}X_2 - N_1}{X_1} \tag{3.28}$$

With the expectancy

$$\mathbb{E}[H_{11}] = \mathbb{E}_{X_1}\Big[\frac{Y_1}{X_1}\Big] - \mathbb{E}_{H_{12}}[H_{12}] \cdot \mathbb{E}_{X_1,X_2}\Big[\frac{X_2}{X_1}\Big] \tag{3.29}$$

For QPSK the variance is yielded:

$$\begin{aligned} \mathbb{V}[H_{11}] &\approx \mathbb{E}_{X_1,X_2}\Big[\mathbb{V}[H_{11}|X_1,X_2]\Big] + \mathbb{V}\Big[\mathbb{E}_{X_1,X_2}[H_{11}|X_1,X_2]\Big] \\ &= \mathbb{V}[H_{12}] + \sigma_N^2 + \mathbb{V}_{X_1,X_2}\Big[\frac{Y_1}{X_1} - \mathbb{E}[H_{12}] \cdot \frac{X_2}{X_1}\Big] \end{aligned} \tag{3.30}$$

The two unknown values in this equation can be approximated with values from the WIF as $\mathbb{E}[H_{12}] \approx \hat{H}_{WIF}$ and $\mathbb{V}[H_{12}] \approx \sigma^2_{WIF}$, or alternatively with values from the last iteration. Then a 2D position-dependent Wiener filter is applied jointly for all positions in filter area, i.e. for pilot and data positions. The filter is applied independently per transmit-receive antenna pair. Computation of the position-dependent filter coefficients is

$$\boldsymbol{c}(m,n) = (\boldsymbol{R}_{\mathcal{A}} + \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathcal{A}}(m,n)))^{-1}\boldsymbol{r}_{\mathcal{A}} \quad , \tag{3.31}$$

where $\mathrm{diag}(\boldsymbol{\sigma}^2_{\mathcal{A}}(m,n))$ is the diagonal matrix containing the vector of variances $\mathbb{V}[H_{i,j}](m,n)$ of the positions in filter area $\mathcal{A}$.

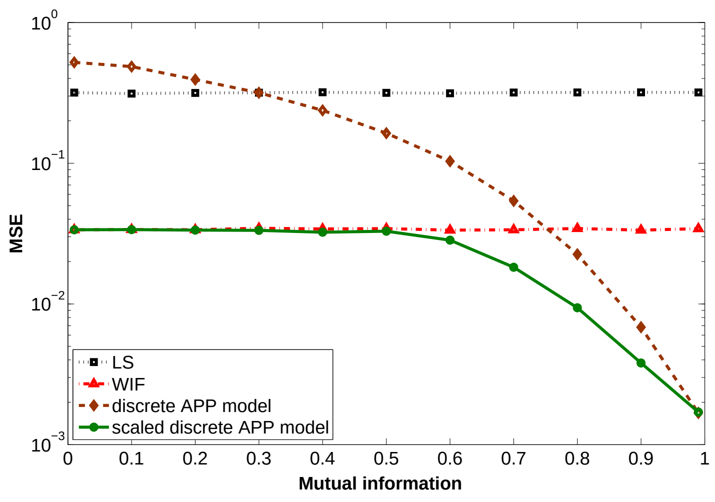


Figure 3.17: MIMO-OFDM APP channel estimation accuracy in dependence on a-priori MI.

Computation	WIF 3x3 pilots	APP 15x15 symbols
Symbol probabilities	-	8
E[h]	1	25
V[h]	-	76
scaling	-	20
filter		
inverse	18	1800
coefficients	9	225
apply	3	15
$\sum$	31	2169

Table 3.2: Complexity of APP MIMO-OFDM channel estimation: number of multiplications per position to estimate for 2x2 QPSK.

Accuracy Fig. 3.17 shows the MSE in dependence on a-priori MI for different estimators in the following transmission scenario. The scenario is 2x2 MIMO-OFDM transmission with QPSK modulation over a flat channel with full crosstalk, $H = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, at 5dB SNR. A rectangular pilot grid with pilot distance 5 is used, and the grid is offset between the transmit antennas. At the subcarrier and time where one antenna transmits a pilot, the other one does not transmit. Fig. 3.17 shows that the discrete APP model outperforms the WIF only for a-priori MI of more than 0.75. This bad accuracy at low a-priori MI is attributed to the following model mismatch. While the Wiener noise reduction filter assumes Gaussian densities, here there are actually mixture Gaussians. On the other hand the APP formulas include the WIF as special case for using only pilot positions, so it should be possible to combine the two curves for best accuracy. The heuristic solution is to scale down the weight of data positions for low a-priori MI, which is indicated by small absolute value of the expectany of the transmit data symbol. In this way, more variance is counted for uncertain data positions, in order to account for neglected higher order distribution moments. Instead of Eq. (3.30), the scaled version is used:

$$\tilde{\mathbb{V}}[H_{11}] = \frac{\mathbb{V}[H_{12}] + \mathbb{V}_{X_1,X_2}\left[\frac{Y_1}{X_1} - \mathbb{E}[H_{12}] \cdot \frac{X_2}{X_1}\right]}{|\mathbb{E}[X_1]|^{12}} + \frac{\sigma_N^2}{|\mathbb{E}[X_1]|^6} \tag{3.32}$$

The resulting curve has the same accuracy as the WIF for low a-priori MI, and has better accuracy for a-priori MI ≥ 0.4.

Complexity Complexity of the scaled APP algorithm is compared to that of the WIF, in the same scenario of Fig. 3.17. Complexity is measured as the number of multiplications necessary for one filter output sample. The APP algorithm needs extra computation to map LLRs to symbol probabilities and to compute the variance of LS estimates. A break-down is shown in Tab. 3.2. In this scenario, the number of multiplications increases by a factor of 70 when applying the APP algorithm instead of the WIF. On the other hand the table shows that this increase is mainly due to more filter coefficients, whose number increases from a $3^2 x 3^2$ WIF matrix to a $15^2 x 15^2$ APP matrix.

Notes Both estimation accuracy and complexity increase considerably. They can be scaled by choosing the size of the filter area. To reduce complexity, it is also an option to not include data symbols with small LLR magnitudes into the filtering. This is comparable to the list-based approach in [KBH06], with the difference that all used symbols are still adequately weighted. Apart from single- and multi-user MIMO, the presented algorithm is also applicable to coordinated multi-point (CoMP) transmission (distributed MIMO; interference also on pilots). For maximum filter accuracy it seems possible to extend the algorithm to 3D filter areas, to also exploit MIMO correlation for noise reduction.

3.1.4 Accuracy Increase with A Priori Information

Accuracy of APP channel estimation increases with growing a priori information from that of the WIF to that of the case where only pilots would be transmitted. The proposed algorithm thus extends the receiver accuracy/complexity tradeoff: it means higher complexity due to position-dependent filter coefficients, but offers an improvement for high-accuracy receivers. Better channel estimation accuracy can be exploited for reception at lower SNR, for higher modulation or for reduction of pilot overhead. In practical implementation it may be beneficial to use the Wiener interpolation filter in the first iteration: it is less complex with the same accuracy, as long as there is no a priori information.

3.2 MIMO Demapper

In classical non-iterative MIMO demapping, a soft-output MIMO demapper computes likelihoods of the transmit bits being 1 or 0, given the received symbol vector. In iterative MIMO demapping-decoding, detection accuracy is improved by exploiting apriori information about bit probabilities from the decoder [HtB03]. Transmit bits are viewed as random variables and the optimum demapper performs Bayesian updating of transmit bit probabilities to compute the aposteriori probabilities (APP). The iterative demapping-decoding setup ('Turbo receiver' [Hag02]) is illustrated in Fig. 3.18. The 'maximum likelihood (ML) MIMO detector' [PGNB04] is the special case of APP demapping for hard decision (binary output) for no a priori information.

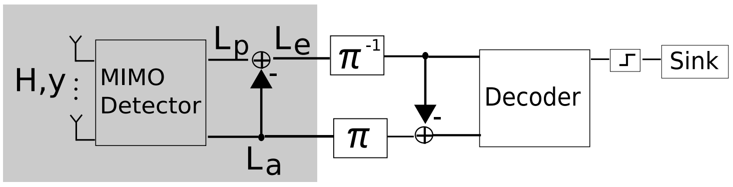


Figure 3.18: Setup for iterative MIMO demapping-decoding [IKB09].

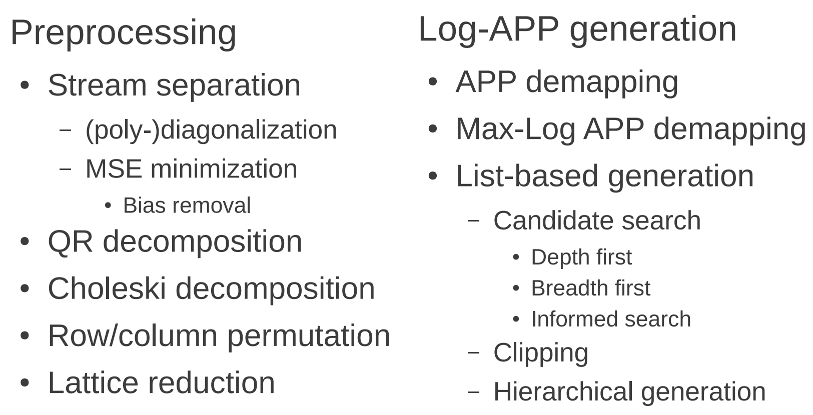
The optimal APP MIMO demapper as well as its Max-Log approximation (using 'only' the closest two candidate symbol vectors for LLR generation) are NP-hard problems [ABSS97]. This can be seen as reason why a large 'MIMO demapper zoo' of different algorithms has been developed.

Since the demapper outputs soft information, its accuracy cannot be adequately measured by bit error rates (as a bit is only the sign of an LLR). The adequate measure for demapper accuracy is the mutual information (MI) between the correct transmit bits and LLRs:

$$I(C,L) = I(L,C) = \sum_{l\in L}\sum_{c\in C}\mathbb{P}(l,c)\ln\frac{\mathbb{P}(l,c)}{\mathbb{P}_l(l)\cdot\mathbb{P}_c(c)}, \tag{3.33}$$

where $\mathbb{P}(l,c)$ is the joint distribution of transmit bits c and receiver LLRs l, and $\mathbb{P}_c(c)$ and $\mathbb{P}_l(l)$ are the marginal distributions. MI is a value between 0 and 1, where 1 means complete information (correct detection, a posteriori LLRs). In simulations, MI can be computed using histograms or (with better accuracy) Kernel density estimation [Mod89, Par62].

Many of the proposed MIMO demappers have partial algorithms in common. To give an overview, common partial algorithms are shown in Fig. 3.19. They are divided into preprocessing algorithms and Log-APP generation algorithms. These components can be freely combined to yield a specific MIMO demapper. Subsection 3.2.1 shortly describes the purposes of the preprocessing algorithms, subsection 3.2.2 those of the Log-APP generation methods, and subsection 3.2.3 presents some interesting complete demappers as composition of the parts. The algorithms in this chapter assume that the (elements of the) channel matrices **H** are input as Dirac distributions (quantized to numbers, MAP channel estimation) to the demapper.



Preprocessing

- Stream separation
  - (poly-)diagonalization
  - MSE minimization
    - Bias removal
- QR decomposition
- Choleski decomposition
- Row/column permutation
- Lattice reduction

Log-APP generation

- APP demapping
- Max-Log APP demapping
- List-based generation
  - Candidate search
    - Depth first
    - Breadth first
    - Informed search
  - Clipping
  - Hierarchical generation

Figure 3.19: MIMO demapper ’tool box’.

3.2.1 Preprocessing Transformations

Stream Separation

Stream separation (by matrix multiplication of the received vector) enables separate LLR generation (per-stream demapping).

(Poly-)Diagonalization One way of stream separation is to diagonalize H by multiplication with its inverse (zero-forcing, ZF). A negative side-effect is that the receiver AWGN also undergoes the matrix multiplication and in case of badly conditionned matrix is largely enhanced. A generalization is poly-diagonalization (remaining nonzero elements not only on main diagonal, less noise enhancement). In [YL08] this approach is combined with tail-biting trellis decoding to yield a (hard output) MIMO demapper.

Minimize MSE Stream separation according to the MMSE criterion is analog to ZF, but minimizes the remaining MSE also considering noise enhancement. This reduction of the MSE introduces bias. The ’unbiased MMSE’ [Zim07] removes the introduced bias afterwards per-stream (as described in Sec. 1.4).

QR decomposition

QR-decomposing the channel matrix enables list-based Log-APP generation by tree search algorithms (breadth first, depth first or informed search) [YKGI03]. QR-decomposition can be also applied to the extended channel matrix (regularized with noise covariance matrix) – taking channel noise into account but introducing the MMSE bias [ZF06a].

Choleski decomposition

Same as QR decomposing H, Choleski decomposition of $H^H H$ enables tree search algorithms [dJW02].

Row/column permutation

For sequential algorithms like list candidate search, the result depends on the row/column order of H. Early errors and their propagation can be reduced by first sorting according to the vector norm (best channel vector first).

Lattice Reduction

Problematic for stream separation are ill-conditioned channel matrices (noise enhancement). Lattice reduction tries to circumvent this problem by changing the basis vectors to yield an almost orthogonal base. The receiver AWGN does not undergo a linear transformation. This approach spends some effort for finding an adequate basis, normally according to the LLL algorithm [LLL82, W05̈].

3.2.2 Log-APP Ratio Generation

The three approaches considered here are correct APP demapping, demapping using the Max-Log approximation and list-based LLR generation. These methods can be applied jointly for all streams or (with reduced complexity) separately after stream separation.

Correct APP demapping

In correct APP demapping, all possible transmit vectors $\mathbf{x}$ contribute to the metric. Such a demapper computes [Hag02]:

$$L(c|\mathbf{y}) = \ln \frac{\mathbb{P}(c = +1|\mathbf{y})}{\mathbb{P}(c = -1|\mathbf{y})} = \ln \frac{\sum_{\mathbf{x}\in\mathcal{X}^{+1}} \mathbb{P}(\mathbf{y}|\mathbf{x})\mathbb{P}(\mathbf{x})}{\sum_{\mathbf{x}\in\mathcal{X}^{-1}} \mathbb{P}(\mathbf{y}|\mathbf{x})\mathbb{P}(\mathbf{x})} \tag{3.34}$$

where $\mathcal{X}^{+}$ means the set of all possible transmit vectors $\mathbf{x}$ where the bit whose LLR is to be computed has the value +1. This computation uses apriori information (from the decoder) in the form of the $\mathbb{P}(\mathbf{x})$ to compute the aposteriori LLR. In iterative processing only the extrinsic information is forwarded, which can be obtained from the aposteriori LLR by simple subtraction [HtB03]:

$$\begin{aligned}
ln\frac{\mathbb{P}(c_i = +1|\mathbf{y})}{\mathbb{P}(c_i = -1|\mathbf{y})} &= ln\frac{\sum_{\mathbf{x}\in\mathcal{X}_i^{+1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n=1}^{N_T N_B} P(c_n)}{\sum_{\mathbf{x}\in\mathcal{X}_i^{-1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n=1}^{N_T N_B} \mathbb{P}(c_n)} \\
&= ln\frac{\sum_{\mathbf{x}\in\mathcal{X}_i^{+1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n\neq i} \mathbb{P}(c_n)}{\sum_{\mathbf{x}\in\mathcal{X}_i^{-1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n\neq i} \mathbb{P}(c_n)} + ln\frac{\mathbb{P}(c_i = +1)}{\mathbb{P}(c_i = -1)} \\
\underbrace{L_p(c_i)}_{\text{aposteriori information}} &= \underbrace{L_e(c_i)}_{\text{extrinsic information}} + \underbrace{L_a(c_i)}_{\text{apriori information}}
\end{aligned} \tag{3.35}$$

where the LLR is computed for bit c_i, and there are N_T transmit antennas and N_B bits per symbol.

Max-Log APP demapping

For practical implementation, the max-log approximation is often used in the demapper as well as in the decoder (max-log-BCJR) [RVH95]:

$$ln\sum a_n \approx max(\ln(a_n)) \tag{3.36}$$

Result is that only two candidate vectors (and the apriori LLRs) contribute to an LLR, which reduces computational effort spent on computing the LLR. On the other hand, searching for each of the two Max-Log hypothesis vectors is still NP-hard [ABSS97, AEVZ02]. A further complexity reduction is often achieved by using a separable modulation set, meaning that real and imaginary components can be independently demapped (e.g. in LTE [3GP10b], compare Sec. 1.4). Applying the Max-Log approximation to the APP detector yields [RVH95, RBO04]:

$$\begin{aligned}
L_p(c_i) \approx\ & \max_{\mathbf{x}\in\mathcal{X}_i^+}\Big(\ln(\mathbb{P}(\mathbf{y}|\mathbf{x})) + \sum_n \min(c_n L_a(c_n); 0)\Big) \\
& - \max_{\mathbf{x}\in\mathcal{X}_i^-}\Big(\ln(\mathbb{P}(\mathbf{y}|\mathbf{x})) + \sum_n \min(c_n L_a(c_n); 0)\Big)
\end{aligned} \tag{3.37}$$

The max-log approximation is also applied to the mapping from LLR to probability, e.g.:

$$\ln(\mathbb{P}(c = -1)) = \ln\frac{1}{1+e^L} = \ln 1 - \ln(e^0 + e^L) \approx 0 - max(L; 0) \tag{3.38}$$

This approximation is illustrated in Fig. 3.20. Many decoder implementations use a ’softmax’ function by table lookup to soften the break in the approximation [Vog02]. Considering the noise to be Gaussian, the (Max-Log approximated) extrinsic LLR is:

$$\begin{aligned} L_e(c_i) \approx & \max_{\mathbf{x}\in\mathcal{X}_i^+}\Big(-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{x}(\mathbf{c})\|^2 + \sum_{n\neq i}\min(c_n L_a(c_n);0)\Big) \\ & -\max_{\mathbf{x}\in\mathcal{X}_i^-}\Big(-\frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{H}\mathbf{x}(\mathbf{c})\|^2 + \sum_{n\neq i}\min(c_n L_a(c_n);0)\Big) \end{aligned} \tag{3.39}$$

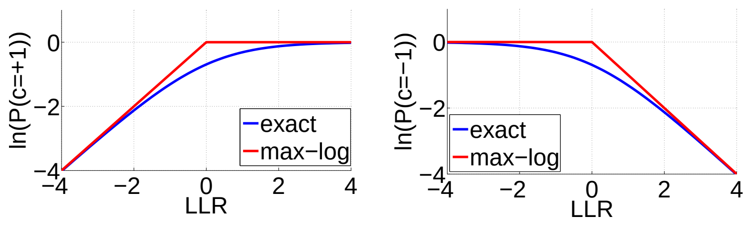


Figure 3.20: Exact and Max-Log approximated relation between LLR and bit probabilities [IKB09].

List-based generation

List based LLR generation searches for candidate vectors for the Max-Log approximation. Complexity is reduced compared to the Max-Log solution by not visiting all possible transmit vectors (not computing all metrics). An ordered search strategy (as opposed to random methods) in terms of graph algorithms is normally enabled by the preprocessing step of QR decomposing the channel matrix (or Choleski decomposition). Graph search algorithms are depth first, breadth first and informed search (special enumeration, best partial metric first) and may be combined with branch and bound (pruning the search tree). A number of found candidates with good metric is added to the list from which the LLRs are generated. An example of depth first search with branch and bound is the ’sphere decoder’ [VB99, WG04]. Special enumerations are the Fincke-Pohst [FP85] and Schnorr-Euchner [SE94] enumerations. Adaptations of the sphere decoder to use a priori probabilities are described in [SB10, LLN+09, WBA+10]. An example of breadth first search (using a priori information) for the M best candidates is the ’M-algorithm’ [JA71, RBO04], and with a variable candidate number the ’T-algorithm’ [Sim90]. Unbiasing regularized tree search (MMSE criterion) is described in [ZF06b]. An approximation for further complexity reduction in breadth-first search is ’modulation set partitioning’ [dJW02], which converts the tree into a narrower but deeper one. If one searches only for one candidate vector, the demapper is reduced to hard output (closest point search in lattice [AEVZ02]). Soft output needs the Max-Log hypothesis vector and counter-hypothesis vector per bit (for this bit being positive and negative respectively). The reduced search has two effects: first, not always the best hypothesis vectors are in the list, which leads to reduced accuracy compared to the Max-Log solution. Second, a hypothesis vector for a certain bit value may be missing. For a missing hypothesis, two approaches have been proposed: ’clipping’ the LLR to a predefined value with correct sign [HtB03], or preferably using as hypothesis also partial vector candidates whose subtrees have been pruned (hierarchical or per-level LLR generation) [WM04].

3.2.3 Demapper Synthesis

In this subsection, four example MIMO demapper algorithms are described as composition from the partial algorithms.

Joint Max-Log APP demapping

For all possible transmit vectors the metric

$$\mu(\mathbf{c}) = \frac{-1}{2\sigma^2}||\mathbf{y} - \mathbf{Hx}(\mathbf{c})||^2 + \frac{1}{2}L_a(\mathbf{c})^T\mathbf{c} \tag{3.40}$$

has to be computed. According to Eq. (3.39), to generate $L_e(c_i)$, the vector with best metric from $\mathcal{X}_i^+$ and $\mathcal{X}_i^-$ has to be found. The LLR is then their difference. Implementation aims at reusing intermediate results from metric computation and searches.

MMSE stream separation with per-stream Max-Log demapping

This is the least complex algorithm which serves as 'baseline' and has already been described in Sec. 1.4. Here a lower level algorithm for implementation is described [KI08]. It is based on the Greville algorithm and needs not more multiply-accumulate (MAC) operations than the ZF solution (using Greville). The Greville algorithm can be used to compute the ZF matrix (Moore-Penrose pseudo inverse $\mathbf{H}^\dagger$) with an iteration over the k columns of $\mathbf{H}$:

$$\vec{p}_k^H = \frac{\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{H}_{k-1}^\dagger)}{||\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{H}_{k-1}^\dagger)||^2} \tag{3.41}$$

$$\mathbf{H}_k^\dagger = \begin{cases} \frac{\vec{h}_1^H}{\vec{h}_1^H\vec{h}_1} & \text{if } k = 1 \\ \begin{bmatrix} \mathbf{H}_{k-1}^\dagger(\mathbf{I} - \vec{h}_k\vec{p}_k^H) \\ \vec{p}_k^H \end{bmatrix} & \text{else} \end{cases} \tag{3.42}$$

assuming that $\vec{p} \neq \vec{0}$. $\mathbf{H}_{k-1}$ consists of colums $\vec{h}_1$ .. $\vec{h}_{k-1}$. The MMSE matrix differs from the ZF one by an additive diagonal matrix $\mathbf{D}$ for noise regularization [ABI03] before inversion. The Greville algorithm can be modified to account for this by [KI08]:

$$\vec{p}_k^H = \frac{\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{G}_{k-1})}{||\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{G}_{k-1})||^2 + d_k^2 + ||\mathbf{D}_{k-1}\mathbf{G}_{k-1}\vec{h}_k||^2} \tag{3.43}$$

$$\mathbf{G}_k = \begin{cases} \frac{\vec{h}_1^H}{\vec{h}_1^H\vec{h}_1 + d_1^2} & \text{if } k = 1 \\ \begin{bmatrix} \mathbf{G}_{k-1}(\mathbf{I} - \vec{h}_k\vec{p}_k^H) \\ \vec{p}_k^H \end{bmatrix} & \text{else} \end{cases} \tag{3.44}$$

Where only the denominator of $\vec{p}_k^H$ differs from the original Greville algorithm in Eq. (3.41). The first good property of this algorithm is its low complexity. Rewriting (3.43) for better reuse of intermediate results gives

$$\begin{aligned} \vec{p}_k^H &= \frac{\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{G}_{k-1})}{||\vec{h}_k^H(\mathbf{I} - \mathbf{H}_{k-1}\mathbf{G}_{k-1})||^2 + d_k^2 + ||\mathbf{D}_{k-1}\mathbf{G}_{k-1}\vec{h}_k||^2} \\ &= \frac{\vec{h}_k^H - (\mathbf{G}_{k-1}\vec{h}_k)^H\mathbf{H}_{k-1}^H}{d_k^2 + (\vec{h}_k^H - (\mathbf{G}_{k-1}\vec{h}_k)^H\mathbf{H}_{k-1}^H)\vec{h}} \quad . \end{aligned}$$

Based on this, pseudo-code of the algorithm is given in Alg. 1, the GNU Octave [Eat07] implementation is also reproduced. The algorithm obviously requires m reciprocal calculations. The remaining arithmetic

Figure 3.21: GNU Octave implementation of modified Greville algorithm [KI08]

```
function G = mgreville(H,D)
  n = rows(H);  m = columns(H);
  G = zeros (m, n);

  a = D(1,1)^2 + norm(H(:,1))^2;
  G(1,:) = (1/a) * H(:,1)';
  for k = 2:m
    v = G(1:k-1,:)*H(:,k);
    G(k,:) = H(:,k)' - v'*H(:,1:k-1)';
    a = D(k,k)^2 + real(G(k,:)*H(:,k));
    G(k,:) = (1/a) * G(k,:);
    G(1:k-1,:) = G(1:k-1,:) - v*G(k,:);
  endfor
end
```

operations are counted in terms of real-valued MAC operations, to be consistent with execution units available on current DSP and FPGA architectures. Conjugation operations are not counted as they are folded into MAC operations with a change of sign (i.e. negative multiply-and-accumulate). For algorithm lines 1–2 4 n MAC operations are counted. For the loop body in lines 4–8 the number of MAC operations

Algorithm 1 Pseudo-code for modified Greville algorithm [KI08].

```
1: $a \leftarrow \mathbf{D}_{1,1}^2 + ||\mathbf{H}_{\{1..n\},\{1\}}||^2$
2: $\mathbf{G}_{\{1\},\{1..n\}} \leftarrow a^{-1} \cdot \mathbf{H}_{\{1..n\},\{1\}}^H$
3: for $k = 2$ to $m$ do
4:     $\vec{v} \leftarrow \mathbf{G}_{\{1..k-1\},\{1..n\}}\mathbf{H}_{\{1..n\},\{k\}}$
5:     $\mathbf{G}_{\{k\},\{1..n\}} \leftarrow \mathbf{H}_{\{1..n\}\{k\}}^H - \vec{v}^H\mathbf{H}_{\{1..n\},\{1..k-1\}}^H$
6:     $a \leftarrow D_{k,k}^2 + \mathrm{real}(\mathbf{G}_{\{k\},\{1..n\}}\mathbf{H}_{\{1..n\},\{k\}})$
7:     $\mathbf{G}_{\{k\},\{1..n\}} \leftarrow a^{-1}\mathbf{G}_{\{k\},\{1..n\}}$
8:     $\mathbf{G}_{\{1..k-1\},\{1..n\}} \leftarrow \mathbf{G}_{\{1..k-1\},\{1..n\}} - \vec{v}\mathbf{G}_{\{k\},\{1..n\}}$
9: end for
```

Operation	**Formula**	**Count**
saxpy with real-valued scalar	$\vec{z} \leftarrow \alpha\vec{x} + \vec{y} \quad (\alpha \in \mathbb{R})$	m
complex saxpy	$\vec{z} \leftarrow \alpha\vec{x} + \vec{y}$	$m^2 - m$
real-valued dot product	$\vec{z} \leftarrow \mathrm{real}(\vec{x}^H\vec{y} + \vec{v})$	m
complex dot product	$\vec{z} \leftarrow \vec{x}^H\vec{y} + \vec{v}$	$\frac{1}{2}m^2 - \frac{1}{2}m$
scalar reciprocal	$\beta \leftarrow \alpha^{-1} \quad (\alpha \in \mathbb{R})$	m

Table 3.3: Operation counts on a vector processor handling length-n operations.

is a function of the counter variable k:

$$12\ (k-1)\ n + 4\,n \quad .$$

With $\sum_{k=1}^{m} k = \frac{m^2}{2} + \frac{m}{2}$ the expression for the total number of MAC operations becomes:

$$6\,n\,m^2 - 2\,n\,m \quad .$$

The second good property of the algorithm are its fixed-length loops. Algorithm 1 utilizes only matrix-vector and vector-vector operations for which at least one dimension of the involved matrices resp. vectors is n. It is thus possible to express the algorithm in terms of length-n vector operations (see Alg. 2). Table 3.3 lists the number of length-n vector operations required to implement the algorithm.

The complexity is compared to the two best performing algorithms for calculation of the MMSE equalizer matrix documented in [KSI07]:

Algorithm 2 Reformulation of Alg. 1 that exposes fixed-length loops.

```
1:  $a \leftarrow \mathbf{D}^2_{1,1} + \mathbf{H}^H_{\{1..n\},\{1\}} \cdot \mathbf{H}_{\{1..n\},\{1\}}$
2:  $\mathbf{G}_{\{1\},\{1..n\}} \leftarrow a^{-1} \cdot \mathbf{H}^H_{\{1..n\},\{1\}}$
3:  for $k = 2$ to $m$ do
4:      for $j = 1$ to $k-1$ do
5:          $v_j \leftarrow \mathbf{G}_{\{j\},\{1..n\}} \cdot \mathbf{H}_{\{1..n\},\{k\}}$
6:      end for
7:      $\mathbf{G}_{\{k\},\{1..n\}} \leftarrow \mathbf{H}^H_{\{1..n\}\{k\}}$
8:      for $j = 1$ to $k-1$ do
9:          $\mathbf{G}_{\{k\},\{1..n\}} \leftarrow \mathbf{G}_{\{k\},\{1..n\}} - \overline{v_j} \cdot \mathbf{H}^H_{\{1..n\},\{j\}}$
10:     end for
11:     $a \leftarrow D^2_{k,k} + \text{real}(\mathbf{G}_{\{k\},\{1..n\}} \cdot \mathbf{H}_{\{1..n\},\{k\}})$
12:     $\mathbf{G}_{\{k\},\{1..n\}} \leftarrow a^{-1} \cdot \mathbf{G}_{\{k\},\{1..n\}}$
13:     for $j = 1$ to $k-1$ do
14:         $\mathbf{G}_{\{j\},\{1..n\}} \leftarrow \mathbf{G}_{\{j\},\{1..n\}} - v_j \cdot \mathbf{G}_{\{k\},\{1..n\}}$
15:     end for
16: end for
```

Table 3.4: Comparison of total number of MAC-operations required to obtain $\mathbf{G}_{MMSE}$ [K07].

Algorithm	**real-valued MAC operations**
Cholesky based	$4m^2n - 2mn + \frac{8m^3}{3} - \frac{2m}{3}$
QR-decomposition based	$6m^2n - 2mn + \frac{4m^3}{3} + m^2 - \frac{m}{3}$
modified Greville	$6m^2n - 2mn$

– Cholesky factorization followed by forward-backward substitution, i.e

$$\mathbf{L}\mathbf{L}^H = (\mathbf{H}^H\mathbf{H} + \mathbf{D}^2)$$
$$\mathbf{L}\mathbf{L}^H\mathbf{G} = \mathbf{H}^H \quad ,$$

– QR-decomposition of the extended matrix $\underline{\mathbf{H}}$, followed by backward-substitution

$$\underline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{bmatrix} \mathbf{R}$$
$$\mathbf{R}\mathbf{G} = \mathbf{Q}_1^H \quad ,$$

Comparison of the MAC operation counts is given in table 3.4. Assuming $m = n$, the comparison is illustrated in Fig. 3.22.

Application of the stream separation matrix needs $m \cdot n$ MACs. The last step is then LLR generation per stream using the Max-Log approximation. Assuming a separable modulation set, this can be implemented using independent (one-dimensional) table look-up operations (LU), 1 LU per LLR.

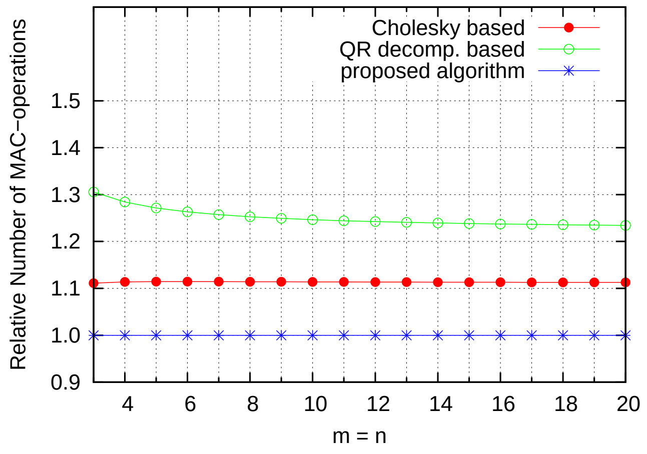


Figure 3.22: Relative number of operations required for computing $\mathbf{G}_{MMSE}$ for $\mathbf{H} \in \mathbb{C}^{n \times n}$ [KI08].

QRD-M tree search with a priori information and clipping Max-Log demapping

The M-demapper also evaluates Eq. (3.40) - not for all possible transmit vectors, but for a greedily chosen subset. To enable tree search detection (illustrated in Fig. 3.23), the channel matrix is QR decomposed. Because of the resulting triangular matrix, detection can be done sequentially for the transmit antennas (where one transmit antenna corresponds to one tree level) [RBO04]. The M-algorithm prunes the search tree at each level and follows only the M best nodes to the next level. By choosing M, the accuracy and complexity of the algorithm can be scaled. The squared distance of the mapped received vector $\mathbf{y}'$ from modulation symbol vector $\mathbf{x}(\mathbf{c})$ at the receiver is:

$$||\mathbf{y}' - \mathbf{Q}^H \mathbf{H} \mathbf{x}(\mathbf{c})||^2 \tag{3.45}$$

with

$$\mathbf{y}' = \mathbf{Q}^H \mathbf{y} \quad \text{and} \quad \mathbf{Q}^H \mathbf{H} = \mathbf{R}$$

The tree search metric is as in Eq. (3.40):

$$\mu(\mathbf{x}) = -\frac{1}{2\sigma^2}||\mathbf{y}' - \mathbf{R}\mathbf{x}(\mathbf{c}')||^2 + \frac{1}{2} L_A(\mathbf{c}')^T \mathbf{c}' \overset{!}{=} max \tag{3.46}$$

with the only difference that the dimensions of $\mathbf{y}'$ and $\mathbf{R}$ are equal to the actual tree search level and the corresponding number of leading bits from $\mathbf{c}$ is used as $\mathbf{c}'$. For MU-MIMO, different modulation levels for different users (i.e. transmit antennas) are possible on the same resources: the search tree becomes a mixed tree. In this case implementation uses different demapper functions (one function per tree level) – function calls are computationally cheap if applied for several subcarriers at once (cf. chapter 8).

To reduce complexity of the tree search for higher QAM modulation levels, set partitioning [dJW02,

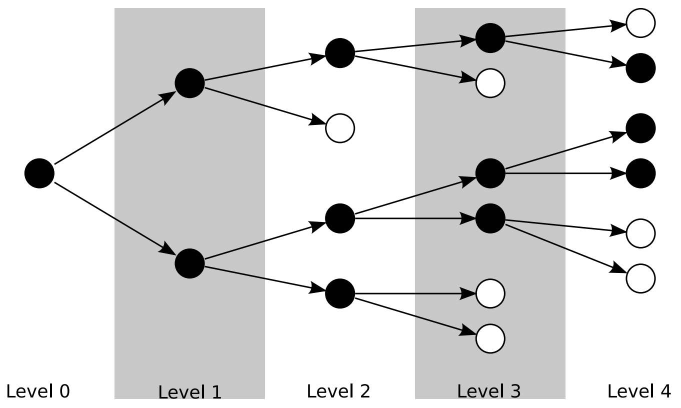


Figure 3.23: QRD-M search tree for BPSK, $M = 3$ and 4 transmit antennas [IKWB09].

Lai08, HKMS04] can be applied, which reduces the search tree to a 4-ary QPSK tree (with two/three times the depth when using 16QAM or 64QAM respectively) and leads to roughly constant computational effort per bit for all modulation levels. Set partitioning with the LTE modulation sets is illustrated in Fig. 3.24: the first two bits of a 16QAM symbol are the closest QPSK symbol, the first four bits of 64QAM are the closest 16QAM symbol. This modulation set is also separable.

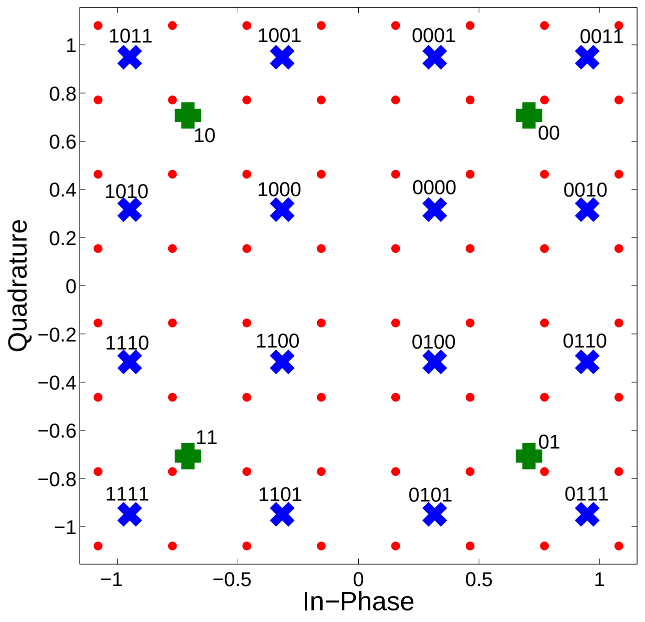


Figure 3.24: Set partitioning with separable modulation set [IKWB09].

In the clipping QRD-M demapper version, LLRs are generated from the candidate list of M *leaf nodes* obtained by the tree search. The subset of candidate leaf nodes with positive bit at position *i* is denoted $\mathcal{X}^{+1}_{i,\,\text{leaf}}$. The extrinsic LLR is:

$$L_E(c_i) = \begin{cases} \max\limits_{\mathbf{x}\in\mathcal{X}^{+1}_{i,\,\text{leaf}}} \mu(\mathbf{x}) - \max\limits_{\mathbf{x}\in\mathcal{X}^{-1}_{i,\,\text{leaf}}} \mu(\mathbf{x}) & \text{for } \mathcal{X}^{+1}_{i,\,\text{leaf}}, \mathcal{X}^{-1}_{i,\,\text{leaf}} \neq \emptyset \\ \mu_{\text{clip}}/\sigma^2 & \text{if } \mathcal{X}^{-1}_{i,\,\text{leaf}} = \emptyset \\ -\mu_{\text{clip}}/\sigma^2 & \text{else} \end{cases} \tag{3.47}$$

If the list contains only candidates with the same bit value for one LLR, the LLR is clipped to $\pm\mu_{\text{clip}}/\sigma^2$. An iteration of the tree search with actualized a priori information from the decoder only leads to a better result if the search gives a different candidate list. Algorithm accuracy in terms of BER for different values of parameter M in comparison to MMSE and ZF equalization are indicated in Fig. 3.25 for QPSK transmission over a 4x4 MIMO channel with uncorrelated Rayleigh fading. As stated before, BER is not a reliable measure for accuracy of soft-output demapping, as it evaluates only the signs of LLRs. MI with transmit bits of hard versus soft output may differ by several dB.

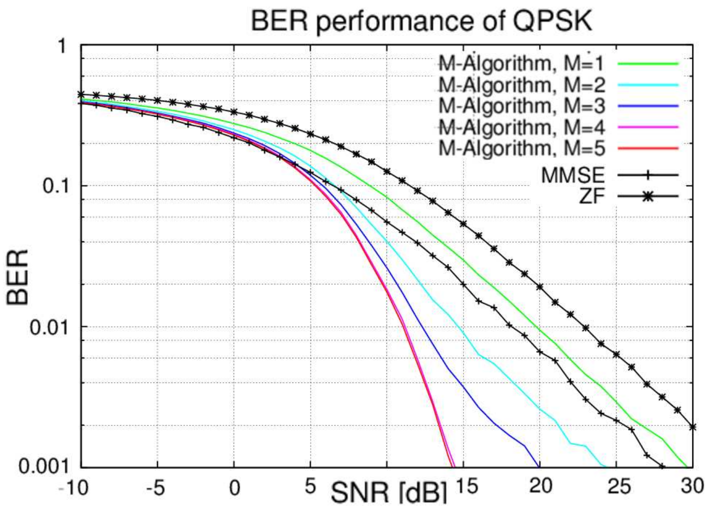


Figure 3.25: QRD-M MIMO demapping BER for 4x4 QPSK [WKI].

Hybrid unbiased MMSE / Subspace-Max-Log-APP

This demapper employs 'partial marginalization', described in [LJ08]. It scales in accuracy and complexity between unbiased MMSE with per-stream Max-Log demapping and joint Max-Log APP demapping.

Unbiased MMSE demapping gives reliability information about bits (in form of LLRs) at low complexity. Fig. 3.26 illustrates the uncoded error rates in 4x4 16QAM transmission (uncorr. Rayleigh fading) for the subset of the n LLRs with largest magnitude. For the SNR range of interest, the error rate e.g. of the 4 LLRs with largest magnitude (per MIMO vector) is far lower than that of maximum likelihood (ML) detection of all bits. The idea is to reduce the candidate search space for the Max-Log-APP solution to a fixed predefined size, based on reliability: the n bit positions with largest LLR magnitude are assumed as correct, and the remaining subspace of size $2^{N_T k-n}$ is 'most likely' to contain the ML solution and Max-Log-APP counter-hypotheses (N_T transmit antennas and k bit per transmit antenna, gives a sum of $N_T k$ LLRs to generate). For this remaining (smaller) 'unreliable' subspace, joint Max-Log-APP postprocessing can be performed.

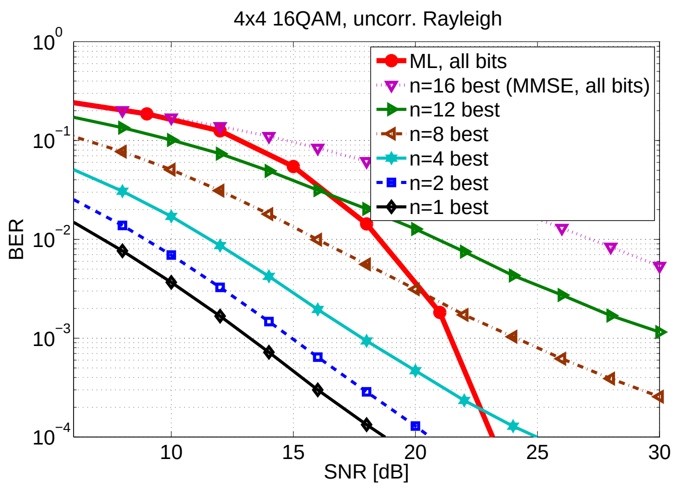


Figure 3.26: Partial BER of linear MIMO detection for n bit positions with largest LLR magnitude [IKB10].

The demapping algorithm consists of three steps. The first step is to perform unbiased MMSE demapping as described in Sec. 1.4.

In the second step the n LLRs with largest magnitude $|L(c_{i,j})|$ are selected as 'reliable', where n is a predefined constant. The bit values are the signs of the LLRs. The remaining $N_T k - n$ bit positions are the 'unreliable' subspace.

In the third and last step the algorithm computes the joint Max-Log-APP solution for the 'unreliable' candidate subspace. Since this subspace is relatively small, binary enumeration of the candidates can be performed. For each bit position, the subspace always contains hypothesis and counter-hypothesis vectors, so that clipping operations like in the list sphere decoder [HtB03] are not necessary. The extrinsic (Max-Log) LLR for the 'unreliable' bit positions are computed according to Eq. 3.39 on page 58 (using a priori LLRs from the decoder). The LLR vector which the detector outputs consists of n values generated by linear detection and $N_T k - n$ values generated by subspace-Max-Log-APP computation. For iterative detection-decoding only step 3 needs to be repeated after running the decoder. A pseudo-code formulation of the algorithm is given in Alg. 3.

Now it is illustrated that the accuracy in terms of error rates and mutual information varies between that of unbiased MMSE and Max-Log-APP. Both are included in the algorithm as special cases for $n = N_T k$ and $n = 0$ respectively. Simulation assumes 16QAM transmission over 4x4 uncorrelated

Rayleigh fading, and perfect channel estimation at the receiver. Uncoded error rates are shown for different n in Fig. 3.28a. The detector curve for $n = 0$ is denoted ML, since Max-Log-APP without apriori information and with hard output (bits instead of LLRs) reduces to searching the most likely candidate bit vector. Postprocessing for a small subspace of four bit positions (16 candidate vectors) already results in more than 1dB improvement for the practically interesting uncoded BER range around 10^{-1}. To assess the accuracy of soft-output demappers, mutual information (MI) is a more suitable measure than uncoded BER. MI in dependence on SNR is shown in Fig. 3.28b. By this measure, Max-Log processing only brings gains for SNR larger than 10dB in this scenario. For enhanced demapper accuracy also at low SNR, iterative demapping-decoding can be applied. Demapper accuracy for this case is illustrated in form of an EXIT chart in Fig. 3.28c. While the linear detector does not benefit from apriori information, the presented algorithm increasingly exploits it with decreasing parameter n.

Algorithm 3 Subspace-Max-Log by enumeration [IKB10]

```
$\hat{\mathbf{x}} = G_{unb}\mathbf{y}$ {MMSE stream separation}
**for all** tx-antennas **do**
    **for all** bit-positions **do** {bits of this antenna}
        $L_e(c_j) = (\min_{x_i \in \mathcal{X}^1_+} \Delta - \min_{x_i \in \mathcal{X}^1_-} \Delta)$ {max-log per stream}
    **end for**
**end for**
select $n$ positions of largest $|L(c_i)|$ {'reliable'}
**for all** $\mathbf{c}^{(sub)} \in 2^{N_T k - n}$ **do** {'unreliable' subspace}
    $metric(\mathbf{x}(\mathbf{c})) = \frac{-1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \frac{1}{2}L_a(\mathbf{c})^T\mathbf{c}$
**end for**
**for** $i = 1$ to $N_T k - n$ **do** {'unreliable' bit positions}
    $max\_p = \max_{\mathbf{c}^{(sub)}|c_j=+1} (metric(\mathbf{x}(\mathbf{c})))$
    $max\_m = \max_{\mathbf{c}^{(sub)}|c_j=-1} (metric(\mathbf{x}(\mathbf{c})))$
    $L_e(c_j) = max\_p - max\_m - L_a(c_j)$
**end for**
```

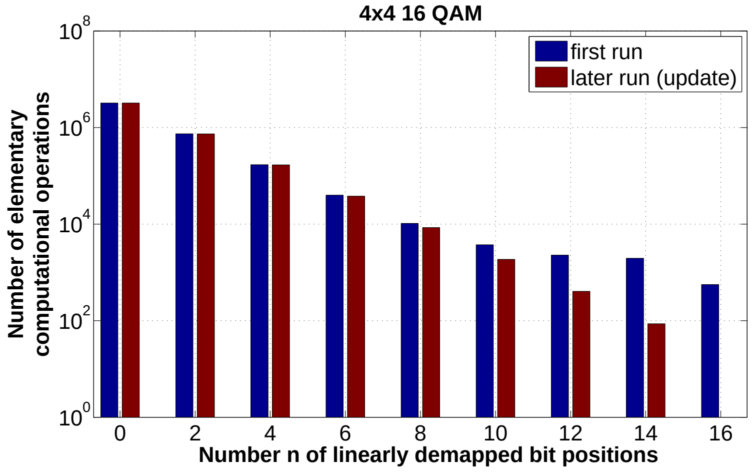
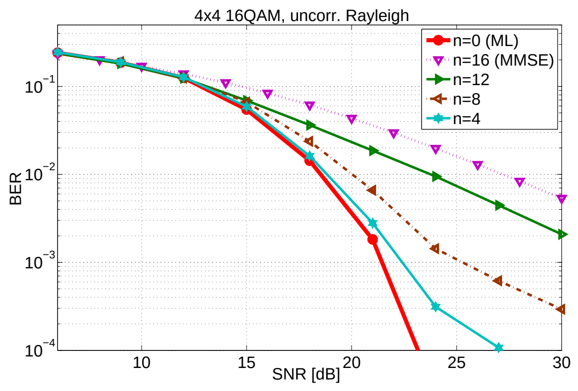
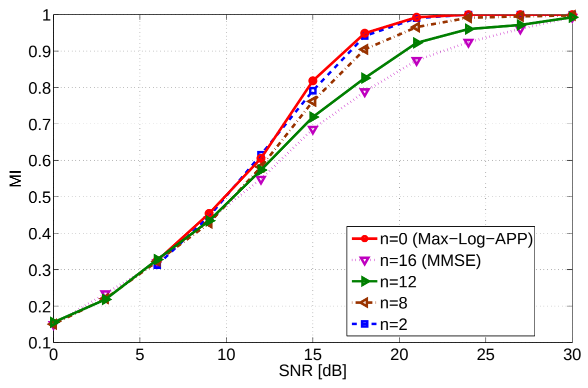


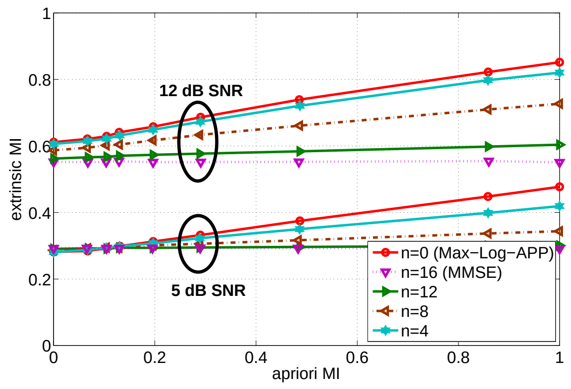
Figure 3.27: Complexity scales exponentially with parameter n [IKB10].



(a) Error rate of proposed algorithm for different number of linearly detected bit positions.



(b) Mutual information for different number of linearly detected bit positions.



(c) EXIT chart: usage of apriori knowledge from the channel decoder.

Figure 3.28: Hybrid uMMSE / subspace Max-Log-APP demapper [IKB10].

Algorithm complexity: the number of elementary real-valued computational operations for different n is illustrated in Fig. 3.27. Operations like *Multiply-Accumulate* and *Compare-Select* are counted as the same unit. For hardware independence the possibility of reuse of intermediate results ('infinite' memory) and cost-free *Load/Store* operations are assumed. The figure shows that complexity scales exponentially with n (apart from MMSE preprocessing), which is due to the problem being NP hard.

3.2.4 Accuracy Increase with A Priori Information

If instead of MIMO transmission a SIMO system is used (one transmit antenna and several receive antennas), the optimal receiver exploits the receive diversity by using maximum ratio combining (MRC): the symbols received on each antenna are combined by weighting them according to signal amplitude [GLMZ07]. Here it is shown that with growing apriori information the accuracy of both the MIMO APP detector and its max-log-APP approximation increase up to that of SIMO MRC, when transmitting with the same energy per symbol [IKB09]. And even better: SIMO MRC accuracy for a (possibly shifted) BPSK modulation. This quantifies the maximum benefit of a priori information for MIMO detection and can serve as an upper bound for demapper accuracy in iterative MIMO demapping-decoding. This is not a statement about the accuracy of iterative demapping-decoding, this statement is about demapper accuracy in dependence on the amount of available a priori information.

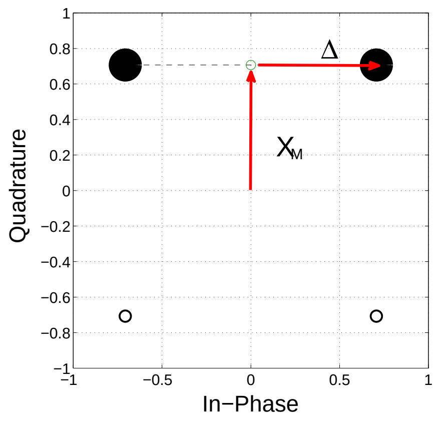


Figure 3.29: If all but one bits in a QAM symbol vector are fixed (or known at the receiver), this corresponds to a shifted and scaled BPSK modulation [IKB09].

APP demapping with large a priori information The APP extrinsic LLR is (subsection 3.2.3):

$$L_e(c_i) = ln \frac{\sum_{\mathbf{x}\in\mathcal{X}_i^{+1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n\neq i} \mathbb{P}(c_n)}{\sum_{\mathbf{x}\in\mathcal{X}_i^{-1}} \mathbb{P}(\mathbf{y}|\mathbf{x}) \prod_{n\neq i} \mathbb{P}(c_n)} \tag{3.48}$$

For no apriori information about a bit n, $\mathbb{P}(c_n)$ is:

$$\mathbb{P}(c_n = +1) = \mathbb{P}(c_n = -1) = 1/2, \tag{3.49}$$

while for the limit of full apriori knowledge it is either

$$\mathbb{P}(c_n = +1) = 1 \quad \text{and} \quad \mathbb{P}(c_n = -1) = 0 \tag{3.50}$$

or vice versa.

In this limit case, the symbol vector candidates are ’filtered’ by the $\mathbb{P}(c_n) = 0$, so that in the nominator and denominator of eq. (3.48) only the candidate vector with

$$\prod_{n\neq i} \mathbb{P}(c_n) = \prod_{n\neq i} 1 = 1$$

remains. The extrinsic LLR then yields

$$L_e(c_i) = -\frac{1}{2\sigma^2}\left(||\mathbf{y} - \mathbf{H}\mathbf{x}_+||^2 - ||\mathbf{y} - \mathbf{H}\mathbf{x}_-||^2\right) \tag{3.51}$$

where all bits except c_i are chosen according to the a priori information. Assume that the bit c_i is transmitted on antenna k. The vector hypotheses $\mathbf{x}_+$ and $\mathbf{x}_-$ then differ in exactly the k-th component. With the vector $\mathbf{x}_I$ of symbols which both hypotheses have in common

$$\begin{aligned}\mathbf{x}_I &= (x_{+_1}, \dots, x_{+_{k-1}}, 0, x_{+_{k+1}}, x_{+_{n_{Tx}}})^T \\ &= (x_{-_1}, \dots, x_{-_{k-1}}, 0, x_{-_{k+1}}, x_{-_{n_{Tx}}})^T\end{aligned}$$

it is substituted

$$\mathbf{y}_{SIMO} = \mathbf{y} - \mathbf{H}\mathbf{x}_I \tag{3.52}$$

and obtained:

$$L_e(c_i) = -\frac{1}{2\sigma^2}\Big(||\mathbf{y}_{SIMO} - \mathbf{h}_k x_{+_k}||^2 - ||\mathbf{y}_{SIMO} - \mathbf{h}_k x_{-_k}||^2\Big)$$

The symbols transmitted on all other antennas were given by a priori information, so that this known interference cancelled out and the MIMO problem has become a SIMO problem. The transmit symbol hypotheses (complex numbers) x_{+_k} and x_{-_k} for antenna k differ in only one bit, i.e. one direction in the complex plane. This modulation can be regarded as a shifted BPSK:

$$x_{+_k} = x_M + \Delta, \qquad x_{-_k} = x_M - \Delta \tag{3.53}$$

where x_M is the middle point between the two symbol hypotheses and $2 \cdot |\Delta|$ is their distance. This shifted BPSK modulation is illustrated in Fig. 3.29. It is substituted

$$\mathbf{y}_{SIMO,BPSK} = \mathbf{y}_{SIMO} - \mathbf{h}_k x_M \tag{3.54}$$

and the LLR becomes [IKB09]:

$$\begin{aligned}L_e(c_i) &= -\frac{1}{2\sigma^2}\Big(||\mathbf{y}_{SIMO,BPSK} + \mathbf{h}_k\Delta||^2 \\ &\quad -||\mathbf{y}_{SIMO,BPSK} - \mathbf{h}_k\Delta||^2\Big) \\ &= -\frac{4}{2\sigma^2}\Re\{\mathbf{y}^H_{SIMO,BPSK}\mathbf{h}_k\Delta\}\end{aligned} \tag{3.55}$$

Max-Log-APP demapping with large a priori information In this paragraph it is shown that for large a priori information, only one bit of one transmit antenna remains unknown – the rest is cancelled by the Max-Log-APP detector [IKB09]. It is assumed that the $L_a(c_n)$ are correct and have a 'large enough' magnitude, so that they and not the Euclidean metric term decide about the candidate vector selection. This is certainly fulfilled if:

$$|L_a(c_n)| > \max_{\mathbf{x}_1 \in \mathcal{X}_n^+, \mathbf{x}_2 \in \mathcal{X}_n^-} \left| \frac{1}{2\sigma^2} \, ||\mathbf{y} - \mathbf{H}\mathbf{x}_1||^2 - ||\mathbf{y} - \mathbf{H}\mathbf{x}_2||^2 \, \right|$$

i.e. the absolue value of $L_a(c_n)$ is larger than the maximum possible contribution of bit n to the Euclidean distance part of the metric, so that the Max-Log bit hypothesis is chosen according to the a priori LLR. For the computation of one LLR, all other bits in both Max-Log transmit vector hypotheses $\mathbf{x}_+$ and $\mathbf{x}_-$ are then chosen according to the a priori LLRs, so that the a priori LLRs cancel out [IKB09]:

$$L_e(c_i) = -\frac{1}{2\sigma^2}\left(||\mathbf{y} - \mathbf{H}\mathbf{x}_+||^2 - ||\mathbf{y} - \mathbf{H}\mathbf{x}_-||^2\right) \tag{3.56}$$

This is the same equation as for APP detection with full a priori information (eq. (3.51)).

Maximum Ratio Combining Now it is assumed that the transmitter has one antenna, and that the receiver uses its multiple antennas for maximum ratio combining (which is the SNR-optimal diversity combining method [GLMZ07]). The channel vector is denoted $\mathbf{h}$ and the system model becomes:

$$\mathbf{y}_{MRC} = \mathbf{h}x + \mathbf{n} \tag{3.57}$$

The weighting of the symbols received on the different antennas is done according to signal amplitude: the weight vector for the received symbols is $\mathbf{w} = \mathbf{h}^H$. The equalized symbol after combining is [GLMZ07]:

$$\hat{x} = \frac{\mathbf{h}^H \mathbf{y}_{MRC}}{\mathbf{h}^H \mathbf{h}} = x + \frac{\mathbf{h}^H \mathbf{n}}{\mathbf{h}^H \mathbf{h}} \tag{3.58}$$

The resulting noise power is:

$$\begin{aligned} \sigma^2_{MRC} &= \mathbb{E}\{|\frac{\mathbf{h}^H \mathbf{n}}{\mathbf{h}^H \mathbf{h}}|^2\} = \frac{\mathbb{E}\{|\mathbf{h}^H \mathbf{n}\mathbf{n}^H \mathbf{h}|\}}{||\mathbf{h}||^4} \\ &= \frac{\mathbf{h}^H \mathbb{E}\{\mathbf{n}\mathbf{n}^H\}\mathbf{h}}{||\mathbf{h}||^4} = \frac{\sigma^2}{||\mathbf{h}||^2} \end{aligned}$$

As modulation set a (possibly shifted) BPSK modulation with the two possible symbol values x_1 and x_0 is assumed. Soft demodulation yields the LLR:

$$L = -\frac{1}{2\sigma^2_{MRC}} ln \frac{\mathbb{P}(\hat{x}|x_1)}{\mathbb{P}(\hat{x}|x_0)} \tag{3.59}$$

The two symbol candidates can again be written as

$$x_1 = x_M + \Delta, \qquad x_0 = x_M - \Delta \tag{3.60}$$

where for normal BPSK x_M would be zero (Fig. 3.29).

The LLR becomes

$$\begin{aligned} L &= -\frac{||\mathbf{h}||^2}{2\sigma^2}\left\{|\hat{x} - (x_M + \Delta)|^2 - |\hat{x} - (\, x_M - \Delta)|^2\right\} \\ &= -\frac{||\mathbf{h}||^2}{2\sigma^2}\, 4\Re\left\{\Delta\overline{(x_M - \hat{x})}\right\} \end{aligned}$$

To subtract the modulation set center it is substituted

$$\hat{x}_{BPSK} = \hat{x} - x_M \tag{3.61}$$

and obtained

$$L = \frac{||\mathbf{h}||^2}{2\sigma^2}\, 4\Re\left\{\Delta\overline{\hat{x}_{BPSK}}\right\} \tag{3.62}$$

Equivalence The MRC channel vector $\mathbf{h}$ can be seen as one column of the MIMO channel matrix, e.g. the k-th: $\mathbf{h} = \mathbf{h}_k$. It is noted that with same energy per symbol:

$$|\mathbf{y}^H_{SIMO,BPSK}\mathbf{h}| = ||\mathbf{h}||^2|\hat{x}_{BPSK}|$$

This means that with the same average transmit power per data stream, the MRC solution for (shifted) BPSK transmission in a SIMO system is equivalent to the Max-Log-APP solution and the APP solution for MIMO transmission (provided sufficient a priori information). For each LLR to compute in the MIMO system, a maximum ratio combiner remains after cancellation of the known signals.

Simulative illustration For simulation, uncorr. Rayleigh fading is assumed. It is looked at a 4x4 and 1x4 system, for different amounts of a priori information and channel SNR.

EXIT-Chart based To measure demapper accuracy in dependence on a priori information, the a priori LLRs generated in the common way [ten01]: the transmit bits c (± 1 values) are perturbed by additive White Gaussian Noise and scaled according to noise variance, yielding the L_a. The chosen noise variance determines the mutual information $I(L_a; c)$ of the apriori LLRs L_a with the transmit bits c. MI is measured using Eq. (3.33) on page 54. Extrinsic information transfer (EXIT) charts for 4x4 QPSK transmission and Max-Log-APP demapping are shown for different E_S/N_0 in Fig. 3.30. With increasing $I(L_a;c)$, demapping accuracy measured as $I(L_e;c)$ increases up to the value for SIMO MRC with shifted BPSK modulation (denoted SBPSK, as illustrated in Fig. 3.29).

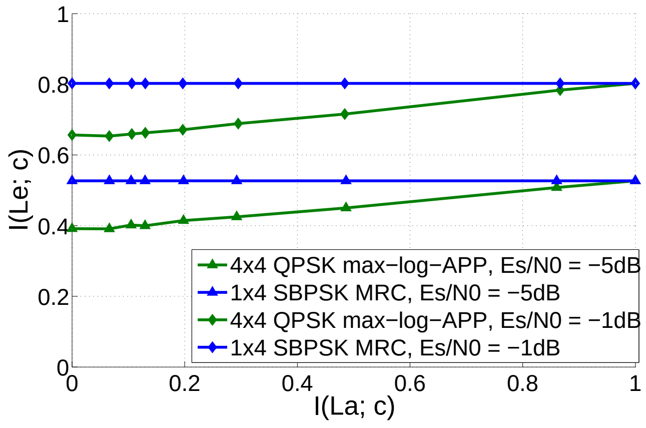


Figure 3.30: With an increasing amount of a priori information, max-log-APP MIMO detector accuracy for QPSK increases up to SIMO MRC accuracy for (shifted and scaled) BPSK modulation [IKB09].

Varying $\mathbf{E_S/N_0}$ To illustrate the maximum possible demapper accuracy gain through a priori information in this scenario, Fig. 3.31 shows the accuracy of 4x4 QPSK max-log-APP without a priori information and 1x4 SBPSK MRC detection in dependence on E_S/N_0. The distance between the two curves is roughly 2dB.

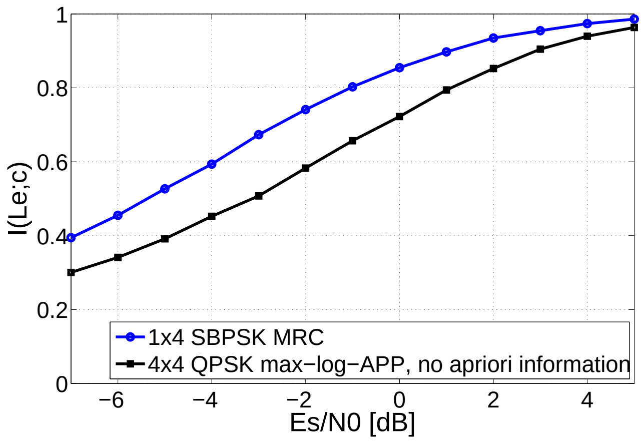


Figure 3.31: The difference between MIMO max-log-APP accuracy without apriori information and max-log-APP accuracy for large apriori information is roughly 2dB for 4x4 QPSK transmission. For channel E_S/N_0 of -1dB and -5dB the transition from the lower curve to the upper curve is shown in Fig. 3.30 [IKB09].

Some Comments The maximum accuracy of a demapper in iterative MIMO demapping-decoding was quantified to be that of SIMO BPSK transmission with maximum ratio combining detection. This equivalence holds if the average transmit power per antenna is equal. It does not hold if the transmitter's sum power is constrained, since then the power per data stream would be reduced to $1/n$ when using n transmit antennas. If demapper and decoder would be considered jointly as one large processing block implementing an ML receiver on message basis, there would be no such thing as extrinsic information. The notion of extrinsic information only comes into play when performing iterative local processing in different blocks. If full a priori information would be available for the demapper, there would be no need to run it, as the message would already be known. The motivation is to have a converging iterative process. The decoder exploits diversity which is provided by the channel and contained in the LLRs. The potential gain in demapper accuracy using apriori information (difference between MIMO max-log-APP and SIMO BPSK MRC) increases with higher number of antennas and larger modulation size. From a demapper point of view, n-PSK modulation could be a good idea. When a flipped bit in a symbol means the opposite position on the unit circle, the Max-Log-APP demapper reaches SIMO BPSK accuracy for large a priori information: the two symbol hypotheses for one bit would have maximum distance in the complex plane, and no energy would be wasted by a shifted modulation set center.

3.3 Constituent Decoder

Correct computation of bit APPs for a convolutional code is done by the BCJR algorithm [BCJR74]. Suboptimal approximations include the already mentioned Max-Log approximation (Max-Log-BCJR), soft-ouput M and T algorithm [LC04]. Operation counts per LLR of Max-Log BCJR decoding and Turbo (de-)interleaving are shown in Tab. 3.5, assuming LTE parameters (8-state decoder and quadratic permutation polynomial interleaving). Elementary operations are *Add-Compare-Select*, *Add*, *Compare-Select* and *Load/Store*.

Convergence behaviour of Turbo decoding is normally illustrated using EXIT charts [ten01]. An EXIT chart for LTE parameters [3GP09] is shown in Fig. 3.32.

There is one difference in iterative demapping decoding compared to Turbo decoding: while the constituent decoder in a Turbo decoder only has to reconstruct LLRs for information bits, this now has to be done also for parity bits – a demapper iteration uses both information and parity bit a priori LLRs. This difference is due to the serial concatenation of MIMO modulation mapper and channel encoder in difference to the parallel concatenation of the constitutional encoders (compare chapter 2).

With growing a priori information the decoder’s accuracy increases up to that when only one bit of the codeword is uncertain.

Function	Operations
8-state BCJR decoder	
Compute Metrics (fw+bw)	16 ACS/uncLLR
Paths	16 Add/uncLLR
LLR generation	14 CS/uncLLR, 1 Sub/uncLLR
QPP interleaver	
get target index	1 Read/LLR
get source llr	1 Read/LLR
write target llr	1 Write/LLR

Table 3.5: Turbo decoder complexity for LTE parameters using Max-Log-BCJR.

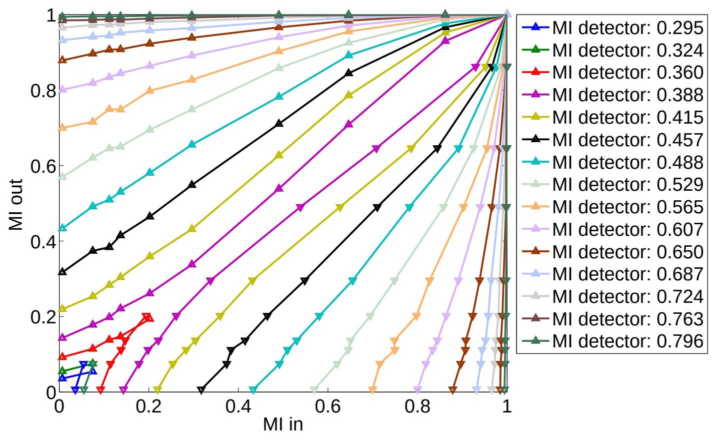


Figure 3.32: EXIT chart for turbo decoder with LTE parameters.

Chapter 4

Receiver Description Language

To describe node update schedules, where different algorithms for each update are possible, a description language is introduced in this chapter. The chapter starts with the notation of the receiver architecture (Sec. 4.1) and of the component algorithms (Sec. 4.2). For serial processing on one processing element, the language has a regular grammar (Sec. 4.3) and can thus be parsed by a finite state automaton. A receiver then corresponds to a path through this finite state automaton. The description of parallel schedules (parallelization using several processing elements) is described in Sec. 4.4. While the purpose of this chapter is instantiation and connection of receiver APP components, an automatic mapping from such a receiver description to prediction of the performance parameters is described in chapter 5.

4.1 Receiver Architecture Notation: Directed Bipartite Graph

A factor graph F is given by the sets of its vertices (nodes) and directed edges $F = \{V, E\}$, with the property that the graph is bipartite: the set of nodes consists of two disjoint subsets, where every edge is between nodes belonging to different sets. For the factor graph describing the generic receiver architecture (compare Fig. 2.1 on page 26), the first node subset are the factor nodes:

$$V_1 = \{\texttt{ce}, \texttt{dem}, \texttt{dec1}, \texttt{dec2}, \texttt{map}\} \tag{4.1}$$

The node abbreviations used here are listed in table 4.1 together with the corresponding factor node and the factor node type. The second subset are the variable nodes:

$$V_2 = \{u, c_1, c_2, y, H, x\}. \tag{4.2}$$

Factor node	Type	Abbreviation
Channel estimation	`channel estimation`	`ce`
MIMO demapper	`demapping`	`dem`
Constituent decoder 1	`decoding`	`dec1`
Constituent decoder 2	`decoding`	`dec2`
Soft Mapper	`mapping`	`map`

Table 4.1: Naming factor nodes .

The complete set of nodes is:

$$V = V_1 \cup V_2 \tag{4.3}$$

The general set of edges $E \subseteq V \times V$ with the bipartite graph property is

$$E = E_1 \cup E_2 \text{ ; with } E_1 \subseteq V_1 \times V_2 \text{ , } E_2 \subseteq V_2 \times V_1 \tag{4.4}$$

where the adjacency matrix can be described as:

$$M = \begin{pmatrix} 0 & M_1 \\ M_2 & 0 \end{pmatrix} \tag{4.5}$$

In the receiver graph case from Fig. 2.1 on page 26 the edges are:

$$\begin{aligned} E \;=\; & \{(u, dem), (u, map), (u, dec1), (u, dec2), (dem, u), (dec1, u), (dec2, u), (c1, dem), \\ & (c1, map), (c1, dec1)(dem, c1), (dec1, c1), (c2, dem), (c2, map), (c2, dec2), \\ & (dem, c2), (dec2, c2), (y, ce), (y, dem), (H, dem), (ce, H), (x, ce), (map, x)\} \end{aligned} \tag{4.6}$$

4.2 Component Algorithm Notation

After naming the factor nodes, now the mapping of an algorithm to a node is described. The exemplarily considered algorithms are listed in table 4.2, together with algorithm type and abbreviation. The set of algorithm abbreviations is

$$A = \{\texttt{wif}, \texttt{snd}, \texttt{ummse}, \texttt{hummse-ml(m=}m\texttt{)}, \texttt{maxlog}, \texttt{bcjr}\} \tag{4.7}$$

Algorithm	Type	Section	Abbreviation
Wiener interpolation filter	`channel estimation`	3.1.2	`wif`
2nd order model CE	`channel estimation`	3.1.3	`snd`
unbiased MMSE	`demapping`	3.2.3	`ummse`
Max-Log-APP	`demapping`	3.2.3	`maxlog`
hybrid uMMSE/Max-Log-APP, m LLRs linear	`demapping`	3.2.3	`hummse-ml(m=`m`)`
BCJR	`decoding`	3.3	`bcjr`

Table 4.2: Naming component algorithms.

To map an algorithm to a factor node, the abbreviations of node and algorithm are concatenated. Algorithm and factor node must have the same type (e.g. the BCJR algorithm is not applicable for channel estimation).

The set of valid algorithm mappings to factor nodes is the alphabet Σ (set of symbols) of the receiver description language:

$$\Sigma = \Big\{(v\_a) \mid v \in V_1,\ a \in A,\ \text{factor node } v \text{ and algorithm } a \text{ have same type}\Big\} \tag{4.8}$$

Examples:

- `ce_wif`: channel estimation using Wiener interpolation filter.
- `dem_hummse-ml(m=8)`: MIMO demapping using the hybrid unbiased MMSE / subspace Max-Log-APP algorithm with parameter $m = 8$.
- `dec2_bcjr`: constituent decoder 2 implementing the BCJR algorithm.

4.3 Serial Computation Schedule Notation: Regular Expression

A schedule is a valid word from the regular receiver description language $\mathcal{L}$. The language can be defined by a starting set of valid words and 'construction rules' [HU79].

Starting set:

- $\mathcal{L}(\emptyset)$: 'empty' receiver is in $\mathcal{L}$.
- $\underset{s\in\Sigma}{\forall} \mathcal{L}(s) = s$: only one factor node update

Construction rules:

- $\underset{s,t\in\Sigma}{\forall} \mathcal{L}(s|t) = \mathcal{L}(s) \cup \mathcal{L}(t)$: alternative
- $\mathcal{L}(st) = \{\alpha\beta \mid \alpha \in \mathcal{L}(s) \wedge \beta \in \mathcal{L}(t)\}$: sequence

- $\mathcal{L}(b^*) = \bigcup\limits_{i\geq 0}\mathcal{L}^i(b)$: repetition
- $\mathcal{L}(b^+) = \bigcup\limits_{i\geq 1}\mathcal{L}^i(b)$: nonzero repetition (at least once)

A receiver design space (search space) is a subset $\mathcal{R} \subseteq \mathcal{L}$ and can be given as regular expression.

Examples:

- $\mathcal{R}_{\text{ex-1}}$ = (`ce_wif dem_ummse`)(`dec1_bcjr dec2_bcjr`)$^+$ describes a 'normal' linear receiver with turbo decoder (at least one turbo decoder iteration).
- $\mathcal{R}_{\text{ex-2}}$ = (`ce_wif`)(`dem_ummse`|`dem_maxlog`|`dec1_bcjr`|`dec2_bcjr`)* describes a receiver with possibly iterative demapping – decoding allowing free concatenation of four demapping/decoding components.

4.4 Parallel Schedule Notation: Parallelization on Factor Node Level

In order to balance computation effort and communication effort, parallelization using several PEs is done on a factor node level. There are strong data dependencies inside one factor, while updates of different factors can be independently computed in parallel with only input/output values as possible data dependencies. So one factor update is computed SIMD parallel on one PE, while different factors are possibly updated in parallel on different PEs.

The demapper consists of different factors for each time instance and subcarrier (compare Fig. 2.1 on page 26), so that the factor updates of a demapper update can be perfectly parallelized with several PEs.

Parallel schedules can therefore be characterized by the mappings of factor updates (with the respective algorithms) to PEs. This can be seen as a list of decisions at a certain times – about which factors to update with which algorithms on how many PEs. A new decision is always done when a factor update is completed. Such a decision list does not follow a regular grammar (like the special case of serial schedules, i.e. for 1 PE does), it needs variables to parse the language.

A synchronous data flow (SDF) graph as used in [KWA$^+$09, LM87] can be generated from the decision list. Fig. 4.3 shows the SDF corresponding to the decision list according to Fig. 4.1. The SDF carries less information since it does not contain the PE mappings.

There is an influence of parallel schedules on receiver accuracy prediction as described in chapter 5: factor update start times and durations have to be considered. A factor collects its input at start time, message updates are output after the update duration. A receiver accuracy prediction (chapter 5) consists of time-sequential look-ups with start and end events (generated from the decision list describing the

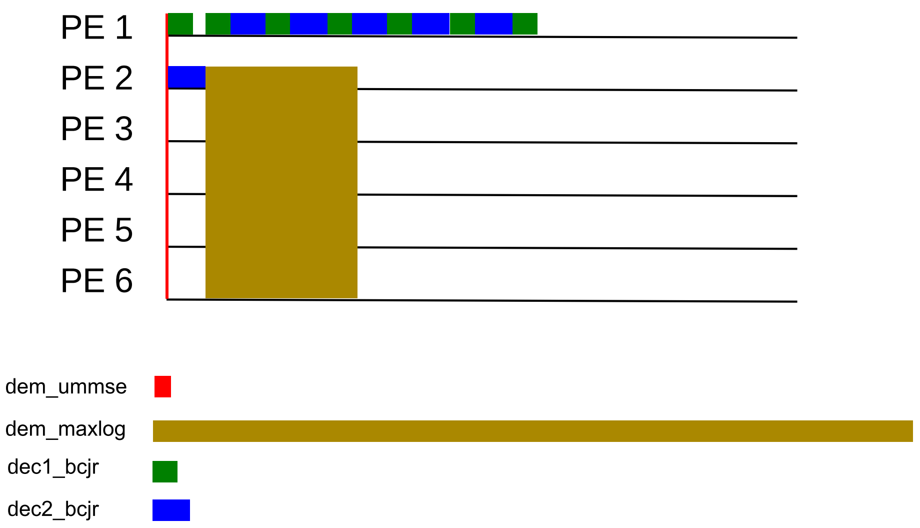


Figure 4.1: Illustration of parallel schedule from Fig. 6.1.

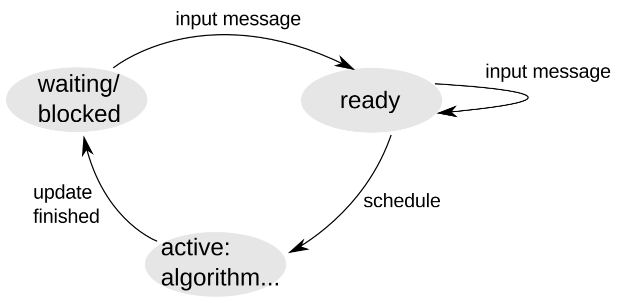


Figure 4.2: Factor states.

schedule).

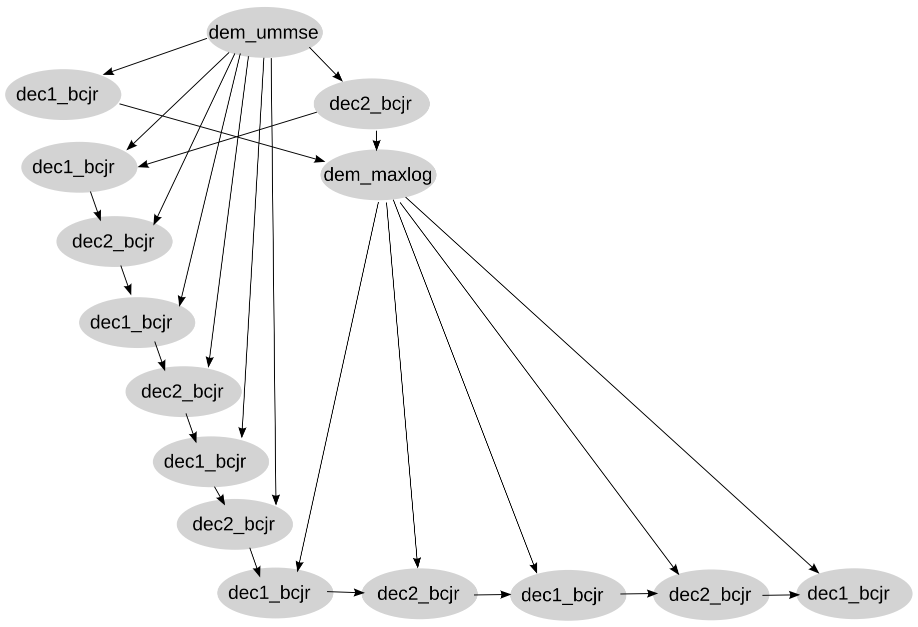


Figure 4.3: Synchronous data flow graph of the parallel schedule example from Fig. 6.1,4.1.

Chapter 5

Receiver Performance Prediction

This chapter deals with the determination of performance parameters for a concrete receiver given a certain transmission mode and channel distribution, where the receiver may be given in a specification according to chapter 4. The relevant hardware parameters are the number of processing elements and the instruction set architecture (ISA). The performance parameters derived in this chapter are the receiver's complexity (Sec. 5.1), delay (Sec. 5.2), throughput (Sec. 5.3) and accuracy (BER in dependence on SNR, Sec. 5.4). Complexity considers the time complexity of receiver algorithms with the specific target ISA. Data space complexity or program length complexity are not considered, i.e. memory is assumed to be large enough for the considered algorithms. Data move operations are often free if the target has independent load/store and arithmetic pipelines and DMA controllers. The fast performance prediction described in this chapter is the basis for automatic receiver optimization in chapter 6.

5.1 Complexity

Algorithm complexity can be considered either hardware independent or hardware dependent. A hardware independent measure is the amount and type of necessary elementary operations. A hardware dependent measure also compares the effort for different types of elementary operations. For hardware dependent complexity, either an implementation can be benchmarked (achieved complexity of implementation), or a theoretical upper limit (assuming optimal implementation, optimal processor utilization) can be used. Here, hardware-independent counts of elementary operations (theoretical upper limit) are used as a first step, which are then mapped to 'complexity' using an example processor core (with specific SIMD width using certain number formats). This processor core is also used in the testbed described in chapter 8, which allows for comparison of optimal and achieved processor utilization. The hardware-dependent measure is ’cycles/bit’, which is independent of clock speed and packet length. ’Retargeting’ of the complexity measure is of course possible for different instruction sets and different processor cores.

Hardware-independent algorithm complexity measure The counted elementary operations are *Multiply-Accumulate* (MAC), *table Look-Up* (LU), *Add-Compare-Select* (ACS), *Compare-Select* (CS) and *Read-/Write* (RW). MACs are used in linear algebra operations for signal processing, especially in channel estimation and MIMO demapping. LUs are used in the demapper’s LLR generation. Decoding uses ACS for trellis traversal and CS for LLR reconstruction in Max-Log BCJR computation. RW operations are used especially in interleavers.

Hardware-dependent measure, theoretical clock cycles on Cell SPU To enable comparison and joint optimization, the different operations have to be expressed in a common metric. For this hardware dependent mapping, theoretically achievable clock cycles on the target hardware under the assumption of full utilization of processor resources are counted. By using this theoretical upper limit, the measure is hardware-dependent, but independent from the actual implementation code quality (different from using benchmark results as complexity measure). Usage of different number formats for different components is of course possible: fixed point or floating point numbers, with certain number of bits. Here, the Cell processor SPU is chosen as example target hardware due to its general-purpose signal processing architecture and high performance. It is used in the SDR testbed described in chapter 8 which gives implementation benchmarks and also discusses how close an actual implementation with reasonable programming effort (C-language using vector intrinsics) can reach the theoretical cycle count. The numbers of elementary operations per cycle on the SPU using SIMD parallel implementation are given in table 5.1. The numbers in the table are a consequence of SIMD processing with 128 bit width, where signal processing operations (like MAC) are performed on 32bit (single-precision) float numbers and decoding uses 16bit integer representation for LLRs. Load/store operations can be done in parallel to arithmetic operations (different processor pipelines) for all blocks except the turbo (de-)interleaver, where they have to be counted explicitly. Retargeting the complexity measure for a different architecture would use a different table.

Complexity of a (parallel) factor update schedule A schedule consists of a number of factor updates with certain algorithms. The schedule may include only sequential updates, or also updates of different factors in parallel on different processor elements (PEs). A schedule with N_a factor updates has the

Operation	SPU Cycles
real-valued Multiply-Accumulate (MAC, 32bit float)	0.25
Select (conditional move, 32bit float)	0.25
table look-up (LU, demapper)	1
Add-Compare-Select (decoder, 16bit)	0.5
Compare-Select (decoder LLR gen., 16bit)	1
Add (decoder, 16bit)	0.125
QPP read (turbo interleaver)	1
QPP write (turbo interleaver)	1

Table 5.1: Mapping hardware-independent algorithm complexity to joint complexity metric for example target hardware.

complexity

$$C = \sum_{i=1}^{N_a} c(a_i) \tag{5.1}$$

where $c(a_i)$ is the number of cycles per LLR of the factor update number i (with algorithm a), i.e. the complexity of the schedule is the sum of the complexities of the contained factor updates. Processor idle times (at the beginning or end of the schedule or inbetween) to not contribute to the complexity measure.

5.2 Processing Delay

Delay measures the processing time of a packet, which is e.g. relevant to avoid expiring of protocol timers (like HARQ ACK/NACK). As for complexity, the unit is [cycles/LLR], to be independent of clock speed and packet length. If one processor element works exclusively on a packet (serial processing schedule), the processing delay is equal to the complexity. If more than one PE is available, the PEs can work concurrently on different packets – with the same complexity (per packet). In order to reduce delay (possibly at the expense of increased complexity), several processor elements can work on the same packet in parallel (parallel processing schedule). The delay is

$$d = \max_i \{t_{end}(a_i)\} - \min_i \{t_{start}(a_i)\} \tag{5.2}$$

with start times t_{start} and end times t_{end} of the factor updates.

5.3 Throughput

The throughput of a receiver can be determined from the computation schedule, the available number of PEs and the clock speed. For repeated execution of a parallel schedule for different packets, pipelining at schedule level may be possible: depending on the schedule, idle times of some PEs at the beginning and end of the schedule can be used for execution of the neighbouring schedule instance. The pipeline speedup is denoted S_{pipe}. Pipelining may especially be possible if the number of available PEs is larger than the number used by the schedule. If the number of available PEs is an integer multiple of the number of PEs used by the schedule, then the schedule can even be run with several instances in parallel (on different packets). This includes concurrent application of a serial schedule to different packets. Although this would rather be parallelization of schedules than pipelining, the speedup is nevertheless counted in S_{pipe}. The throughput is

$$TP = S_{pipe} \cdot f_{clock}/d \tag{5.3}$$

with clock speed f_{clock} and delay d. The unit of throughput is [bit/s].

5.4 Accuracy: BER and MI in dependence on SNR

The usage of iterative processing naturally leads to the question of convergence. Since a wireless receiver has to work in a variety of scenarios, convergence has to be quantified for a probability distribution of the multidimensional radio channel. The question of convergence of receiver processing for a channel distribution necessitates a stochastic analysis. The last chapters illustrated a vast design space for iterative receiver algorithms. An interesting objective is to search for the Pareto-efficient [Par] algorithms which determine the accuracy/complexity tradeoff — this comprises removing all Pareto-inefficient algorithm candidates and algorithm combinations. Monte Carlo simulation of complete receiver processing of any iterative processing scheme is too slow for the large design space. In contrast to ’slow’ link-level simulation, a ’faster’ convergence prediction method is needed.

Extrinsic information transfer charts (EXIT charts) are widely used for predicting and illustrating convergence of iterative decoding of concatenated codes [ten01, BRG05]. The model underlying the chart assumes that the log-likelihood ratios (LLRs) of the transmit bit values are distributed after the symbol demapper according to BPSK transmission over an AWGN channel – resulting in a 1-parametric conditional Gaussian distribution (conditioned on the transmit bit value).

EXIT charts have also been used to model convergence of iterative MIMO detection-decoding. In [YW05] they are applied to optimize irregular repeat accumulate codes for MIMO transmission and iterative receiver processing. An optimization of Turbo coded space-time block code transmission based on EXIT charts is presented in [UYLW09]. [HSM05] uses EXIT charts to analyse and optimize MIMO transmission with low-density parity-check codes. EXIT charts have also been used to optimize activation order of demapper, decoder and channel estimator in iterative receivers in [BRG05, YKJ10].

On the other hand, in [Fu05] it is argued that even if the input of a log-APP decoder follows a 1-parametric Gaussian distribution, the output needs to be described by two parameters (mean and variance) to adequately represent the dynamics of Turbo decoding. [RHV07] presents an analytic model of the MIMO MMSE interference cancelling demapper in terms of the transfer of means and variances of the demapper input to those of the output.

This chapter elaborates on the applicability of the stochastic decoding analysis methods. Following [Fu05, RHV07], a two-parametric chart based prediction method is used. Prediction offset compensation is used to account for higher order distribution moments. The prediction method is verified to yield acceptable prediction accuracy for different receiver computation schedules for the case of iterative MIMO detection-decoding with Turbo codes [IB10].

5.4.1 Parametric Tracking of LLR Density Evolution

For illustration it is referred to the iterative demapping-decoding setup shown in Fig. 2.3 on 29. For this case with three nodes, the order of factor node updates is arbitrary (which was pointed out in the context of iterative decoding of arbitrarily concatenated codes in [BMDP98, BRG05]). The joint APP approximation is given by:

$$L_p(u_i) = L_e^{(det)}(u_i) + L_e^{(dec1)}(u_i) + L_e^{(dec2)}(u_i) \tag{5.4}$$

The aim is to predict convergence of iterative receiver processing for any schedule. The approach is to track the conditional LLR distributions corresponding to the messages in Fig. 2.3 for all node updates. Receiver accuracy is then given by the mutual information (MI) between the $L_p(u)$ and the transmit bits u.

To evaluate the accuracy of the presented prediction method for concrete demapper/decoder schemes, the following common algorithms are picked: the constituent decoders perform log-APP decoding according to the BCJR algorithm [BCJR74], the MIMO demapper uses max-log-APP detection [HtB03]. For the exemplary channel distribution, uncorr. Rayleigh fading for each time instance t, and noise variance σ_N^2 is assumed. Three different schedules are arbitrarily picked, for which the prediction accuracy is assessed:

– schedule 1: 'normal' receiver with Turbo decoder. First the demapper is updated once, then the constituent decoders are run alternatingly.

– schedule 2: the demapper is run first, and then again always after four Turbo decoder iterations (eight constituent decoder updates).

– schedule 3: ’round-robin’ schedule. Demapper, decoder 1 and decoder 2 are run periodically in this order (demapper update after each Turbo decoder iteration).

For simulation, 4x4 QPSK transmission and channel coding with the 3GPP LTE Turbo code (rate 1/3) is assumed.

Shortcomings of 1-parametric model EXIT charts [ten01, BRG05] are based on a 1-parametric conditional Gaussian distribution model of LLRs. This model is derived from the assumption of BPSK transmission over an AWGN channel:

$$y = x(b) + n \tag{5.5}$$

Under this assumption the extrinsic LLRs generated by the demapper follow a (conditional) Gaussian distribution with the special property that the (conditional) absolute expectancy value is half of the (conditional) variance [ten01]:

$$\left|\mathbb{E}\left(L_e^{(det)}(b)\middle|b\right)\right| = \frac{1}{2}\mathbb{V}\left(L_e^{(det)}(b)\middle|b\right) \tag{5.6}$$

An LLR distribution is therefore completely described by one parameter, e.g. by the standard deviation σ. As consequence, there is a bidirectional mapping $J : \sigma \mapsto I$ between this parameter and the mutual information carried by this distribution (MI of LLRs with the transmit bits, Eq. (3.33)). This mapping is the basis of EXIT charts [ten01]. EXIT charts assume that the 1-parametric distribution property is sustained after a BCJR decoder. The parameter transfer $I(L_a) \mapsto I(L_e)$ is tabularised in a table T, its graph is the EXIT curve. To track LLR density evolution for convergence prediction, $I(L_e)$ can be looked up from this table for known $I(L_a)$ for information bits and code bits:

$$I_e(b) = T(I_a(u), I_a(c)) \tag{5.7}$$

The 1-parametric property (Eq. (5.6)) is also sustained for summation of LLRs, since mean and variance of the sum distribution are the sum of the means and variances, respectively. The MI of the LLR sum can therefore be determined by using J^{-1} and adding the variances [ten01]:

$$I_{sum}(u) = J\Big(\sqrt{\sum_i J^{-1}(I_e^{(i)}(u))^2}\Big) \tag{5.8}$$

To see why this model is not adequate in the scenario at hand, EXIT charts are applied to predict

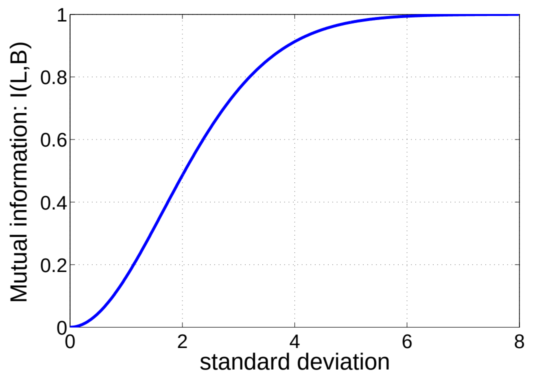


Figure 5.1: Bijective mapping between standard deviation and MI for the 1-parametric Gaussian model underlying EXIT charts (after [ten01]).

convergence of schedule 1 ('normal' receiver, no iterative demapping) for channel SNR of 1dB. The prediction of MI after each factor node update is shown in Fig. 5.2. The figure also shows the measured MI, which is obtained by Monte Carlo simulation of the complete receiver processing and non-parametric conditional LLR distribution estimation after each factor update number. While EXIT charts predict convergence after 8 node updates, measurement shows a saturation at MI of 0.53. An EXIT chart prediction for 0dB channel SNR predicts saturation at higher MI than 0.53. The prediction error in this case is therefore larger than 1dB, which is so large that it renders the prediction method useless.

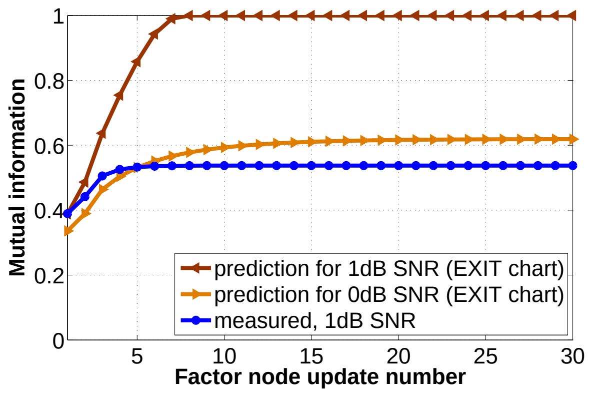


Figure 5.2: For the Rayleigh fading MIMO channel, EXIT chart based prediction produces a large error; in this case (4x4 QPSK, max-log-APP demapper) the prediction error corresponds to more than 1dB channel SNR. Simulation uses maximum LTE packet length of 6144 information bits [IB10].

The misprediction is explained by the actual LLR distribution after the demapper (max-log-APP demapping, uncorr. Rayleigh fading), which is shown in Fig. 5.3. While it does resemble a conditional Gaussian distribution, Eq. (5.6) is clearly violated: the mean value is not half the variance. Fig. 5.3 also shows a conditional Gaussian distribution with the same MI which satisfies Eq. (5.6) (mean and variance are different from the measured distribution). This is the curve which EXIT chart prediction assumes for this MI value, and it is the reason for the wrong prediction trend. The problem is not that the demapper or decoder EXIT curves would be wrong: histogram based measurement of the extrinsic MI as in [ten01] is indeed correct. The problematic 1-parametric fitting occurs when the output LLRs become input for the next factor node, because the EXIT curves are computed with 1-parametric input distributions.

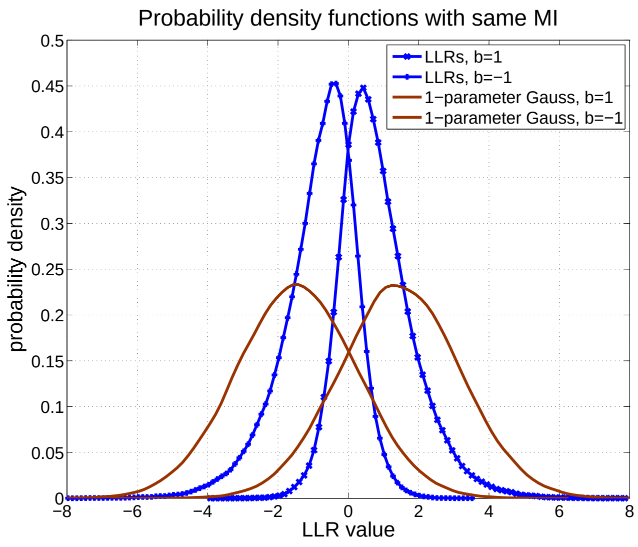


Figure 5.3: LLR conditional probability density function for 4x4 QPSK MIMO transmission with uncorrelated Rayleigh fading and max-log-APP demapping. The corresponding conditional density according to the 1-parametric Gaussian model is also shown: both densities have the same MI with the transmit bits [IB10].

5.4.2 2-Parametric Gaussian Model and Offset Compensation

It is noted that while EXIT charts track the MI value corresponding to an LLR distribution, they could equivalently track a different parameter describing the 1-parametric Gaussian distribution, e.g. the standard deviation [Fu05].

Until now the conclusion has been drawn that the 1-parametric Gaussian model where the expectancy μ is half the variance σ^2 (Eq. (5.6)) is not adequate in the scenario at hand. But it could still be the case that another 1-parametric model, maybe with a nonlinear relation between μ and σ^2, can be used. To test this, Monte Carlo simulation of the complete receiver processing is run according to schedule 3 ('round robin'), and μ and σ of the LLR distributions are measured after each factor update number. Looking at the value pairs of μ and σ, the result is that a 1-parametric description does not work. The evolution of mean and variance of the information bit a posteriori LLRs is shown in Fig. 5.4. Since a high mean value at low variance implies high mutual information, the MI growth through processing can be qualitatively observed.

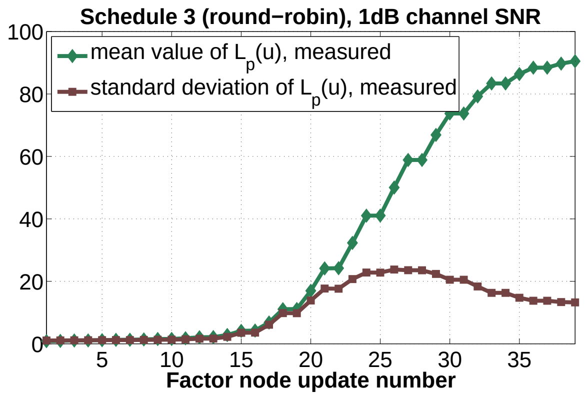


Figure 5.4: The progress of iterative processing can be observed in this example: large mean value and small standard deviation means that the conditional LLR distribution has high mutual information with the transmit bits. Every third factor update in this schedule is the MIMO demapper, for which the chart shows no increase of LLR mean value but a small reduction of standard deviation, corresponding to interstream interference reduction.

Therefore one parameter is added to the model and in accordance with [Fu05, RHV07] it is assumed that the LLRs are conditionally Gaussian distributed with arbitrary mean μ and standard deviation σ, leaving out Eq. (5.6). Table look-ups for the extrinsic information transfer of decoders or demapper now have more dimensions: based on mean and standard deviation of the input distributions, the mean and standard deviation of the extrinsic output distribution are looked up. A decoder look-up becomes:

$$(\mu_e(b),\ \sigma_e(b)) = T\Big((\mu_a(u),\ \sigma_a(u)); (\mu_a(c),\ \sigma_a(c))\Big) \tag{5.9}$$

The MIMO demapper look-up in the scenario at hand has six input values (three input vectors with two parameters each, compare Fig. 2.3, page 29).

The mapping from distribution parameters (μ, σ) to MI (function J) now has one dimension more. MI of the Gaussian distribution is only determined by the ratio $q = \mu/\sigma$ of mean value and standard deviation, the corresponding bit error rate (for a posteriori LLRs) is given by the tail probability [Fu05]:

$$BER = Q(q) \tag{5.10}$$

Therefore as coordinates for the 2-dimensional mapping function, mean value μ, quotient $q = \mu/\sigma$, and MI are used:

$$J : (\mu, \frac{\mu}{\sigma}) \mapsto I \tag{5.11}$$

The function is illustrated in Fig. 5.5. The figure also shows the curve for the 1-parametric case, embedded as special case in the MI surface.

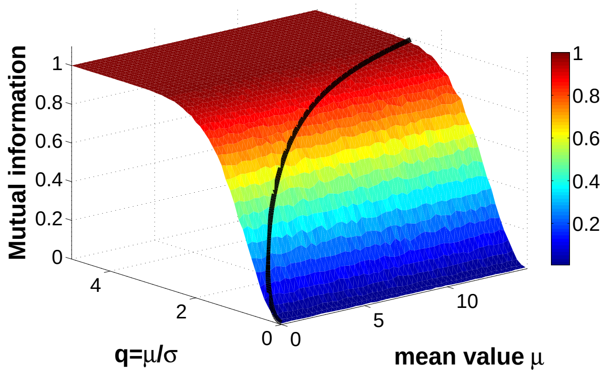


Figure 5.5: Mapping LLR distribution parameters to mutual information. The mutual information is determined by the ratio q of mean value and standard deviation. 'Full' MI corresponding to BER smaller 10^{-4} is achieved for $q > 3.7$. The 1-parametric model is included as special case and shown as curve in the MI surface [IB10].

A BER smaller than 10^{-4} corresponds to $q > 3.7$. Fig. 5.5 therefore also shows the parameter range which has to be covered by the look-up tables. Since there are infinitely many Gaussian distributions with same q, the function J is no longer invertible. Due to this, the distributions are tracked for iterative decoding using only their Gaussian parameters μ and σ, the mapping to MI (or BER) is only necessary when the iterations are stopped. For a sum of LLRs it is now instead of Eq. (5.8):

$$\begin{aligned} \mu^{(sum)}(u) &= \sum_i \mu^{(i)}(u) \\ \sigma^{(sum)}(u) &= \sqrt{\sum_i (\sigma^{(i)})^2} \end{aligned} \tag{5.12}$$

i.e. the sum is still (conditionally) Gaussian distributed.

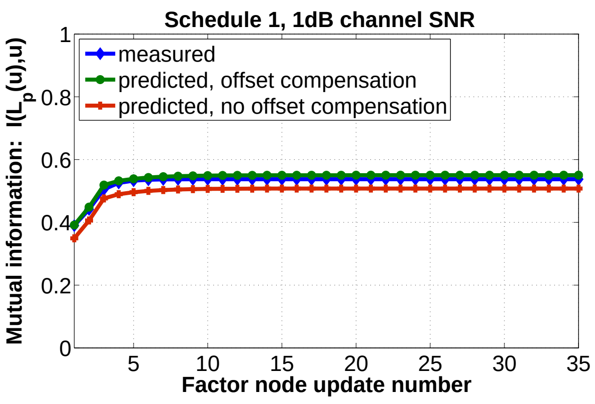


Figure 5.6: Mutual information of a posteriori LLRs during iterative decoding (factor updates) according to Fig. 2.3, and MI prediction. Node update number 1 is the demapper, then the constituent decoders are iteratively updated.

Compensating MI Offset for Higher Order LLR Distribution Moments

Prediction accuracy of the 2-parametric model is illustrated in Fig. 5.6. As expected, the more flexible 2-parametric model reproduces the actual MI evolution trend and yields better accuracy – but beginning from the first demapping, the prediction has an MI offset compared to the measured MI. This offset can be explained by the fact that the MIMO demapper LLRs do not exactly follow a Gaussian distribution: not all cumulants of the distribution for order larger than 2 are zero. This is illustrated in Fig. 5.7. The figure shows the LLR distribution from the MIMO example as well as the Gaussian distribution which has the same mean and variance. The measured LLR distribution shows a nonzero skewness, it is not symmetric. MI of the assumed Gaussian distribution is smaller, causing the initial prediction offset. The assumed Gaussian distribution can either have the same mean and variance as the real distribution, or the same MI - but not both.

For a consistent concatenation of table look-ups the demapper table is determined using the Gaussian distribution with same mean and variance as the real one. To compensate the inital MI loss, it is also computed at table generation time. For one channel SNR value, the demapper table now is a mapping

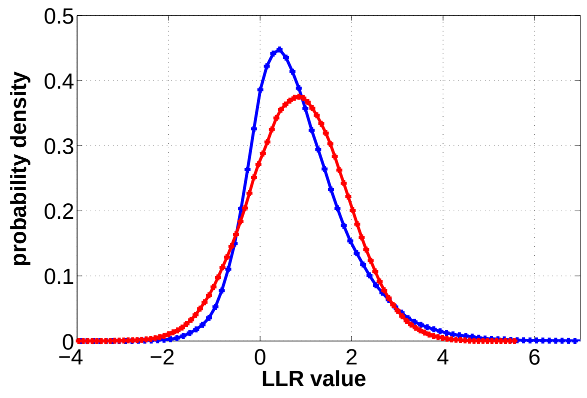


Figure 5.7: LLR probability density for positive transmit bits from the example (as in Fig. 5.3) and the corresponding 2-parametric Gaussian density. The two densities have same mean and variance, but different MI (LLRs: 0.39; Gauss: 0.34) [IB10].

from 6 input dimensions to 3 output dimensions (compare Fig. 2.3):

$$(\mu_e(b),\ \sigma_e(b),\ I_{\text{offset}}) = T\Big((\mu_a(u), \sigma_a(u));\ (\mu_a(c_1), \sigma_a(c_1));\ (\mu_a(c_2), \sigma_a(c_2))\Big) \tag{5.13}$$

Adding channel SNR as input dimension makes the demapper table input 7-dimensional. For the presented prediction results, the input LLR distributions are sampled with 8 points per dimension ($0 \le \mu \le 15,\ 0 \le q \le 5$), resulting in 260000 entries in the demapper table per channel SNR value. Using the fact that the roles of u, c_1 and c_2 are interchangeable for the demapper, only 46000 table entries have to be computed. The table for a constituent decoder was already described earlier in this chapter (4 input dimensions to 2 output dimensions). Since the two constituent decoders are identical for the LTE Turbo code used in the scenario at hand, they are both described by the same table. For table look-ups, linear interpolation between neighbouring sample points is used. The demapper look-up table depends on channel SNR, a constituent decoder table is independent of this.

The predicted Gaussian parameters (μ_p, σ_p) of the distribution of the a posteriori LLRs $L_p(u)$ are then mapped to MI by table look-up (function J), and the I_{offset} value returned by the last demapper table look-up for $L_e^{(det)}(b)$ is added:

$$I_{\text{predict}} = J(\mu_p, \frac{\mu_p}{\sigma_p}) + I_{\text{offset}} \tag{5.14}$$

5.4.3 Mutual Information Prediction Accuracy for Different Schedules

MI prediction accuracy is verified by comparison with MI measurement, for the three example receiver processing schedules. ’Prediction’ uses the described concatenation of table look-ups, where the concatenation order of look-ups from the two tables is determined by the schedule. ’Measurement’ performs Monte Carlo simulation of the complete receiver and measures MI using non-parametric estimation of the joint distribution of a posteriori LLRs and transmit bits according to Eq. (3.33), independently for each schedule.

The results of prediction and measurement are shown in Fig. 5.8. Schedule 1 ('normal' receiver) does not converge for this low SNR level, which is now correctly predicted. The MI of a posteriori LLRs saturates after around 7 factor computations (6 constituent decoder updates) at 0.53. Schedule 2 converges after around 40 factor computations (including 5 demapper updates and 35 constituent decoder updates). A demapper update only brings a small MI improvement in itself, but afterwards decoder updates gain more again. Schedule 3 ('round-robin') converges already with around 25 factor computations.

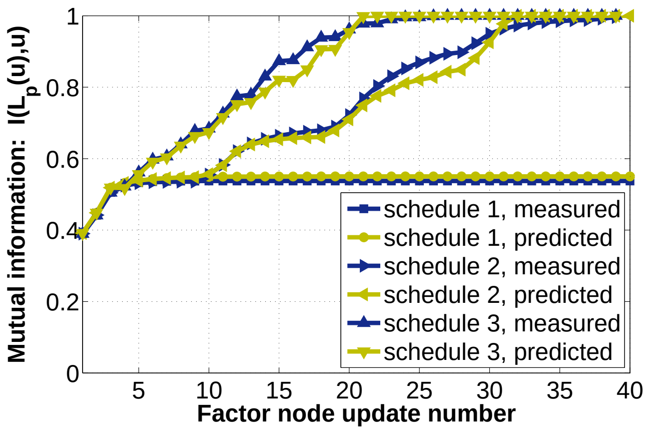


Figure 5.8: Verifying MI prediction accuracy: predicted and measured MI for the three different schedules (packet length 6144 information bits) [IB10].

All periodic schedules which include the same factors converge to the same MI limit value [BRG05], since they completely use the same information sources. The maximum MI value which can be reached by the extrinsic MIMO demapper output $L_e^{(det)}$ is that of SIMO maximum ratio combining for (shifted) BPSK modulation [IKB09]: if the demapper a priori LLRs $L_a^{(det)}$ have full MI (implying that the receiver algorithm has already converged), for each LLR to compute all transmit bits of the MIMO vector are known except one, meaning that only two symbol constellation points remain.

The MI prediction curves in Fig. 5.8 do show small deviations from the also shown measurement curves, which are due to higher order cumulants (order higher than 2) of LLR distributions and finite granularity of the look-up tables.

5.4.4 Verifying BER and Threshold Prediction

Prediction of the APP LLR distribution includes bit error rate (BER) prediction according to Eq. (5.10). To verify BER and SNR threshold prediction, this mapping from the LLR distribution to BER is applied for the two models and compared with measurement for very long packets. For the proposed method it is:

$$BER = Q(\frac{\mu_p}{\sigma_p}), \tag{5.15}$$

while for EXIT chart based prediction this reduces to one parameter:

$$BER = Q(\sqrt{\frac{\mu_p}{2}}) \tag{5.16}$$

Prediction and measurement for varying SNR (for a fixed schedule) are evaluated with focus on the SNR threshold required for a target BER like e.g. 10^{-4}. Fig. 5.9 illustrates results for the 'normal' schedule with 21 factor updates. As implied by MI prediction (Fig. 5.2), EXIT charts predict the threshold for this schedule more than 1.5dB too small, while the proposed method predicts it 0.1dB too high. For BER prediction, no compensation is applied to the MI offset, as this would affect the complete BER curve and not only the BER threshold. MI offset causes the SNR threshold to be predicted too high.

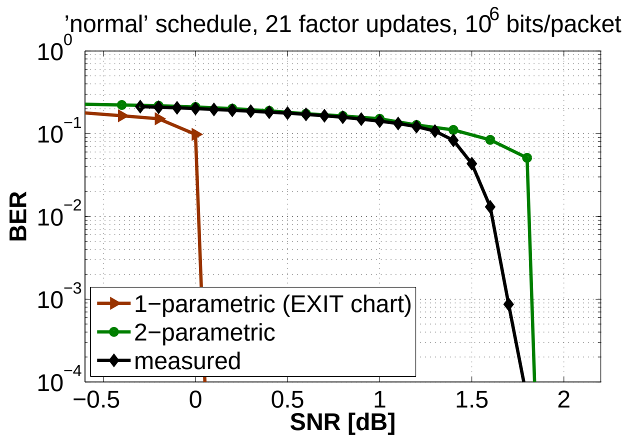


Figure 5.9: Predicted and measured bit error rate for one example schedule, using very long packets (10^6 bits). The curves for smaller packet length (6144 information bits) differ only insignificantly [IB10].

To sum up, EXIT charts in the normal way as applied to AWGN channels are not applicable to some practically relevant scenarios with fading MIMO channels. How well the underlying 1-parametric model fits the demapper LLR distribution depends on the demapper algorithm, modulation and MIMO fading distribution. This may explain why the presented results [IB10] seem to differ from e.g. [HSM05], where a 'good match' was found between simulation and EXIT chart based prediction in a different scenario. The 2-parameter extension improves prediction accuracy by better fitting to the real LLR distribution. Together with offset compensation for higher order distribution moments it achieves satisfactory MI prediction accuracy. For non-Gaussian distributions a systematic error remains (higher order moments), so that prediction accuracy is less accurate than for AWGN channels. Prediction accuracy for other channel models – especially intersymbol interference (ISI) channels – has not been investigated. The proposed method is however applicable to MIMO-OFDM, as OFDM converts an ISI channel into a set of individually flat fading channels. The higher dimensionality of the extended charts causes the charts to be less illustrative. Complexity of look-up table computation increases due to the higher dimensionality. On the other hand, computational effort is reduced again a bit by the parametric density estimation: estimating mean and variance is faster than estimating MI (with non-parametric density estimation like histograms or kernel methods). This could also be used for computation of normal EXIT charts, as it is also consistent with the 1-parametric model. In principle, the prediction accuracy can be improved by increasing the number of parameters used to describe LLR distributions: look-up tables could be extended to include higher order moments. This is limited in practice by the time necessary to compute the tables, the advantage of fast prediction compared to slow link-level simulation would erode. The presented prediction method serves as a basis for receiver optimization at receiver design time (choice of algorithms and processing schedule) in chapter 6. Comparing all receivers for the described scenario (three factor nodes) which have a schedule length of exactly 20 factor node updates (10^6 different receivers) may well be too much for link-level simulation based comparison. Using the presented method, all of them can be compared after generating only two look-up tables. Comparison of different factor computation algorithms (especially demapper algorithm alternatives) can be done by changing the respective factor look-up table. A criterion for optimization can be the sum computational cost for reaching the target MI (corresponding to a required packet error rate) at a certain SNR. The prediction accuracy of the proposed method is sufficient to reduce the receiver design space to a few interesting algorithm candidates, which can then be verified by more time-consuming link-level simulation.

Chapter 6

Automatic Receiver Optimization

Several publications deal with the problem of optimizing the node activation order in an iterative receiver (or more generally in a graphical model). It can be distinguished between a statistical optimization (pre-determining the best activation order at design time for a PDF of received vectors) and between a run-time optimization (determining the best activation order for one concrete received vector). For a statistical activation order optimization of demapper, decoder and channel estimator, [BRG05, YKJ10] use EXIT charts. [EMK06] describes a greedy approximation which orders the nodes to update by the difference in output message magnitude compared to their last update. It can be used both for design-time and for run-time optimization. For statistical optimization, the greedy ordering can also be applied to MI increase or to attainment of an objective function (e.g. including complexity) instead of message magnitude. The greedy activation ordering from [EMK06] has been applied to iterative demapping-decoding in [ZLNA10].

This chapter deals with joint optimization of component algorithm selection and activation order at design-time. It is therefore based on the generic receiver architecture and receiver description language described in chapter 2 and chapter 4, the component algorithm alternatives discussed in chapter 3 and the fast performance prediction method from chapter 5. The aim of finding the 'best' receiver with respect to a certain optimization criterion, where the receiver components are selected from the large body of published algorithms, faces several difficulties. While the structure of a receiver is clear and follows from the interrelations of transmission variables, the optimization lies in the selection and concatenation of component algorithms. For each receiver component, a whole 'algorithm zoo' is available, containing algorithm candidates which are Pareto-optimal [Par] regarding an accuracy/complexity tradeoff. Attainment of receiver 'optimality' further requires evaluation of a component in the complete receiver: while receiver complexity is the sum of component complexities, there is no such simple relation for the accuracy. For optimization of iterative receivers, usage of different component algorithms in each iteration is possible. The criterion for receiver optimality can be chosen as utility value — a combination of receiver operational SNR, complexity, packet processing delay, throughput and number of processor cores. Operational SNR can be determined using the method from chapter 5 as that SNR level (for assumed channel distribution), where BER or MI cross a threshold. The receiver description language from chapter 4 allows enumeration of the design space (using branch and bound graph search), where for each receiver candidate its utility (according to the chosen criterion) can be quickly assessed. Automatic receiver optimization is illustrated in an example scenario, where the optimized receivers show both an extended operational SNR range compared to the standard receiver architecture, as well as significantly reduced complexity compared to iterative processing according to round-robin iteration scheduling using the same algorithm components. Further, minimum processing delay in dependence on the number of parallel processing elements is discussed, as well as the relation between complexity and delay when parallelizing on the factor level.

6.1 Optimization Criteria

The task is an optimal distribution of computational power to a number of homogeneous multiprocessor cores (compare SDR baseband hardware model from Sec. 1.6), where the number of cores as well as the update schedule and component algorithm for each factor update are flexible. For given transmission mode (modulation, MIMO scheme and code rate) and channel characteristics, receiver processing quality can be described by the performance parameters from chapter 5. The general optimization target function is a utility function from these performance parameters. This utility function may contain constraints like e.g. maximum delay. In the 5-dimensional (target SNR, complexity, processing delay, number of processor cores and throughput) Pareto-optimal receiver algorithm utility space, the following optimization criteria could be chosen (among any other weighted combinations):

1. Minimum complexity for fixed operational SNR. Which receiver satisfies the operational requirements with minimum computational effort? The answer may be interesting as complexity is related to power consumption.
2. Minimum operational SNR. Given a fixed hardware with certain computational power, what are the achievable operating conditions (and with which algorithms can this be reached)?
3. Minimum delay for fixed operational SNR. How far can the processing delay be reduced by parallelizing node updates using multiple processor cores?

6.2 5D Pareto Efficiency: Searching in the Decision Tree

All schedules can be seen as different paths in a decision tree. One node in this tree is a decision at a certain time – about which factors to update with which algorithms on how many PEs. A new decision is always done when a factor update is completed. The decisions of one decision path in the tree are therefore ordered by increasing decision times.

Part of the decision tree for 6 PEs is illustrated in Fig. 6.1. One path in the tree (one parallel schedule) is highlighted as example with red nodes. The schedule is described by the list of corresponding decisions (written next to the nodes).

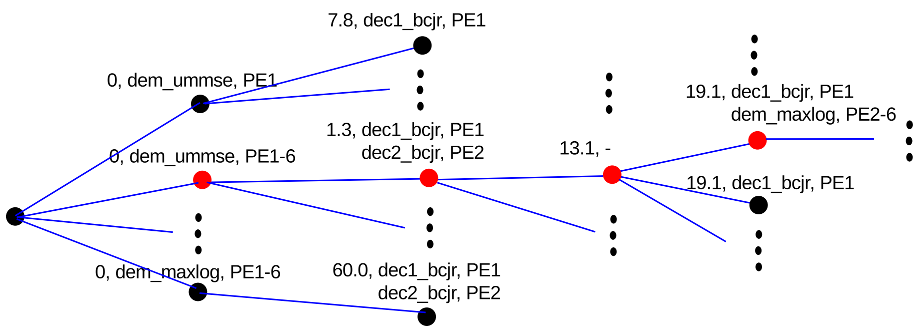


Figure 6.1: An example decision tree for 6 PEs. For each decision the start cycle/LLR, factors to update and corresponding algorithms with PE mapping are noted. One decision path (parallel schedule) is highlighted with red decision nodes.

All possible decisions which are children of a node in the decision tree can be enumerated. A factor can be updated if it received at least one new input message, and if this factor is not currently active being updated. The output of a factor update enables activation of other factors according to the factor graph and their current state (like process scheduling in a non-preemptive operating system, compare Fig. 4.2). All parallel schedules can be enumerated by tree traversal.

The Pareto efficient receivers differ in the five dimensions operational SNR, complexity, delay, number of processing elements and throughput. Search for the ’best’ receiver or Pareto efficient receiver sets is implemented by level-order traversal of the decision tree (using a list of decision paths) with branch and bound. For each pair of operational SNR and number of PEs, a search tree is opened. Node expansion needs to enumerate combinations of ready factors, which can be done in a binary way as illstrated in Tab. 6.1. Then the Cartesian product of candidate algorithms is needed, which can implement the different factors to activate. A Cartesian product implementation is shown in Fig. 6.2.

Adding a decision to a decision path can not decrease delay or complexity, so that branch and bound can be used conveniently with a utility measure (one target function as measure) or in a search for Pareto efficient receivers (several measures at once).

No.	Factor 1	Factor 2	Factor 3	No. Factors
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	2
4	1	0	0	1
5	1	0	1	2
6	1	1	0	2
7	1	1	1	3

Table 6.1: Enumeration of combinations of three factors which can be activated (having new input and not being currently active).

Figure 6.2: Matlab implementation of Cartesian product.

```
function p = CartesianProduct(listvector)
  k = length(listvector); % number of dimensions
  for i=1:k
    num_elem(i) = length(listvector{i});
  end
  num_cp_elem = prod(num_elem);
  for i=1:k
    num_elem_restdim = prod(num_elem(i+1:k));
    col = kron(listvector{i}, ones(num_elem_restdim,1));
    repetitions = num_cp_elem / length(col);
    cp(:,i) = kron(ones(repetitions,1),col);
  end
end
```

6.3 Example Search Results

This section illustrates the automatic receiver optimization for Cell SPU as target hardware by giving an exemplary optimization of iterative demapping-decoding for 4x4 QPSK, assuming perfect channel estimation. Resulting block complexities (in cycles/LLR) are illustrated in Fig. 6.3a for uMMSE and Max-Log demapper, and for decoder 1 and decoder 2 using the BCJR algorithm. The decoder 2 block is a bit more complex then decoder 1 because Turbo interleaver and Turbo deinterleaver are counted as belonging to this block. For the Max-Log demapper's complexity, independent computation of all candidate vector metrics is assumed (no usage of intermediate partial metrics).

6.3.1 Operational SNR versus Receiver Complexity

Minimum receiver complexity is achieved with a serial computation schedule. Considered are concatenations of the four processing blocks illustrated in Fig. 6.3a. For each SNR the objective is to find the cheapest (least complex) schedule which achieves convergence, defined here as $MI > 0.99$ (equivalently the BER for convergence can be defined, termed 'just acceptable error rate' in [LAMC11] in the context of channel adaptive MIMO demapper switching). The following three receiver spaces are considered:

- $\mathcal{R}_{\text{normal}} = (\texttt{dem\_ummse})(\texttt{dec1\_bcjr}\ \ \texttt{dec2\_bcjr})^*$

  'Normal' receiver with Turbo decoder. The number of Turbo decoder iterations is variable.
- $\mathcal{R}_{\text{round-robin}} = (\texttt{dem\_maxlog}\ \ \texttt{dec1\_bcjr}\ \ \texttt{dec2\_bcjr})^*$

  Iterative MIMO demapping-decoding using the Max-Log demapper and a round-robin schedule.
- $\mathcal{R}_{\text{iterative-1}} = (\texttt{dem\_ummse} \mid \texttt{dem\_maxlog} \mid \texttt{dec1\_bcjr} \mid$
  $\texttt{dec2\_bcjr})^*$

  All possible concatenations of the four blocks are used as input to the optimization.

The receiver space $\mathcal{R}_{\text{iterative-1}}$ is illustrated in Fig. 6.3b in form of a state transition graph (finite state machine) containing the processing blocks as states.

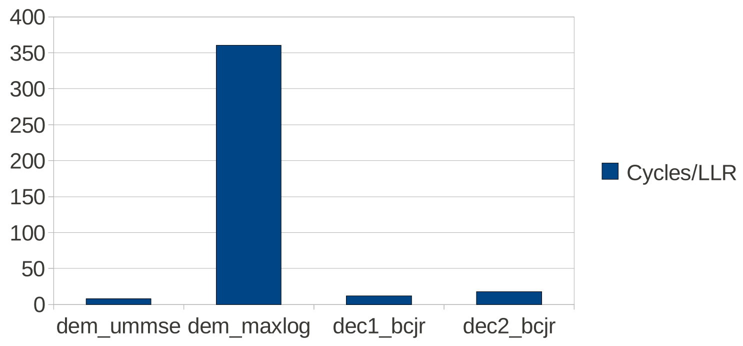
The optimization consists of predicting performance for each receiver visited in the search tree. For the search space $\mathcal{R}_{\text{iterative-1}}$, the branch-and-bound costs can be initialized with the results from $\mathcal{R}_{\text{round-robin}}$: if there is a converging receiver for this SNR, then there is a converging receiver with round-robin schedule ([BRG05] – although it is probably not the optimal receiver according to the chosen criterion).

Optimization results are shown in Fig. 6.3c:

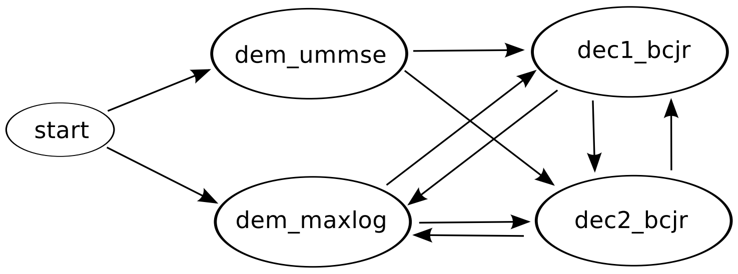
- standard receivers according to $\mathcal{R}_{\text{normal}}$ are computationally inexpensive, but do not work at very low SNR. In this scenario (4x4 QPSK, Rayleigh fading, rate 1/3), the necessary number of Turbo decoder iterations increases from 1 iteration at 5dB to 10 iterations at 2.1 dB.
- $\mathcal{R}_{\text{round-robin}}$ extends the operational SNR by 1 dB to the low SNR regime, although at largely increasing computational cost.
- optimization among $\mathcal{R}_{\text{iterative-1}}$ reduces the necessary cost compared to $\mathcal{R}_{\text{round-robin}}$ by a factor of 2 with the same operational SNRs. At the same complexity the operational SNR is reduced by around 0.5 dB. The standard receiver architecture $\mathcal{R}_{\text{normal}}$ is included as special case and is result of the optimization for the high SNR regime.

The Pareto-optimal receivers lie on what seems like a hyperbola in the operational SNR / complexity diagram. Receivers inside the hyperbola are suboptimal, dominated by the Pareto-optimal ones. Which receiver from the hyperbola is the optimal one depends on the chosen criterion. To the lower SNR region (left in the diagram), the hyperbola is limited by channel capacity (if arbitrary processing is allowed) and transmitter characteristics. The hyperbola may be improved (to the lower left of the diagram) by improving one of the contained APP computation components (towards better accuracy of APP approximation or towards complexity reduction). The optimization process itself does not have to change to include new algorithm developments.

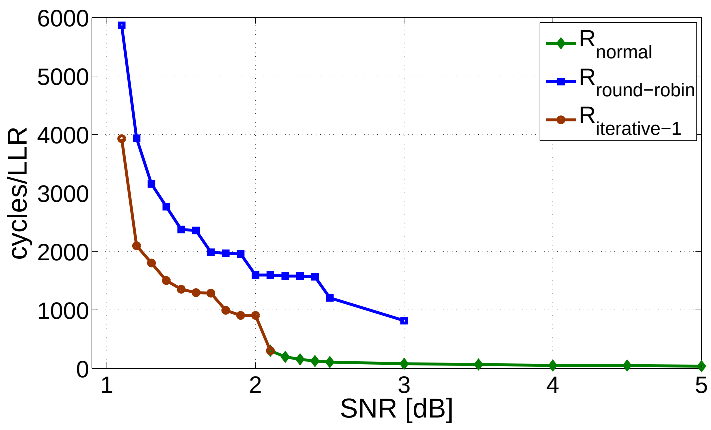
The number of factor node updates of the found ’optimal’ receivers in the example is 3 for 5dB, 47 for $\mathcal{R}_{\text{round-robin}}$ at 1.1 dB and 56 for $\mathcal{R}_{\text{iterative-1}}$ at 1.1 dB. $\mathcal{R}_{\text{iterative-1}}$ contains $4^{56} > 10^{33}$ receivers of schedule length exactly 56. These numbers clearly show the necessity of an optimization approach based on fast performance prediction, as it would have been unfeasible to obtain results for the presented optimization problem by Monte-Carlo simulation of the receivers. Receiver design spaces further grow exponentially if more algorithm alternatives are considered and if the optimization approach is extended to also include iterative channel estimation algorithms.



(a) Computational effort of the considered signal processing and decoding blocks on Cell SPU.



(b) Example search graph and finite state automaton for example receiver design (sub-)space with serial schedule. Can be seen as ’folded’ decision tree for 1 PE. Each state can be an end state.



(c) Results for 4x4 QPSK.

Figure 6.3: Automatic receiver optimization example [IB11].

6.3.2 Processing Delay versus Number of Processing Elements

An example search is run to find the receiver with minimum delay in dependence on the number of available PEs, for 3dB target SNR. The optimum serial schedule (1 PE) found is:
`dem_ummse` (`dec1_bcjr` `dec2_bcjr`)^2 `dec1_bcjr`
i.e. linear demapping followed by 5 constituent decoder updates, with a sum delay of $78.5 cycles/LLR$. The minimum delays for different number of PEs are shown in Fig. 6.4. The solutions follow the same processing as the serial schedule, only the demappers are parallelized using the available PEs. This can be explained with the comparably high target SNR (compare Fig. 6.3c). Max-Log demapping in this case can not replace one of the five decoder updates, and parallel decoder updates can not replace the last decoder update. The minimum delay in this case therefore follows Amdahl's law:

$$T(N_{PE}) = \frac{T_{par}}{N_{PE}} + T_{ser} \tag{6.1}$$

where T_{par} is the part which can be parallelized (demapping) and T_{ser} is the serial computation part (decoding). The asymptote in Fig. 6.4 lies at $T_{ser} = 70.75 cycles/LLR$. More complicated parallel schedules like in Fig. 4.1 are only relevant for low SNR.

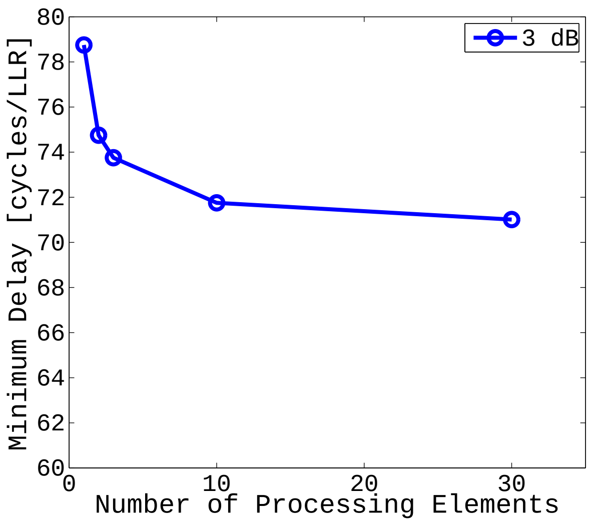


Figure 6.4: Minimum delay in dependence on number of processing elements, for 3dB target SNR.

6.3.3 Receiver Complexity versus Processing Delay

For the same 3dB example the tradeoff between complexity and delay is illustrated in Fig. 6.5. For this high SNR case, only one receiver is Pareto efficient, resulting in the rectangular Pareto efficiency curves in Fig. 6.5. With variable number of PEs, the delay can be reduced according to Amdahl's law by parallelizing the demappers — at the same receiver complexity, resulting in a horizontal Pareto efficiency curve.

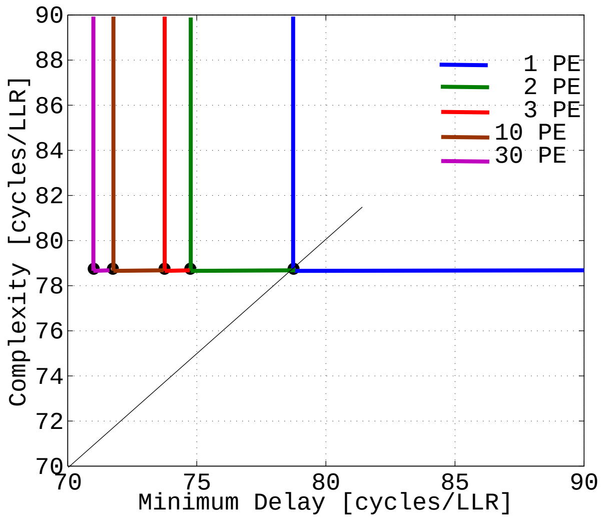


Figure 6.5: Complexity and delay of Pareto efficient receivers for different number of PEs, for 3dB.

Notes for extensions Apart from applying the presented approach for comparing and optimizing receivers including different component algorithms not covered by the examples, four main extensions are possible:

– *multiple transmitter or multi-codeword*: optimization for this case is also possible (compare Fig. 2.2 on page 27 from chapter 2). The determination of the optimal receiver processing sequence is analogous to the single user / single codeword case, with the following differences:
  – different SNRs per user / transmitter are allowed, leading to different (input) receive value densities.
  – some blocks jointly process signals of the different transmitters or codewords (like MU-MIMO demapping), some blocks separately (like decoding).
  – BER is determined per transmitter / receiver pair.
– *including channel estimation*: The optimization can also be extended to include possibly iterative channel estimation. The assumption of Gaussian densities (mean, variance) can not only be applied to model conditional LLR densities, but also to model baseband symbol densities (complex Gaussian densities). Input to the channel estimation prediction are the receive value density $P(y)$, the soft transmit symbol density $P(x)$ and of course the channel statistics (R, σ_N^2). The resulting density $P(H)$ (represented as channel estimation MSE under the assumption of bias-free estimation) is also parameterized as input in the demapper table – which adds one more dimension there.
– *faster tree search for very wide decision trees*: considering a large number of candidate algorithms,

many PEs and low SNR (requiring many updates), the tree becomes too large to search for the optimal solution. In this case suboptimal search strategies like e.g. the M-algorithm can be applied. The problem of too large search spaces and the application of heuristic search strategies is common for search based software design [Har07].

– *PEs with limited local store:* For static instantiation, the local store implies a limit for the sum code size of factor update algorithms run by a PE. This constraint can be taken care of during tree search. The other possibility is dynamic instantiation – with the effect of additional delays between ’factor task’ switches.

Chapter 7

Medium Access Control Aspects

Main task of the medium access control layer is scheduling. Input to the base station's scheduler includes uplink bandwidth requests, downlink buffer filling levels and channel quality information. Uplink channel information is obtained by channel sounding schemes commanded by the base station, and by channel estimation in the resources where a terminal transmits. Downlink channel information is obtained by a terminal by channel estimation, and fed back to the base station over a control channel. Scheduling output is the resource allocation to users, and choice of transmission parameters.

When several users share radio resources, the achievable rates become a multi-dimensional region. Scheduling should clearly select a point on the surface of this region (Pareto-efficient allocation) – but the decision which point to select is to a certain degree arbitrary. Section 7.1 describes scheduling criteria, which are based on different sets of plausible arguments and single out a point from the Pareto-efficient surface of the achievable rate region.

Sec. 7.2 formulates scheduling as combinatorial optimization problem. Since scheduling according to most optimization criteria is NP hard, approximative algorithms are also discussed.

Signalling schemes can only achieve limited and slightly outdated channel knowledge at the transmitter. So Sec. 7.3 deals with channel interpolation and prediction, and scheduling under channel uncertainty. The implication of limits for the gains of adapting transmission parameters is quantified.

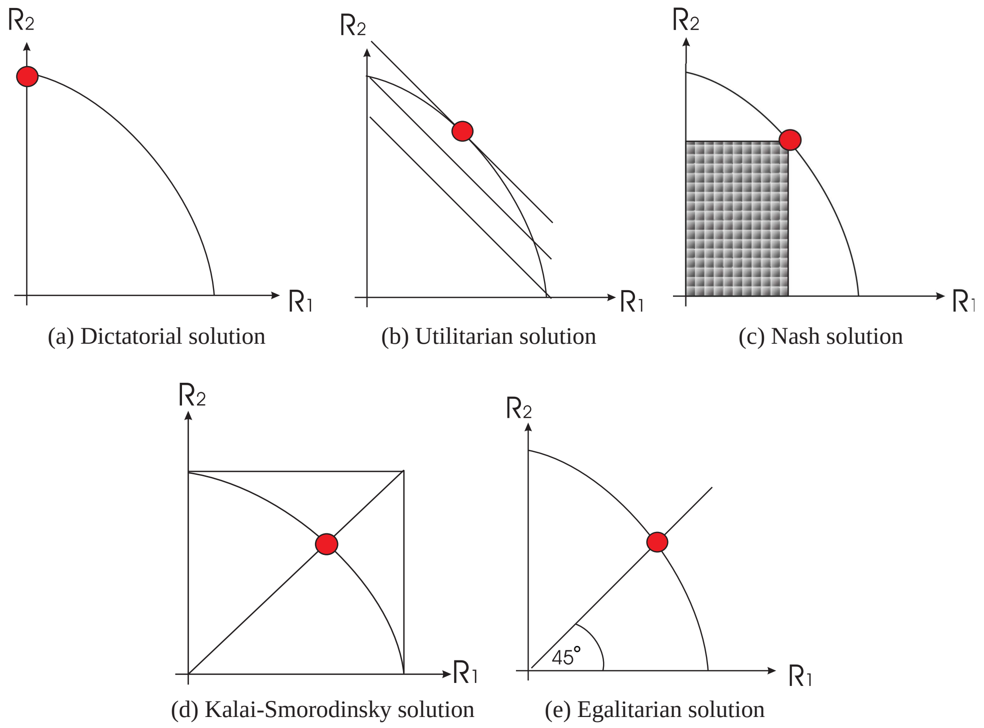
In Sec. 7.4 receiver computational power is treated as variable which can be distributing over parallel messages. This section builds on chapter 6 and shows a potential gain by adding receiver computation as scheduling variable.

Sec. 7.5 deals with protocol extensions for multi-hop relaying, which reduce signalling overhead without violating the scheduling criterion.

7.1 Scheduling Criteria: Fairness vs. Efficiency

Classical scheduling goals in a communication system are to maximize utilization (throughput) and to allow communication for all users (fairness). These two goals are contradictory when like in wireless communications the same physical resource can have different benefits for different users. Game theory derived different solutions to the general problem of distributing some good in a way considering both efficiency and fairness and delivered an axiomatic characterization of these different solutions. In OFDMA systems, different subcarriers can be dynamically allocated to different users. Because of normally different channel quality of the same subcarrier for different users, a clever scheduler can exploit multiuser diversity [RC00]. A good channel quality subcarrier can be higher modulated and thus carry a higher data rate. This section characterizes the efficiency/fairness-tradeoff in OFDMA resource allocation by applying as schedulers and comparing the most important game theoretic solutions for cooperative bargaining.

Applying bargaining theory to scheduling has been proposed by [MMD91]. [KMT98] has proposed proportional fairness, which has been implemented in a TDMA system in [JPP00]. Proportional fairness for multicarrier systems has been e.g. considered in [KH05]. The Kalai-Smorodinsky solution as scheduler has been proposed for multimedia streaming in [PS07]. Scheduling approaches regarding the maximal stability region for elastic traffic also lead to utility based solutions in the game theoretic sense [BW07].



(a) Dictatorial solution (b) Utilitarian solution (c) Nash solution

(d) Kalai-Smorodinsky solution (e) Egalitarian solution

Figure 7.1: Comparison of common scheduling criteria [Tho94, IB07].

7.1.1 Game Theoretic Cooperative Bargaining Solutions and Axiomatic Classification

Assume some good is to be divided between participants. The amount that participant j gets is denoted R_j. The good can be split up in different ways, the set of all feasible choices is denoted $\mathcal{U}$. Each possible solution $F(\mathcal{U})$ is a vector $\mathbf{R} \in \mathcal{U} \subset \mathbb{R}^n$. Each of the game theoretic solutions is defined by a small number of axioms. These axioms can be regarded as normative objectives of fairness [Tho94]. While more solutions have been proposed and characterized, the four described here [IB07] play a central role in bargaining theory.

The Utilitarian solution maximizes the sum of gains (compare Fig. 7.1b). It is defined by the following three axioms [Tho94]:

- Pareto-optimality: $\nexists \mathbf{R}' \in \mathcal{U}$ with $\mathbf{R}' \geq \mathbf{R}$

  All gains should be exhausted, i.e. the solution lies on the boundary of the feasible set.
- Symmetry: If $\mathcal{U}$ is invariant under all exchanges of participants, then $F_i(\mathcal{U}) = F_j(\mathcal{U})$.

  If the participants can not be differentiated on the basis of the feasible set, then the solution should treat them the same.
- Linearity: $F(\mathcal{U}^1 + \mathcal{U}^2) = F(\mathcal{U}^1) + F(\mathcal{U}^2)$

  Participants are indifferent between solving problems separately or consolidating them into a single problem and solving that problem.

The Utilitarian solution is not always unique. When it is not unique, continuity also is an issue, i.e. small changes in the feasible set should lead to small changes of the solution.

The Nash solution is obtained by maximizing the product of benefits $\prod R_j$ (compare Fig. 7.1c). It is the only solution satisfying [Tho94]:

- Pareto-optimality
- Symmetry
- Scale invariance: $\lambda(F(\mathcal{U})) = F(\lambda(\mathcal{U}))$

  A participant by participant scale dilation results in dilation of the solution with the same factors.
- Contraction independence: If $\mathcal{U}' \subseteq \mathcal{U}$ and $F(\mathcal{U}) \in \mathcal{U}'$, then $F(\mathcal{U}') = F(\mathcal{U})$

  If an alternative is the best solution for some problem, then it should still be the best solution for any subproblem that contains it.

The Kalai-Smorodinsky solution Gains are proportional to their maximal possible values (compare Fig. 7.1d), which are given by the Dictatorial solutions (Fig. 7.1a). This solution has an interesting monotonicity property: expansion of the feasible set in a direction favorable to a particular participant always benefits him. The solution is the maximal point of $\mathcal{U}$ on the segment connecting the origin to the 'ideal point' **a** defined by $a_j = max\{R_j | \mathbf{R} \in \mathcal{U}\}$ (where every participant would get his maximum possible benefit, i.e. his Dictatorial solution). The solution is uniquely defined by the following axioms [Tho94]:

- Pareto-optimality
- Symmetry
- Scale invariance
- Individual monotonicity (for n=2): if $\mathcal{U}' \supseteq \mathcal{U}$ and $R_j(\mathcal{U}') = R_j(\mathcal{U})$ for $j \neq i$, then $F_i(\mathcal{U}') \geq F_i(\mathcal{U})$.

By simply replacing contraction independence with individual monotonicity in the list of axioms characterizing the Nash solution, the Kalai-Smorodinsky solution is obtained.

The Egalitarian solution In the Egalitarian solution all gains are equal (compare Fig. 7.1e). All participants benefit from any expansion of opportunities. The solution is the only one satisfying [Tho94]:

- Pareto-optimality
- Symmetry
- Strong monotonicity: if $\mathcal{U}' \supseteq \mathcal{U}$, then $F(\mathcal{U}') \geq F(\mathcal{U})$.

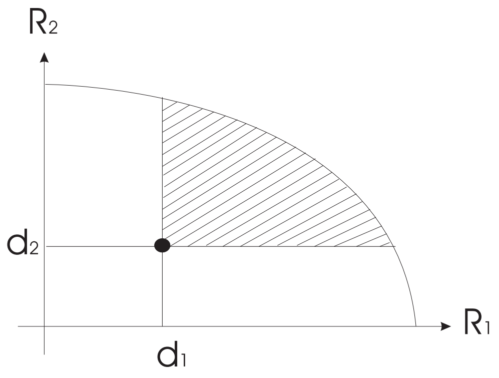


Figure 7.2: Disagreement point **d**: the benefit R_j of participant j must be higher than the disagreement point coordinate d_j, otherwise the participant will leave the negotiation (minimum requirement) [SB07, IB07].

The Disagreement Point A disagreement point $\mathbf{d} \in \mathcal{U}$ can be considered, a solution must then satisfy $R_j \geq d_j$, otherwise user j would leave the negotiation (see Fig. 7.2). The disagreement point was ignored before, which can be described as disagreement point equal to zero. The Nash solution with disagreement point for example maximizes $\prod_j (R_j - d_j)$. An interesting property which the Nash solution satisfies is strong individual rationality, meaning that all users strictly gain from compromise (gains are larger than the disagreement point). The Utilitarian solution does not satisfy strong individual rationality [Tho94].

7.1.2 Application as Schedulers

The good to be distributed is data rate. The amount of data that user j receives/transmits during one scheduling interval is R_j. The allocation of resources is a mapping (according to the selected criterion) which determines the operating point of the system.

Maximum throughput: Utilitarian solution To maximize system throughput, every subcarrier is allocated to the user for which the channel quality allows the highest data rate. The scheduling is:

$$S^{maxTP} = \underset{\mathcal{U}}{\operatorname{argmax}} \sum_j R_j \tag{7.1}$$

This solution is maximally efficient, but unfair: users with bad channel quality (e.g. cell edge users) may be completely excluded from communication.

Proportional fairness: Nash solution Proportional fairness is currently the most popular scheduling criterion. It is:

$$S^{PF} = \underset{\mathcal{U}}{\operatorname{argmax}} \sum_j \log R_j = \underset{\mathcal{U}}{\operatorname{argmax}} \prod_j R_j \tag{7.2}$$

Using this scheduling, all users are guaranteed to receive some resources.

Kalai-Smorodinsky fair scheduling Coordinate a_j of the ideal point $\mathbf{a}$ is the data rate that user j would achieve if only he was scheduled on the whole frequency band. The scheduler can be implemented as a weighted max-min scheduler, where coordinate j is stretched with the factor $1/a_j$:

$$S^{KS} = \underset{\mathcal{U}}{\operatorname{argmax}} \left\{ \min_j \left(\frac{R_j}{R_{j,\max}} \right) \right\} \tag{7.3}$$

If the channel quality of a user improves, he will get a higher data rate without any reduction for the other users (individual monotonicity). Also here, all users are guaranteed to receive some resources.

Max-min fairness: Egalitarian solution As the name indicates, the max-min fair scheduler is

$$S^{mm} = \underset{\mathcal{U}}{\operatorname{argmax}} \left\{ \min_j R_j \right\} \tag{7.4}$$

It is the maximally fair scheduler (all users get the same data rate), but a user with bad channel quality limits system performance. If the channel quality of one user improves, all users will get a higher data rate (strong monotonicity).

Unelastic Traffic: Use of the Disagreement Point Different traffic classes and prioritization can be included in a scheduler using the disagreement point. Users may have minimum rate requirements due to unelastic traffic – streaming data like VoIP, videos etc. Such a hierarchical ressource allocation can also be used in time direction: e.g. longer-term scheduling grants, which are partly used to reduce signalling overhead, describe unelastic traffic. The users' minimum rate requirements can be modelled as disagreement point before bargaining is applied.

Illustration for OFDMA

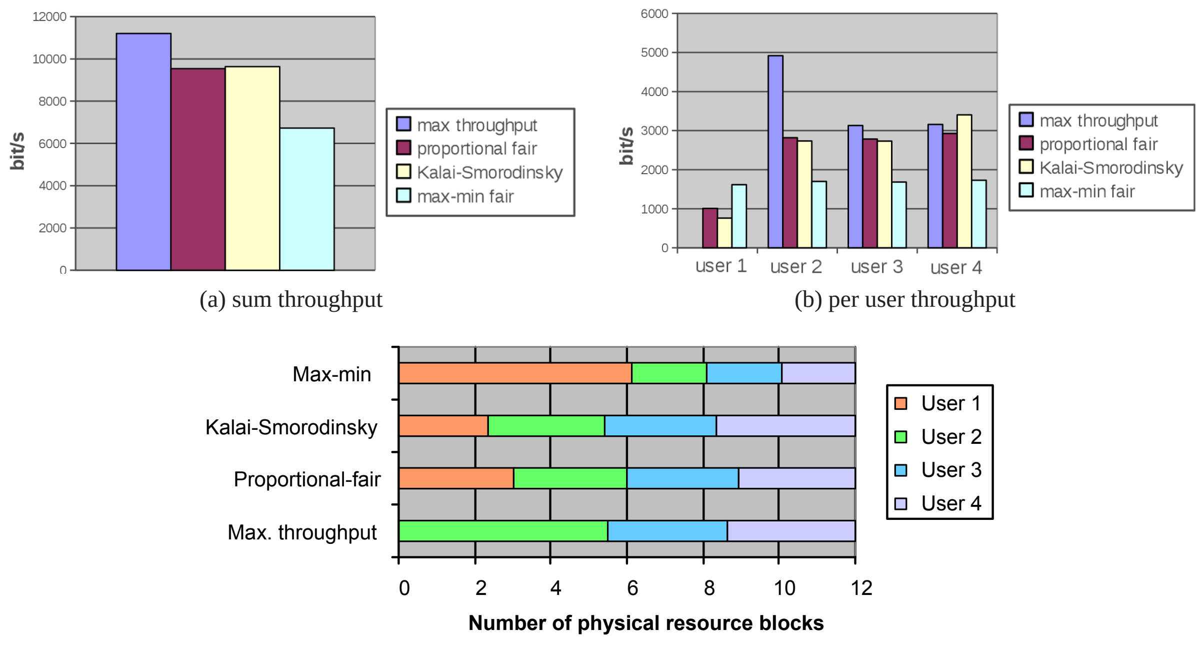
The illustration assumes 12 ressource blocks in frequency direction with 84 ressource elements (symbols) each, and four users. The full buffer model is used (all stations always want to transmit). The users' channel qualities per ressource block are asuumed independent, and the probability distributions given in Tab. 7.1.

Efficiency: Sum throughput First, scenario 1 is considered. The resulting average sum throughputs of the four schedulers are shown in figure 7.3a. The proportional-fair scheduler and the Kalai-Smorodinsky scheduler lie between the max-throughput and max-min schedulers. Kalai-Smorodinsky scheduling achieves a slightly higher sum throughput than proportional-fair scheduling.

Fairness: Per user throughput Figure 7.3b shows average data rates per user for the same scenario. The max-throughput scheduler never schedules user 1, because there is always another user with better channel. Max-min scheduling allocates equal rates. Proportional fairness in this scenario allocates less

	Scenario 1			
	not usable	QPSK	16QAM	64 QAM
User 1	50%	50%	0%	0%
User 2	0%	33%	33%	33%
User 3	0%	33%	33%	33%
User 4	0%	0%	50%	50%
	Scenario 2			
	not usable	QPSK	16QAM	64 QAM
all users	25%	25%	25%	25%

Table 7.1: Scenarios for scheduler comparison.



(a) sum throughput

(b) per user throughput

(c) frequency band sharing

Figure 7.3: Scheduler comparison [IB07].

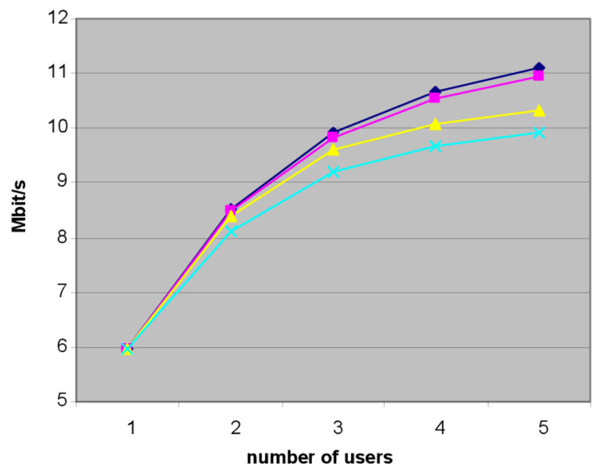
differing rates than Kalai-Smorodinsky, which fits to a smaller sum rate (Fig. 7.3a). In cases where the max-throughput solution is not unique, the implementation gives preference to smaller user indices, which explains the high data rate of user 2 and comparatively small data rate of user 4.

Frequency band sharing The average number of allocated resource blocks per user (not their positions in the frequency band) as consequence of the scheduling done in utility space (data rates) is shown in figure 7.3c. Max-min scheduling allocates on average more than half of the whole frequency band to user 1 to achieve equal rates, while max-throughput scheduling ignores this user completely.

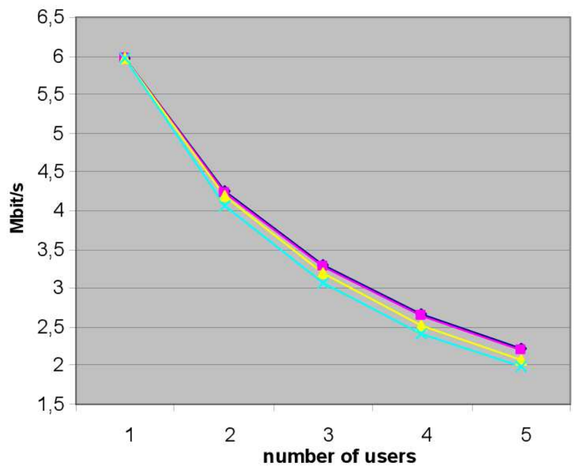
Scaling with the number of users Now, scenario 2 is considered. Expected sum throughput is shown in fig. 7.4a. All schedulers take advantage of multiuser diversity (increasing sum rates with the number of users), and of course max-throughput scheduling exploits it most. The distance to the max-throughput curve is ’fairness loss’. In this scenario of equal distributions proportional fairness is more efficient than the Kalai-Smorodinsky solution.

Average user rate: Scaling of the average per user rate is shown in figure 7.4b. The achieved average rates are very similar in this scenario.

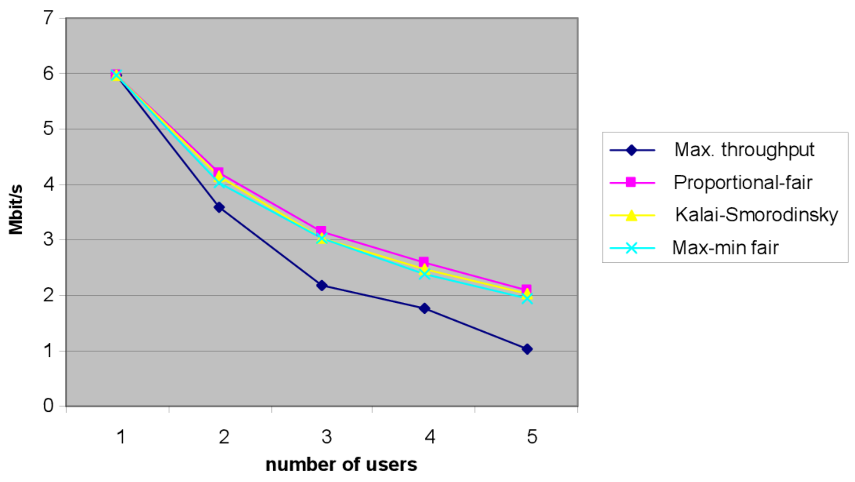
Minimum user rate is shown in fig. 7.4c. Note that in another scenario the max-throughput scheduler’s minimum rate can be equal to zero for user numbers larger than one. The implementation uses independent scheduling for different scheduling intervals. It is also possible to consider achieved rates of previous scheduling intervals (scheduler memory), which becomes a necessity when there are more users than resources in a scheduling interval.



(a) expected sum throughput: multiuser diversity and fairness loss



(b) expected average throughput: bandwidth sharing



(c) expected minimum user rate: fairness

Figure 7.4: Scaling with the number of users [IB07].

Comparison The Utilitarian solution is the only linear solution. As soon as any fairness criterion is used, the problem becomes nonlinear. All four schedulers are Pareto efficient. It is up to the operator to decide how much capacity he trades for fairness. Depending on algorithm implementation this could be adjustable during operation. It would also be possible to provide an interface to the operator where he could implicitly choose the scheduling algorithm by selecting his favoured fairness and efficiency rules. Scheduling according to the Kalai-Smorodinsky solution is an alternative to proportional fairness, both offer a compromise between efficiency and fairness. Which one of them is more efficient depends on the scenario. Other utility functions are of course also possible, but they are not special in a sense that they are uniquely defined by a small number of common sense axioms like the game theoretic solutions.

7.2 Joint Scheduling and Link Adaptation

In this section, uplink and downlink channel knowledge is assumed.

7.2.1 Optimization Problem

Scheduling is performed for MIMO-OFDMA in a 3-dimensional PRB index area $\mathcal{A} \subset \mathbb{N}^3$.

Variables to optimize:

- matrix $\mathbf{u}$ allocating physical ressource block indices to user indices.
- matrix $\mathbf{m}$ describing the PRBs' modulation indices. Same modulation inside a PRB is assumed.
- per subcarrier matrix $\mathbf{w}_{i,j,k}$ of weight vectors. These matrices also describe the power allocation (by amplitude of the weight vectors).
- code rate $\mathbf{c}$ per user. Assuming constant code rate for one user's ressources in a scheduling interval means uniform puncturing pattern for this packet.

Derived variables:

- SINR $\gamma_{l,i,j,k}$ after the demapper per user and subcarrier.
- number of (coded) bits per PRB in dependence on modulation: $B(m)$.
- vector $\mathbf{R}$ of user rates in $\mathcal{A}$.

Target function: All scheduling criteria can be written as utility functions $U(\mathbf{R})$, which a solution should maximize.

To maximize throughput, the utility function would be

$$U_{\mathrm{maxTP}} = \sum_{j=1}^{k} R_j \tag{7.5}$$

For proportional fairness the utility function is [KMT98]

$$U_{\mathrm{PF}} = \sum_{j=1}^{k} \log R_j = \prod_{j=1}^{k} R_j \tag{7.6}$$

For max-min fairness the utility function is

$$U_{MM} = \min_j R_j \tag{7.7}$$

Optimal scheduling means selecting user combinations (per PRB), modulations (per PRB), weight vectors (per subcarrier) and code rates (per user) which maximize the utility:

$$(\mathbf{u}, \mathbf{m}, \mathbf{w}_{i,j,k}, \mathbf{c}) = \underset{(\mathbf{u},\mathbf{m},\mathbf{w}_{i,j,k},\mathbf{c})}{argmax} \; U(\mathbf{R}) \tag{7.8}$$

Achieved utility can be determined for one choice of allocation parameters by the following steps:

1. compute power per user and subcarrier: $P_i(w)$.
2. using channel knowledge, compute SINRs per subcarrier and user: $\gamma_{l.i.j.k}(u, w)$.
3. using knowledge about the receiver algorithm, compute expected mutual information per subcarrier after the demapper: $I(\gamma_{()}, m)$
4. compute expected MI per user: $\overline{I}$
5. determine code rate individually per user: $c(\overline{I})$, for packet error rate of choice
6. compute expected user rate: $R_i = \sum_A B(q) * c_i u \delta_i$
7. compute utility from user rates $U(\mathbf{R})$

It is implicitly assumed that a code block is completely contained in the scheduling area.

Downlink constraint is a sum power constraint for the base station. The power constraint may take a special form, like limitation of equivalent isotropically radiated power (EIRP).

Uplink constraints are per user power constraints (possibly also EIRP).

Additional constraints differ according to scenario, protocol or system. They often aim at reducing signalling overhead. Additional constraints may include:

– contiguous allocations
– same modulation per user allocation
– constant power allocation
– fixed weight vectors. Applied in downlink, feedback is reduced. Leads to decoupling of parallel MIMO channels, so that SINR feedback can be independently reported per transmit stream.
– coding block size equivalent to user's allocation in one subframe
– single MIMO stream per user

7.2.2 Approximative Scheduling Algorithms

The optimization problem contains discrete variables, so it is a combinatorial problem. Since exhaustive search needs unreasonably high complexity for larger number of users, suboptimal approximative algorithms are of interest. Algorithms either operate in the discrete variable domain, or in a correspondig continuous relaxation.

Discrete algorithms Search strategies for discrete optimization are described in [PS82]. Since the set of feasible decisions is assumed too large for an exhaustive search, an algorithm tries to avoid looking at most of the possible decisions. Possible approaches include search space partitioning (dimensionality reduction, e.g. first subcarriers, then power), greedy, informed or exhaustive search in subspaces and iterative local search. For single-user bitloading, [HH] iteratively assigns one more bit to that subcarrier where the least amount of power is necessary, which leads to the optimal solution. For multi-user bitloading, [YL00] decomposes the problem by first determining the number of subcarriers and amount of power per user, then assigning subcarriers and loading bits with the Hungarian method. In [HJL05] an iterative algorithm is proposed, which exchanges subcarriers between pairs of users in each step.

Continuously relaxed algorithms Relaxation leads to a Lagrange or Karush-Kuhn-Tucker (KKT) formulation as constrained nonlinear optimization problem. (relaxation mean e.g. infinite granularity in modulation levels). Standard solvers are sequential quadratic programming, using active sets, interior point method or the trust-region reflective algorithm. After the continuous solution, the 'closest' discrete solution needs to be found, e.g. using cutting planes and branch and bound (branch and cut method). [KRJ00] relaxes single-user bitloading problem to a linear continuous one and finds the optimal solution with less complexity than [HH]. For the multi-user case, [YSC07] gives a Lagrange based iterative algorithm, [SAL99] employs a barrier-function based interior point method. [WCLM99] minimizes power for given rate requirements using Lagrange relaxation and iterations to satisfy requirements.

Separation of short-term allocation and longer-term weight adjustment Equal algorithmic treatment of the different scheduling criteria can be enabled using a linearization of U as weighted sum rate:

$$U(\mathbf{R}) = \sum_{j=1}^{k} v_j R_j \tag{7.9}$$

This of course necessitates computation and updates of weights v_j, which can possibly be done in a different time frame.

Example algorithm using iterative local search

In this subsection, an iterative algorithm for the multiuser fair scheduling problem of adaptive OFDMA systems is described. It uses iterative local search with k-opt switches in the combinatorial solution space. The algorithm can be used with different scheduling criteria like proportional fairness and max-min fairness, both for constant and adaptive allocation of power to subcarriers/resource blocks. The algorithm is initialized by a greedy heuristic, which iteratively allocates the resource block and modulation level with most marginal utility (allocations are iteratively done in a way to maximize utility function increase in each step). This gives a reasonably good allocation, but is suboptimal because of the nonlinearity of a fair utility function. If adaptive power allocation is used, initialization needs to also take the power restriction into account. After initialization, the iterative local search starts. In each step, allocated resource blocks are switched between users (and modulation levels), if this increases the utility function. A 1-opt-switch means taking away a resource block from one user and allocating it to another one (in case of adaptive power allocation, user and modulation are jointly considered). A k-opt-switch is a sequence of k switches. Searching for k-opt-switches would mean higher complexity but may avoid termination in a local optimum. The algorithm can also be used with a lower-complexity initialization if needed. The utility function can be any function of the R_j, e.g. max throughput, max-min fairness or proportional fairness.

Algorithm initialisation using greedy heuristic Free resource blocks which could be allocated to one user are kept in a sorted queue for this user. The queue is sorted according to the marginal utility (utility increase in case of allocation) of resource blocks. Such a queue is kept for all users, the data structure worked on is thus a vector of priority queues. To find the (greedy) best next allocation in each step, the top entries of the user queues are compared. If an allocation is done, the marginal utility values in the queue of the user who got the allocation have to be updated (nonlinearity). The allocated resource block has to be removed from all users' queues. Updating marginal utility values can be accelerated by considering that there is only a fixed small number of possible values (number of modulation levels) per queue.

Equal power allocation: If transmission power is distributed equally over subcarriers, for a predefined target bit error rate the channel quality feedback can be directly mapped to a modulation level. The only question is which user gets which resource block.

Adaptive power allocation: If transmission power can be adaptively distributed over subcarriers, each resource block can be allocated to any user with arbitrary modulation level (if maximum power is not exceeded). Deciding about the greedy initialization is not straight-forward: maximizing marginal utility in each allocation step leads to always choosing maximum modulation level and the available power is rapidly consumed (not all resources can then be used for transmission). Maximizing marginal utility per power leads to always choosing the lowest modulation and probably spare power. In any way, initialization stops when all resources are allocated or when the maximum power is reached.

Iterative local search using k-opt-switches The data structure used to find a 1-opt-switch is basically an array. An array entry corresponds to a switch and contains the marginal utility of the switch (difference of utility increase for new user/modulation and lost utility of old user/modulation). After a switch, array values of the two affected users have to be updated (again a fixed small number of values, corresponding to the number of possible modulation levels). To satisfy timing requirements, the maximum number of iterations can be limited. The algorithm could also be just terminated (after the available processing time) and the result of the last iteration used.

For Equal power, a switch is done if it increases utility. If no such switch is found, the algorithm terminates in this local optimum.

For adaptive power allocation, A switch is done if it increases utility and satisfies the power constraint. Changing modulation for a user's chunk is also considered an iteration.

Complexity considerations How does the computational effort scale with the number of users and resource blocks?

Greedy initialization:

1. a) Equal power, sort all k queues with q different possible values: $O(km)$.
   b) Adaptive power, sort: $O(km \log m)$.
2. Find greedy next allocation: $O(k)$
3. Remove allocated resource block from queues: $O(km)$
4. Rescale values in queue: $O(m)$
5. Loop m-times back to 2)

The resulting complexity for greedy initialization is thus $O(km^2)$.

Iterations:

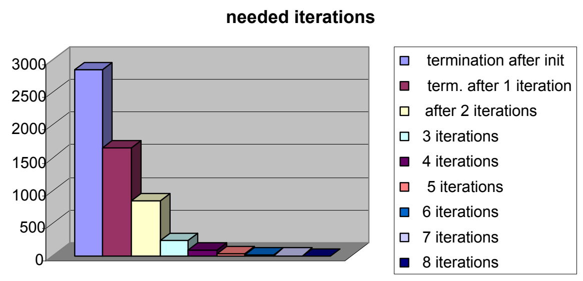
1. Build array: $O(km)$

2. Find switch: $O(km)$
3. Rescale two users' marginal utility vectors: $O(m)$
4. Loop back to 2).

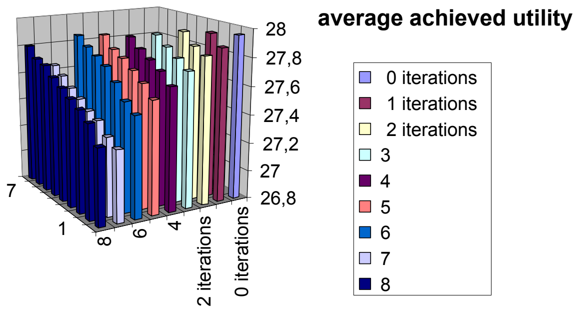
Each iteration thus has a complexity of $O(km)$.

Illustration Equal power allocation: For illustration, scenario 1 from Tab. 7.1 on page 112 is used, with proportional fairness as scheduling criterion. 5719 scheduling intervals have been simulated, both employing the algorithm for constant power and a computationally intensive search for the optimal solution (using branch-and-bound). Figure 7.5a shows a histogram of the number of iterations until termination. Greedy initialization achieves good results, in about 50% of the intervals local search is not able to improve the initialization by finding a 1-opt-switch. The algorithm almost always terminates after less than 5 iterations. Essence of the algorithm is utility improvement in each iteration. Figure 7.5b shows the average achieved utility in each step, separately for each total number of iterations until termination. The final iteration in all cases achieves similar utility values. The better (higher utility) the initialization is, the less iterations are needed.

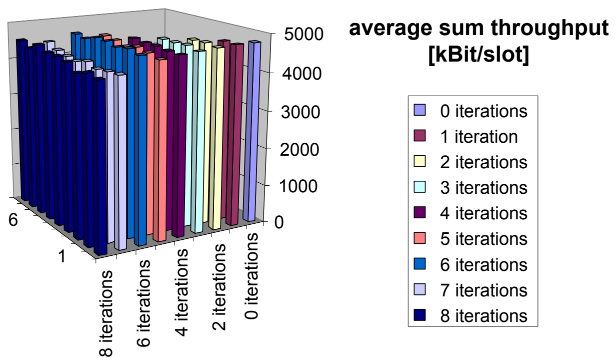
Average sum throughput in each iteration, separately for each total number of iterations, is shown in figure 7.5c. The average sum throughput is roughly constant. Iterations thus do not lead to higher sum throughput, but to more (proportional) fairness. The best way to answer the question of how far the algorithm solution is away from the optimal solution, is to compare utility values (figure 7.5b). Another way is chosen in figure 7.5d. The average mean square error (MSE) of per-user rates with respect to the optimal solution is shown for each iteration: the distance vector between the vector of user rates of the iteration and the vector of user rates of the optimal solution is computed, and its entries are squared and summed up to give the MSE. The average MSE is decreasing over iterations. If the algorithm would always find the optimal solution, the last iteration's MSE would be zero. This is however not the case due to stopping in local optima.



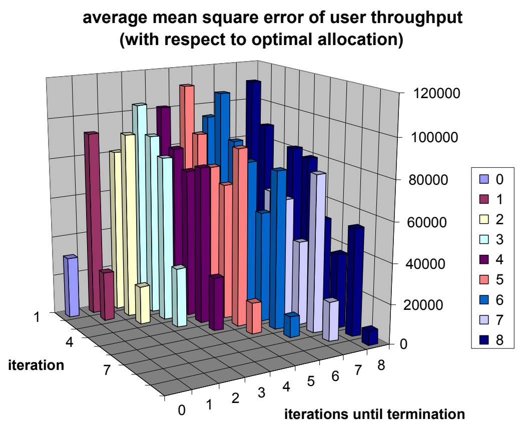
(a) (constant power) Number of scheduling intervals with algorithm termination after n iterations. Greedy initialization achieves good results: the algorithm almost always terminates after less than 5 iterations (1-opt-switches).



(b) (constant power) Average achieved utility in each step, separately for each total number of iterations until termination. Utilities of the final iterations are similar. The better the initialization is, the less iterations are done. The shown values are computed as proportional fairness measure, using the natural logarithm and each user's bits/slot as data rate. The average optimal utility has simulatively (branch-and-bound) been found to be 27,98. Colours correspond to figure 7.5a.



(c) (constant power) Average sum throughput per iteration, separately for each total number of iterations. Iterations do not lead to higher sum throughput, but to higher (proportional) fairness (cf. figure 7.5b). Colours correspond to figure 7.5a.



(d) (constant power) Average mean square error of per-user rates with respect to the optimal allocation. Another way (from utilities) to illustrate approximation of the optimal solution. The MSE is decreasing to smaller values in the last iteration (diagonal in the front).

Figure 7.5: Equal power allocation [IB08].

Adaptive Power Allocation: Now a basestation power constraint of 1250 mW is assumed. CQI feedback is simplified to be the power necessary for QPSK modulation (lowest modulation) of each resource block. The basestation computes necessary powers for 16QAM and 64QAM by adding 6dB resp. 12dB. Necessary powers for QPSK modulation are modelled as independant continuous random variables for resource blocks and with different distributions for different users. Uniform distribution in the log-power domain (dBm, decibel compared to milliWatt) is assumed. The used distributions are shown in table 7.2.

In the greedy initialization, in each step the resource block allocation with highest marginal utility per power is chosen. This leads to only QPSK or no modulation of a resource block (in case of no remaining

	Scenario 3			
	User 1	User 2	User 3	User 4
distribution from	20dBm	10dBm	10dBm	0dBm
distribution to	40dBm	30dBm	30dBm	20dBm
	Scenario 3, algorithm effect			
	not used	QPSK	16QAM	64QAM
initialisation	81	119919	0	0
last iteration	53	65074	34997	19876

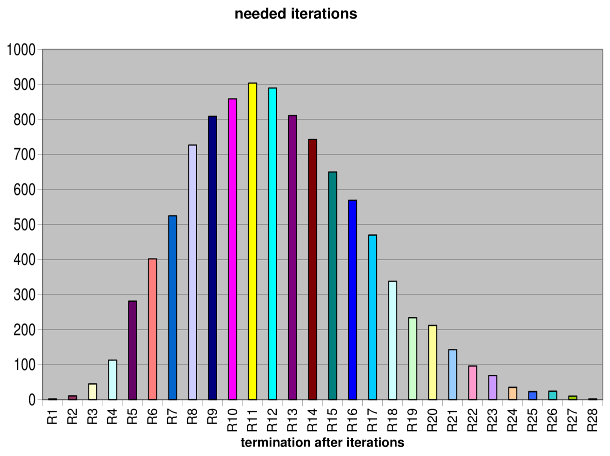
Table 7.2: Scenario for adaptive power allocation [IB08].

power). 10000 scheduling intervals have been simulated, the number of occurrences of needed iterations is illustrated in figure 7.6a, showing an expectancy value of around 13 iterations. Table 7.2 shows the number of chosen modulation levels after initialization and after algorithm termination, clarifying the mentioned property of greedy initialization with marginal utility per power and the effect of the iterative algorithm part.

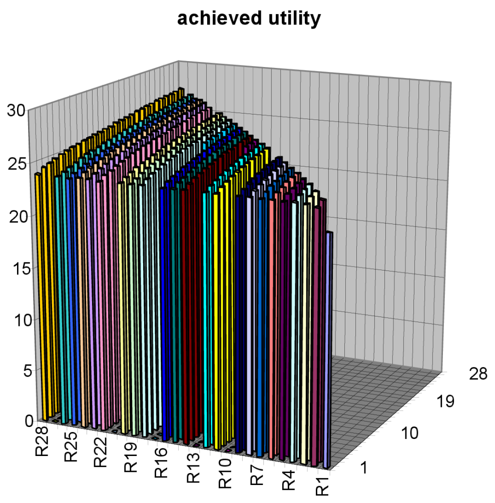
The average achieved utility values over iterations and iteration steps are shown in figure 7.6b.

Figure 7.6c shows used power levels over iterations. After the last iteration, the available 1250mW are fully used. Initially there is low power allocation due to modulation level choices of intitialization. The lower the initial power allocation is (meaning good channel qualities), the more iterations are done by the algorithm.

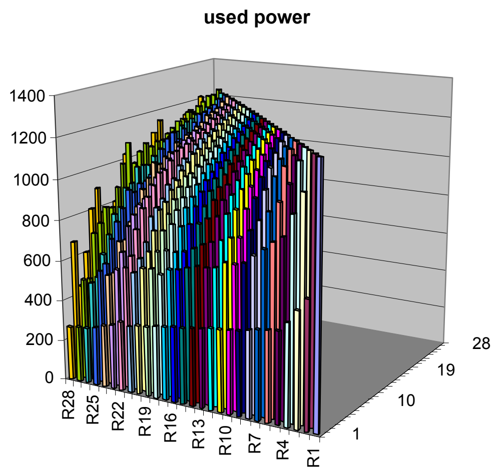
Another effect of the initially low modulation levels is that iterations considerably increase not only fairness, but also sum throughput (fig. 7.6d), which is different from the constant power case (Fig. 7.5c).



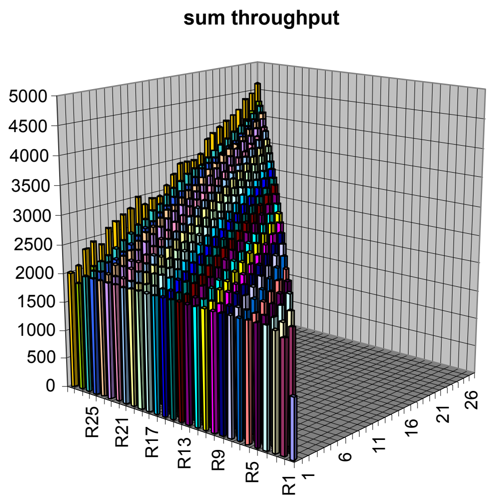
(a) adaptive power) Histogram of number of iterations until termination (1-opt-switches). The curve has a different form than in the constant power case (fig. 7.5a), because greedy initialization with marginal utility per power always chooses the lowest possible modulation level (which is later increased using iterations).



(b) (adaptive power) Average achieved utility values over iterations. Colours correspond to figure 7.6a.



(c) (adaptive power) The lower initial power allocation is (meaning good channel qualities), the more iterations are done by the algorithm. After the last iteration, the available 1250mW are fully used.



(d) (adaptive power) Sum throughput over iterations. Because of initially low modulation levels, iterations considerably increase not only fairness, but also sum throughput, which is different to the constant power case (fig. 7.5c. Colours correspond to figure 7.6a.

Figure 7.6: Adaptive power allocation [IB08].

7.3 Gains and Limits of Adaptation: Imperfect Channel Knowledge

For joint MIMO-OFDMA link adaptation and scheduling with limited signalling overhead, channel quality feedback needs to be quantized, sparse and irregular. For flexible scheduling, prediction of the quantized channel at the base-station is needed. In this section, measurement of 3D downlink channel correlation parameters by the terminals and signalling back to the base station is described, which enables prediction of the quantized channel by multi-dimensional Wiener filtering. The scheme is shown to improve adaptive choice of transmission parameters and to avoid mis-adaptation due to control lag [IOB11]. Different schemes of correlation feedback signalling are discussed in terms of (time-variant) expected throughput.

According to Sec. 1.3, time variance of a radio channel is divided into large-scale fading due to pathloss and shadowing and small-scale fading due to multipath propagation (changes over distance on the order of wavelength). Large-scale fading is traditionally countered by adaptation of transmit power. Adaptation of transmit parameters on a radio link has since been extended to also account for small-scale fading, e.g. in the form of per-subcarrier modulation in adaptive OFDM [Czy96].

Link adaptation as any control loop needs to consider adaptation delay. To cope with delay, channel prediction for adaptive OFDM based on received symbols is described e.g. in [KH00, MH02b, SA03] using Wiener filtering or Kalman filtering respectively. Link adaptation either follows a channel reciprocity assumption for time division duplex (TDD) links, or requires a feedback channel to report the fading behaviour (closed-loop adaption) and possibly also interference. To reduce feedback bandwidth, channel values can be quantized to yield the parameters of adaptation [KH00].

MIMO-OFDMA transmission used in modern systems for increased spectral efficiency, further extends the trend fowards finer-grained adaptation in two directions. In OFDMA, link adaptation can be combined with multi-user scheduling to exploit multi-user diversity in addition to time and frequency diversity [WE08]. In MIMO transmission, MIMO parameters can be adapted depending on channel correlation between different antennas [FMP+07]. Joint multi-user MIMO scheduling and link adaptation is described e.g. in [HHA07].

Feedback for the complete channel grows linearly with the number of users, while the corresponding capacity grows only double logarithmically [HHA07] – so further reduction of feedback information is necessary for joint scheduling and link adaptation. For this reduction, usage of SNR thresholds has been proposed to avoid transmitting feedback for 'bad' channel ressources [Bon04, HAGO05]. These thresholds can be adaptive, relative (quality ordering) and individual per user. [HHA07] evaluates feedback thresholds to achieve a sum feedback rate limitation. A further overview of limited feedback schemes is given in [LHL+08].

Prediction in the base station is discussed in [MK09, JSAM07]. In [JSAM07], terminals feed back SNR values to the base station, which then predicts future SNR values using a 2D Wiener filter, where the spatio-temporal downlink channel correlation is estimated from uplink transmissions. [MK09] discusses prediction in the base station based on quantized feedback. It uses linear predictive coding, where the filter parameters are determined by the base station.

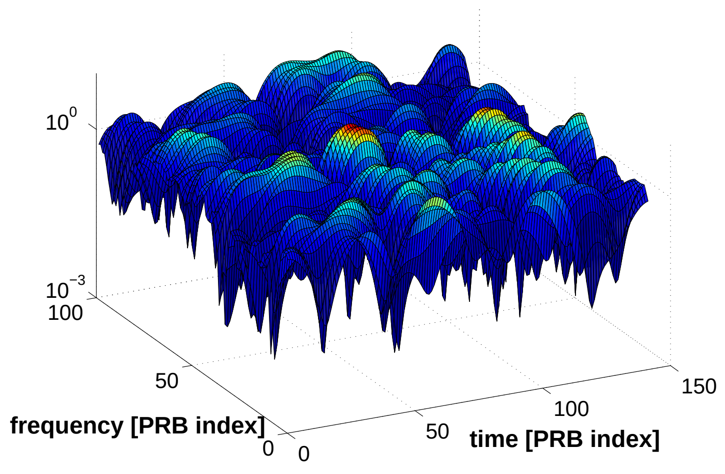


Figure 7.7: 2D fading OFDM channel, quantized to physical resource block granularity (average squared channel gain $\overline{g}$) [IOB11].

Gains of adaptation in MIMO-OFDMA transmission derive from the degrees of freedom in mapping users' tranmission to physical channel resources: user selection in time, frequency and space (MIMO streams) and choice of modulation, code rate and MIMO mode.

Limits of transmission adaptation are set by the variance of the channel (autocorrelation), delays (measurement, round-trip, signalling protocol) and feedback overhead. Delays comprise the risk of mis-adaptation: wrong choice of channel resource and modulation and coding scheme lead to inefficient transmission and high packet error rate. The aim is therefore to balance feedback overhead versus effective adaptation gain.

A recent commercial adaptation and signalling scheme for FDD MIMO-OFDMA is given by LTE [3GP10a, 3GP10b]. Downlink channel measurement uses the pilot symbols for coherent demodulation, feedback regarding the downlink channel is transmitted over an uplink control channel in the form of wideband or subband feedback (subband selection either by base station or by terminal). The feedback consists of channel quality indication (modulation and code rate), rank index (number of MIMO streams) and precoding matrix indication (for MIMO stream selection), depending on transmission mode. Uplink measurement uses a sounding signal, where the bandwidth and antennas are commanded by the base station. Feedback or sounding transmission are thus a description of (part of) the actual space-frequency quantized channel. Time instances for feedback and sounding are commanded by the base station, either as periodic or aperiodic. Signalling overhead is further reduced by the possibility for persistent scheduling.

So on the one hand, flexible scheduling and link adaptation need channel information for all ressources. On the other, reduced feedback for MIMO-OFDMA needs to be quantized, sparse, and irregular (traffic requirements). Collisions of subband and stream selection can occur. To resolve these contradicting requirements as well as the adaptation delay problem, prediction of the quantized channel is necessary. For reduced feedback overhead, this prediction should be performed at the base station. To enable this prediction, also channel correlation information can be fed back (which changes slower than the channel itself) [IOB11]. Prediction (including error prediction) based on non-uniform sampling (feedback) positions can be performed by Wiener filtering and is an approach to avoid mis-adaptation in wireless systems covering a wide range of mobility and location scenarios.

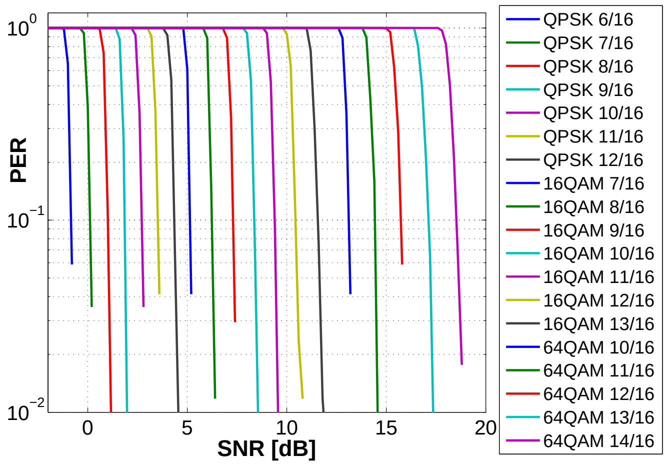


Figure 7.8: Packet error rates for different transmission modes (modulation and code rate), chosen to have $1dB$ distance [IOB11].

7.3.1 Benefit of Channel Correlation Knowledge

The time-variant channel transfer function is denoted as $h(i, j, p)$, with i OFDM symbol index, j subcarrier index and p pair of transmit / receive antennas.

For link adaptation and scheduling, the quantized channel according to granularity of adaptation / scheduling is more interesting: a physical ressource block (PRB) consists of N_{sub} subcarriers over N_t OFDM symbols [3GP10b]. The average squared channel gain (proportional to SNR) of a PRB is:

$$\bar{g}(t, f, p) = \frac{1}{N_{sub}N_t} \sum_{i \in \mathrm{PRB}\ t} \sum_{j \in \mathrm{PRB}\ f} |h(i, j, p)|^2 \tag{7.10}$$

with t PRB time index and f PRB frequency index. The correlation is

$$R_{\bar{g}}(\Delta t, \Delta f, p1, p2) = \frac{E[(\bar{g}(t, f, p_1) - \mu_g)}{\sigma_g} \cdot \cdot \frac{(\bar{g}(t + \Delta t, f + \Delta f, p_2) - \mu_g)]}{\sigma_g} \tag{7.11}$$

where $\bar{g}$ is assumed as stationary random process with mean μ_g and variance σ_g.

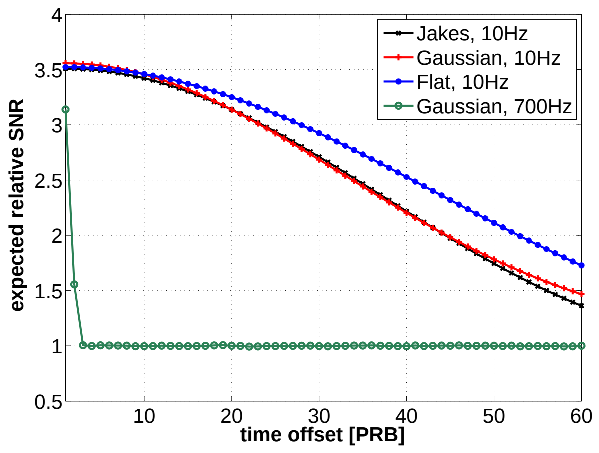


Figure 7.9: 20MHz Urban macro OFDM channel's frequency selection gain in dependence on signalling delay, for different Doppler spectrum shapes and maximum Doppler frequencies [IOB11].

As in [KL08], it is assumed here that the correlation is separable into three factors:

$$R_{\bar{g}}(\Delta t, \Delta f, p1, p2) = r_t(\Delta t) r_f(\Delta f) r_s(p_1, p_2) \tag{7.12}$$

Autocorrelation is the inverse Fourier transform of power spectral density (compare Sec. 1.3).

$$r_t(\Delta t) r_f(\Delta f) = \mathcal{F}^{-1}(S_{del}(\tau) S_{Dop}(\nu)) \tag{7.13}$$

with Doppler power spectrum spectrum $S_{Dop}(\nu)$ and delay power spectrum $S_{del}(\tau)$. Common spectra assumption for the channel h include Gaussian shape, uniform spectrum or Jakes spectrum [Pat02]. Precondition for correlation signalling is of course estimation of the correlation by the terminal – but high-accuracy channel estimation by adaptive filtering requires the receiver to estimate channel correlation anyway [JL07, KID10] (Sec. 3.1).

To predict the expected value μ_p of $\bar{g}$ on a position $(t, f, p1, p2)$ from a vector $\mathbf{f}$ of (possibly non-uniform) feedback samples, the Wiener filter [Wie49] is used:

$$\hat{\mu_p} = (\mathbf{R}_{\mathcal{A}}^{-1} \mathbf{r}_{\mathcal{A}})^T (\mathbf{f} - \mu_g \mathbf{1}) + \mu_g \tag{7.14}$$

where $\mathbf{R}_{\mathcal{A}}$ is the correlation matrix of the feedback samples (within filter area $\mathcal{A}$), $\mathbf{r}_{\mathcal{A}}$ is the correlation vector of these sample positions and the position to predict, and $\mathbf{1}$ is a vector of adequate size containing ones. The process mean μ_g is subtracted from the feedback and added later to the prediction. The corresponding prediction error has the variance [HKR97]:

$$\hat{\sigma_p^2} = \sigma_g^2(1 - \mathbf{r}_{\mathcal{A}}^T \mathbf{R}_{\mathcal{A}}^{-1} \mathbf{r}_{\mathcal{A}}) \tag{7.15}$$

The prediction is quite similar to multidimensional Wiener interpolation filtering for pilot symbol assisted channel estimation as in [HKR97] (where in this case there is a non-zero process mean and non-uniform sampling). For single feedback, $\mathbf{R}_{\mathcal{A}} = 1$ and $\hat{\mu_p}$ decreases with the time-direction autocorrelation $\mathbf{r}_t(\Delta t)$. With growing time lag between feedback positions and PRB to predict, the mean prediction converges to the process mean, and the error prediction to the process variance [BD87]:

$$\lim_{\Delta t \to \infty} \hat{\mu}_p = \mu_g \,, \qquad \lim_{\Delta t \to \infty} \hat{\sigma}_p = \sigma_g \tag{7.16}$$

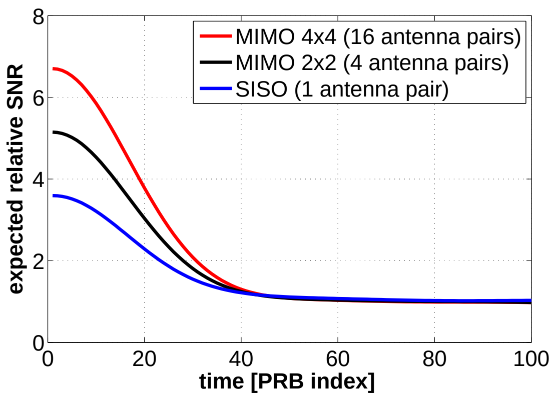


Figure 7.10: MIMO-OFDMA space-frequency selection gain over delay for uncorrelated channels (20Hz max. Doppler, Gauss spectrum, 20MHz urban macro) [IOB11].

Time-direction autocorrelation

Time-direction autocorrelation of $\overline{g}$ as described by power Doppler spectrum can be approximated using a parametric model with few parameters, in the first approximation (one parameter) e.g. terminal speed, maximum Doppler shift or channel coherence time (correlation dropped to 50%). This is illustrated with three common one-parametric models. Using the Jakes model, Doppler spectrum and correlation are [Pat02]:

$$S(f) = \frac{1}{\pi f_d \sqrt{1 - (f/f_d)^2}}, \; |f| \le f_d; \; r_t(\Delta t) = J_0(2\pi f_d \Delta t) \tag{7.17}$$

with f_d for the maximum Doppler shift.

The Gaussian spectrum assumption is:

$$S(f) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{f^2}{2\sigma^2}}; \quad r_t(\Delta t) = e^{-\pi\sigma\sqrt{2}\cdot(\Delta t)^2} \tag{7.18}$$

where f_d is defined as $f_d = \sigma\sqrt{2}$, and σ is the standard deviation.

For the uniform model it is:

$$S(f) = \frac{1}{2f_d}, \; |f| \le f_d; \qquad r_t(\Delta t) = sinc(2f_d\Delta t) \tag{7.19}$$

For single feedback and time-direction prediction, the expected adaptation gain in dependence on feedback delay is illustrated in Fig. 7.9 for different spectra of h. Simulation uses the urban macro channel model [3GP06], a PRB spans $0.5ms$ times $180kHz$ (as in LTE). The expected adaptation gain by selecting the best PRB in frequency direction (compared to channel average) is 3.5, for zero delay. With increasing delay (round-trip and protocol delay) the figure shows the descent to channel average. The three example models (Jakes, Gaussian, uniform) show little difference for the same max. Doppler shift (in the figure $10Hz$, corresponding to low mobility, 5 km/h at 2GHz). The figure also shows the expected adaptation gain over gain for $700Hz$ (medium/high mobility, 60 km/h at 2GHz), which reduces the coherence time reduced from around 40 PRBs to less then 1 (anti-proportional to f_D).

Fig. 7.10 illustrates the expected adaptation gain over delay if the best PRB (urban macro channel) is selected not only in frequency direction, but also over different (uncorrelated) MIMO streams ($20Hz$ f_D, Gaussian spectrum). While the expected adaptation gain increases with the number of uncorrelated antenna pairs, the behaviour over time is not influenced — the larger relative decrease poses a higher risk for misadaptation.

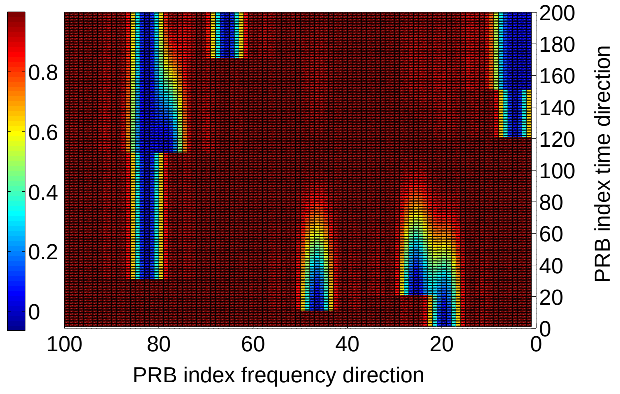


Figure 7.11: Prediction error variance for periodic feedback about best PRB in frequency direction (every 10 PRB) and example channel realisation [IOB11].

In the following, packet error rate and throughput are determined in dependence on signalling (adaptation) lag. Fig. 7.8 illustrates packet error rates for different transmission modes (of one spatial stream; modulation and code rate are varied), for a reference receiver algorithm. With current rate matching methods (e.g. [CNB+08]), the code rate can be chosen with very fine granularity. With strong coding (LTE turbo code in the example), the packet error rates show an 'on-off' behaviour in dependence on SNR: at a threshold SNR the PER drops very fast. The transmission modes in Fig. 7.8 are chosen to have a spacing of approximately 1dB SNR.

In addition to the expected mean adaptation gain as in Fig. 7.9, Fig. 7.12 also shows the expected variance in dependence on delay, i.e. the expected prediction of mean value and prediction error according to Eq. (7.14) and Eq. (7.15) (for urban macro channel parameters and a Doppler spectrum assumption). The expected channel gain for a certain delay Δt is described by a Gaussian distribution $\mathcal{N}(\mu(\Delta t), \sigma(\Delta t))$. The figure also contains SNR thresholds T_m for different transmission modes m (1dB distances). With this, the packet error rate in dependence on delay can be determined as:

$$\begin{aligned} \mathrm{PER}(T_m, \Delta t) &= \int_{-\infty}^{T_m} \mathcal{N}(\mu(\Delta t), \sigma(\Delta t)) \\ &= \Phi(\frac{T_m - \mu(\Delta t)}{\sigma(\Delta t)}) \end{aligned} \tag{7.20}$$

with the tail distribution Φ. The resulting packet error rate for different transmission modes is depicted in Fig. 7.13 in dependence on delay.

A transmission mode m maps B_m bits into a PRB, so the expected relative throughput compared to mode m_1) is:

$$\mathrm{TP}(m, \Delta t) = \frac{B_m \cdot \mathrm{PER}(T, \Delta t)}{B_{m_1}} \tag{7.21}$$

Resulting throughput for different modes and delays is illustrated in Fig. 7.14.

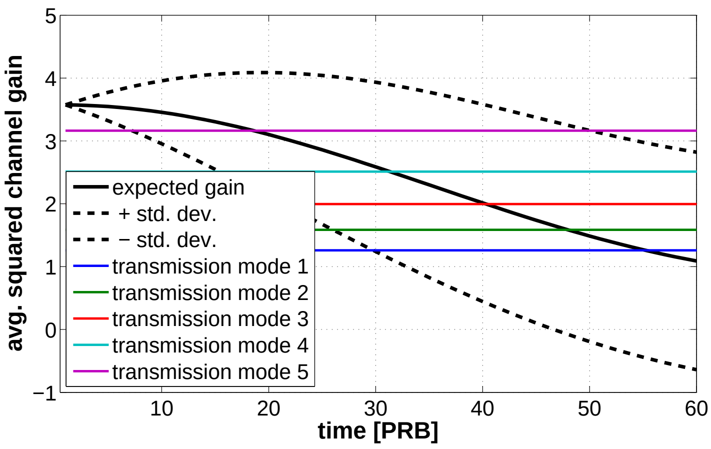


Figure 7.12: Mean and standard deviation of adaptation gain over delay, also transmission mode SNR thresholds (1dB spacing) [IOB11].

Terminals feed back current parameters of the Doppler spectrum model to the base station, so that the base station can choose the transmission mode to optimize expected throughput, considering the control lag. To illustrate the benefit of time-direction correlation knowledge, link adaptation for periodic channel quality feedback with and without correlation knowledge are compared. Following a (short-term) static channel assumption (channel correlation not known), the expected average throughput in dependence on delay t_{lag} and feedback period t_{per} is:

$$\bar{\mathrm{TP}}_{\mathrm{static}}(m, t_{lag}, t_{per}) = \frac{1}{t_{per}} \sum_{\Delta t = t_{lag}}^{t_{lag}+t_{per}} \mathrm{TP}(m, \Delta t) \tag{7.22}$$

Fig. 7.15 shows an illustration.

With known correlation and the same feedback scheme, the average expected throughput becomes:

$$\bar{\mathrm{TP}}_{\mathrm{corr}(t_{lag},t_{per})} = \frac{1}{t_{per}} \sum_{\Delta t = t_{lag}}^{t_{lag}+t_{per}} \underset{m}{max}\, \mathrm{TP}(m, \Delta t) \tag{7.23}$$

The expected best transmission mode can be chosen in dependence on delay, which corresponds to the envelope in Fig. 7.14.

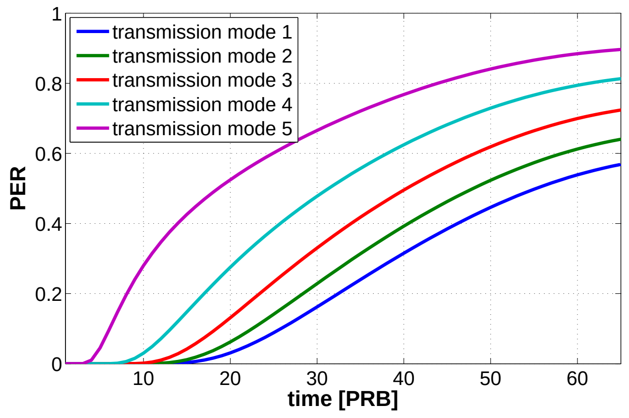


Figure 7.13: Packet error rates over delay for the transmission modes from Fig. 7.12 [IOB11].

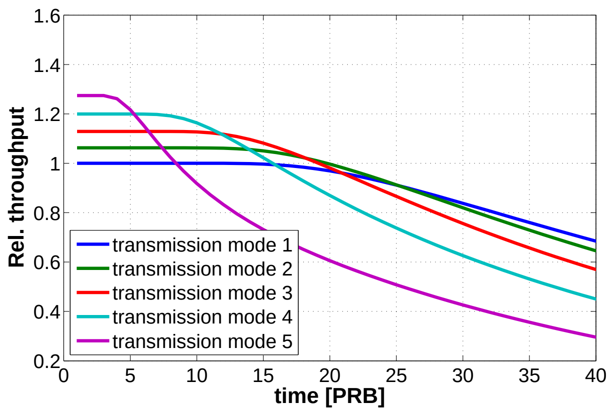


Figure 7.14: Throughput over delay for the transmission modes from Fig. 7.12 [IOB11].

Correlation in frequency direction and over antennas

There are several reasons for the benefit of knowledge about the channel's 3D correlation. First, more correlation between feedback samples improves prediction accuracy (compare Eq. (7.15)). Second, with a growing number of users, the probability of collisions of user-selected subband feedback increases – which necessitates prediction for positions also on other frequencies and/or streams. In addition, spatial (antenna) correlation can be used to select the number of MIMO streams and/or MIMO mode. For 3D Wiener prediction filtering, the equations (7.12), (7.14), (7.15) remain unchanged – only the filter area increases in dimensionality.

Fig. 7.11 illustrates the prediction error for 2D prediction (time-frequency, OFDM channel) for periodic feedback with $t_{per} = 10$. Other parameters for this illustration are Gaussian Doppler spectrum with $f_D = 10Hz$, uniform delay spectrum with $\tau_{max} = 0.5\mu s$, and zero signalling delay. For feedback positions the prediction error is zero, and with increasing distance from feedback positions, the prediction error increases towards process variance.

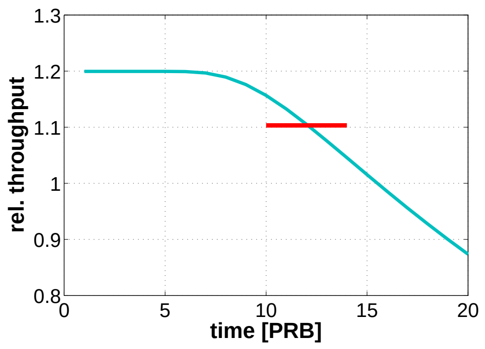


Figure 7.15: Average expected channel throughput for transmission mode 4 from Fig. 7.12 for 10 PRB adaptation delay and 4 PRB feedback period [IOB11].

7.3.2 Signalling and Tracking

It is now assumed that the terminal performs measurement of the short-time 3D correlation of $\overline{g}$ over a window in time direction (on the order of coherence time), frequency and angular direction (or equivalently between antenna pairs) e.g. according to [JL07, KID10].

To include feedback signalling into throughput optimization, the throughput has to be reduced by signalling overhead. If asymmetric power and energy constraints are neglected in an FDD system, then it is reasonable to reduce downlink throughput by the corresponding feedback overhead on the uplink (since the frequency split in downlink/uplink could in principle be adjusted). Alternatively, different weights can be given for uplink versus downlink bits.

Feedback reporting parameters

While the cost of feedback signalling is clear, the benefit is the expected throughput increase with the additional feedback (expected throughput conditioned on feedback positions). For single feedback (request and transmission) this can be determined by Eq. (7.14) and (7.21), for periodic feedback directly from Eq. (7.23). The expected adaptation gain depends of course on the multi-variate probability density function of the channel (compare Fig. 7.9 and Fig. 7.10).

Assuming a constant bit rate source and periodic feedback, the feedback period could be optimized by:

$$t_{per}^{(opt)} = \underset{t_{per}}{\operatorname{argmax}} \left(\overline{\mathrm{TP}}_{corr}(t_{per}) - \mathrm{OV}(t_{per})\right) \tag{7.24}$$

where the feedback overhead $\mathrm{OV}(t_{per})$ is a hyperbola. A more detailed model may include feedback collision probability and expected throughput for alternative allocations.

Time-variant channel autocorrelation

Channel correlation changes slower than the channel itself (by definition and window-based estimation), so separate signalling for channel correlation and channel quality feedback reporting is adequate. While time-direction correlation r_t is velocity dependent, frequency- and space-direction correlation are location dependent and could possibly be stored in the base station to avoid signalling.

The maximum Doppler shift in dependence on terminal velocity is

$$f_d = f_c \cdot \frac{v}{c} \tag{7.25}$$

with f_c carrier frequency, v velocity, and c speed of light. Time-direction correlation change due to acceleration $\frac{dv}{dt}$ (assuming same spectrum shape) with acceleration values achieved by cars for example means a slow correlation change which can be tracked by low-rate correlation change signalling.

A different cause of changing correlation can be variation of the spectrum shape due to alterations of reflectors and scatterers in the channel (e.g. passing cars, trains).

Different ways of correlation change signalling are discussed in the following and illustratied in Fig. 7.16. The urban macro channel and round-trip delay of 10 PRB (5ms) is assumed, further a Gaussian Doppler spectrum and (non-continuous) change of f_D according to a step function from 10Hz to 200Hz.

Other parameters are periodic sounding (3 PRB period) and correlation measurement delay (3 PRB for 200Hz). The blue curve in Fig. 7.16 shows the expected throughput over time for no correlation signalling and (short-time) static channel assumption. The red curve shows the expected throughput for periodic correlation reporting with 10 PRB period. As already seen in Sec. 7.3.1, correlation signalling improves the steady-state throughput, which for higher f_D is smaller. With periodic correlation reporting, the transmission is adapted to the changed correlation after measurement delay, average reporting delay (average offset of reporting time to correlation change time) and round-trip delay.

Terminal-decided aperiodic update of correlation information

To minimize both correlation reporting delay and overhead, it is desirable to update the transmitted correlation only when a (significant) change has occurred: this means terminal-decided correlation feedback reporting. Terminal-decided aperiodic feedback transmission requires that an allocated uplink ressource is available (data or control channel). Implementation can be transmission of correlation info instead of scheduled data, or possibly piggy-back to channel quality reporting. Terminal-decided feedback offers an overhead reduction compared to high feedback periodicity, and packet error rate reduction compared to updating with low periodicity. The black curve in Fig. 7.16 illustrates expected throughput for terminal-decided correlation feedback, which compared to periodic correlation feedback avoids the average reporting delay.

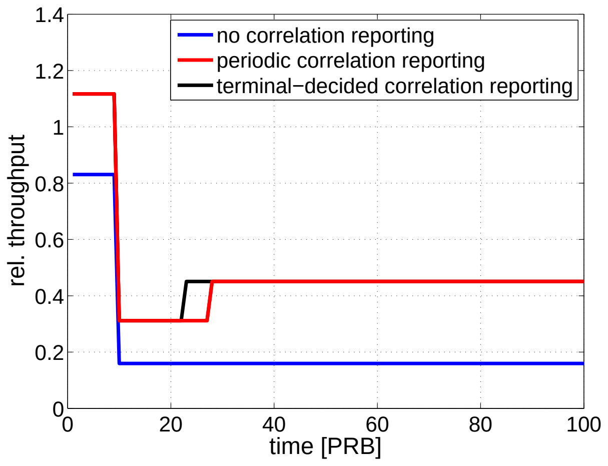


Figure 7.16: Expected throughput over time for step function correlation change (after 10 PRB) and different correlation signalling schemes [IOB11].

Notes Jointly considering downlink and uplink, there is an ’information discrepency’ between the base station and a terminal: while each terminal can continuously track the downlink channel, the base station can estimate an uplink channel only on resources where the terminal transmits (data, control or sounding). So the discussed method to obtain correlation information was measurement by the terminal and feedback signalling. The channel correlation parameters of downlink and uplink are themselves correlated, because changes in channel autocorrelation are caused by the same physical reasons (acceleration, location). This means both that a terminal can also continuously estimate correlation of the uplink channel, and that the base station could infer downlink correlation parameters from uplink sounding (if the sounding signal is transmitted with low period). Conversion of 2D (spatio-temporal) correlation from uplink to downlink is discussed in [JGA09]. Such estimation of correlation information avoids signalling the correlation, although at the expense of a conversion error (reduced prediction accuracy).

7.4 Adapting Receiver Algorithm to Scheduling Parameters

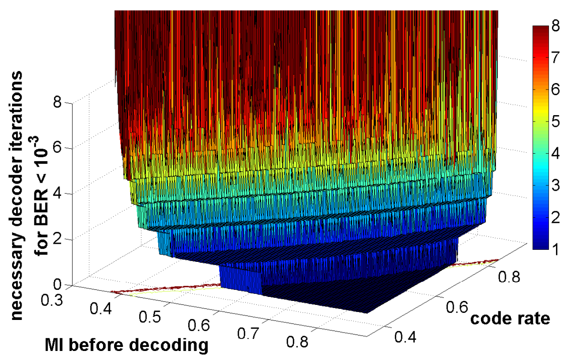
This section considers joint optimization of adaptive transmit and receive processing for sub-optimal (non-ML) receivers, which are constrained in computational power. It demonstrates potential gain of non-equal allocation of receiver computational power to parallel messages (referring to chapter 6 for receive processing, this section deals with optimization at run-time rather than at design-time). The optimal transmission parameters depend on the receiver's computational power constraint and its available algorithms. Specifically the (for block-ML reception capacity-achieving) waterfilling transmit power allocation normally is not optimal in this context.

Optimization of transmission parameters aims at achieving channel capacity, most often with the implicit assumption of an optimal (block maximum likelihood) decoder. For different channel models, capacity-achieving transmit strategies have been found. The optimal power allocation for adaptive transmission over a spectrally shaped channel with channel state information at the transmitter (CSIT) is known as 'waterfilling' [CCB95, Gal68]. Corresponding bitloading algorithms for adaptive modulation and finite transmission block length are based on a gap model, which describes the distance in signal to noise ratio (SNR) from the (infinite block-length) capacity in dependence on bit error rate (BER) [KRJ00, CCB95]. Bitloading according to the waterfilling solution has been applied to OFDM transmission, e.g. for digital subscriber line (DSL, [KRJ00]) and wireless communications [Czy96]. Similarly for MIMO transmission, the capacity achieving transmit parameters are yielded by singular value decomposition and waterfilling power allocation over the resulting parallel channels [Tel99]. Practical link adaptation schemes with fine granularity adapt not only modulation, but also code rate (for OFDM described in e.g. [MH02a]).

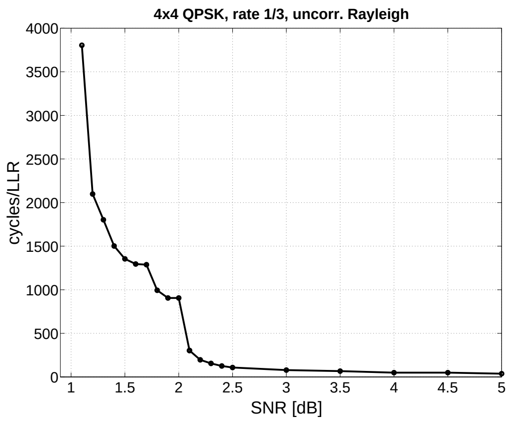
Optimization of receiver processing assumes fixed transmission parameters. In chapter 3 is was seen that for each receiver function, many algorithm alternatives exist, in chapter 6 the receiver's Pareto efficient accuracy/complexity tradeoff was characterized: while the block ML receiver ranges at almost infinite complexity for practically used packet length, combinations of (iterative) algorithms can approximate its accuracy with variable computational effort. For given transmission parameters, the optimal receiver depends on the tolerated complexity.

Several information theoretic models have been developed to characterize channel capacity for non-block-ML receivers, notably the α-decoder [CK81], β-decoder [CK80] and d-decoder [CN95]. These models consider different suboptimal decoding criteria, but as information theoretic models they do not consider computational effort.

In this section, joint optimization of transmit and receive parameters under receiver complexity constraint is considered. Specifically, it is dealt with the following questions: Does the optimial transmit strategy depend on the receiver computation constraint (i.e. , does it differ from the waterfilling solution)? Is non-equal computation allocation by the receiver beneficial? For a given channel model, key to answer is the (Pareto-optimal [Par]) relation between transmit power, achievable rate and receiver complexity. It is shown that if modelled with parameters from available algorithms and an implementation complexity metric, it is possible to quantify the gain obtainable by adaptive computational allocation.



(a) Necessary number of Turbo decoder iterations grows hyperbolically towards high code rate and low MI before decoding.



(b) Computing effort of Pareto-efficient algorithm selection for iterative MIMO demapping-decoding shows steep increase towards singularity at low SNR [IB11].

Figure 7.17: Experimentally obtained performance-complexity-rate tradeoffs.

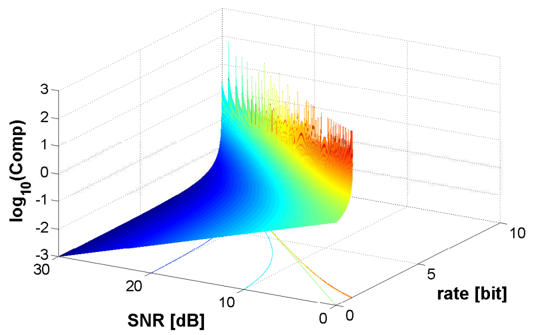
7.4.1 Motivation: Experimental Example Relations of Computation, Rate and SNR

Turbo decoding

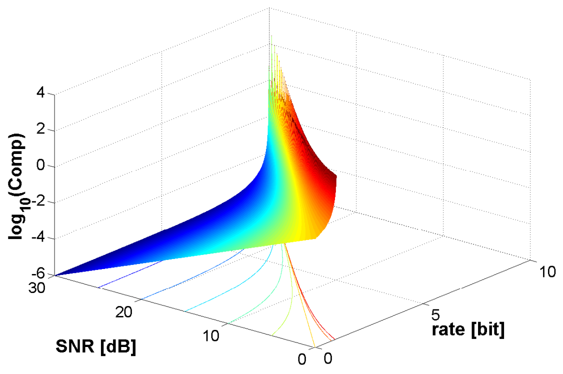
In Turbo decoding, computational effort is variable by the number of decoding iterations – typically 3-11 iterations are used. The code rate can be adjusted by the transmitter in a fine-grained way using rate-matching (puncturing) methods (e.g. [CNB+08]). Fig. 7.17a illustrates the trade-off between receiver computation, transmission rate and channel SNR for the LTE Turbo decoder. Mutual Information (MI) between bit log-likelihood ratios (LLRs) before decoding (decoder input) and transmit bits is used as measure instead of channel SNR, because it is applicable for all modem parameters. The surface shown in the figure is the necessary number of iterations to achieve $BER < 10^{-3}$. The surface is limited by singularities towards high code rate and low MI before decoding.

Iterative MIMO demapping-decoding

Complexity increase towards a singularity at very low SNR can also be observed in results from iterative demapping-decoding, like e.g. soft-input soft-output sphere decoding combined with Max-Log BCJR convolutional decoding in [SB10]. Here it is referred to the automatic algorithm optimization method from chapter 6. Fig. 7.17b reproduces computational effort and channel SNR of the Pareto-optimal receiver algorithms for 4x4 MIMO transmission with rate 1/3 Turbo code over a channel with uncorrelated Rayleigh distributed coefficients.



(a) Example model 1 for necessary computing effort in dependence on rate and SNR. For infinite computation limited by Shannon capacity.



(b) Example model 2, steeper ascent for low SNR.

Figure 7.18: Example Rate-SNR-Computation tradeoffs.

7.4.2 Theoretical Explanation of Potential Gain

General model formulation

For the general model of a rate-SNR-computation relation, the continuous relaxation (continuous interpolation of discrete sets) is considered. The rate is limited by channel capacity and can be achieved for infinite receiver computation. The necessary computation C can be written as function in dependence on SNR γ and rate R (here the Shannon capacity for the AWGN channel is assumed):

$$C = \check{f}(R, \gamma) \tag{7.26}$$

$$\text{for } R < \log_2(1+\gamma); \quad C, R, \gamma \geq 0$$

For this function, the following properties are assumed (which seem plausible based on the experimental observations of the previous section):

1. singularities at channel capacity:
$$\lim_{R \nearrow \log_2(1+\gamma)} C = \infty \tag{7.27}$$
$$\lim_{\gamma \searrow 2^R - 1} C = \infty \tag{7.28}$$
2. asymptotes:
$$\lim_{\gamma \to \infty} C = C_{\min} \quad \text{and} \quad \lim_{R \to 0} C = 0\ , \tag{7.29}$$
where $C_{\min}$ is a minimum complexity which may be needed for channel equalization.
3. strict monotonicity:
$$\frac{\delta C}{\delta \gamma} < 0 \quad \text{and} \quad \frac{\delta C}{\delta R} > 0 \tag{7.30}$$
4. second derivatives:
$$\frac{\delta^2 C}{\delta^2 \gamma} < 0 \quad \text{and} \quad \frac{\delta^2 C}{\delta^2 R} > 0 \tag{7.31}$$

The inverse function for rate R in dependence on SNR γ and computation C is denoted (it exists due to strict monotonicity of f):

$$R = f(C, \gamma), \tag{7.32}$$

and the inverse function for SNR γ in dependence on rate R and computation C:

$$\gamma = \widetilde{f}(R, C) \tag{7.33}$$

A parameter fit of a model to interpolate experimental data is possible.

Example model

For the following discussion, a concrete model is assumed:

$$C(\gamma, R) \quad = \quad (1 + \gamma - 2^R)^{1/\alpha}; \qquad -\infty < \alpha < 0 \tag{7.34}$$

with rate:

$$\begin{aligned} R(C,\gamma) &= \max(0; \log_2(1+\gamma - C^{\alpha})) \\ &= \max(0; \log_2(1+\frac{Pg}{N} - C^{\alpha})) \end{aligned} \tag{7.35}$$

with signal power P, squared channel gain g and noise power N. The properties from Sec. 7.4.2 are met when assuming $C_{\text{min}} = 0$ for simplicity. The function is illustrated for $\alpha = -1$ as example model 1 in Fig 7.18a, and for $\alpha = -0.5$ as example model 2 in Fig. 7.18b.

Distributing computation over parallel channels / messages

Transmission over parallel channels $1 \le i \le N$ with different squared channel gains g_i is considered, where the problem is power allocation (bit loading) with sum power constraint and receiver computation allocation with a sum computation constraint. The following schemes are compared:

1. transmitter uses equal power allocation, receiver uses equal computation allocation.
2. transmitter uses equal power allocation, receiver optimizes computation allocation.
3. transmitter uses waterfilling power allocation (needs channel state information at the transmitter, CSIT), receiver uses constant computation allocation (if subchannels remain unused, the available computing power is not completely used).
4. transmitter uses waterfilling power allocation, receiver uses equal computation allocation for the used channels.
5. transmitter uses waterfilling power allocation, receiver optimizes computation allocation.
6. joint optimization of transmit and receive parameters (needs CSIT and knowledge of the receiver accuracy/complexity/rate tradeoff at the transmitter).

In the following, differences of the results of these schemes are compared based on properties of the computation model. The target function to maximize is chosen to be the sum rate. So in the standard form, the negative sum rate is minimized:

$$f_0 = -\sum_i R_i(C_i, P_i) \overset{!}{=} \min \tag{7.36}$$

The sum power constraint, sum computational constraint, and non-negativity constraints are:

$$\begin{aligned} h_1(P_1,\dots,P_N) &= P_{max} - \sum_{i=1}^{N} P_i = 0 \\ h_2(C_1,\dots,C_N) &= C_{max} - \sum_{i=1}^{N} C_i = 0 \\ g_k(P_k) &= P_k \le 0\ ; \qquad 1 \le k \le N \\ g_l(C_{l-N}) &= C_{l-N} \le 0;\ N+1 \le l \le 2N \end{aligned} \tag{7.37}$$

This nonlinear constrained optimization problem is a convex optimization problem [BV04]. Convexity of f_0 can be shown by showing that the Hessian matrix is positive semi-definite using Sylvester's criterion.

Necessary and sufficient for the global solution point are the Karush-Kuhn-Tucker (KKT) conditions [BV04]:

$$\begin{aligned} \nabla f_0 + \sum_{i=1}^{2N} \lambda_i \cdot \nabla g_i + \sum_{i=1}^{2} \nu_i \nabla h_i &= 0 \\ \lambda_i g_i &= 0, \quad i = 1,\dots,2N \\ \lambda_i &\ge 0, \quad i = 1,\dots,2N \end{aligned} \tag{7.38}$$

named stationarity, dual feasibility and complementary slackness. The value function describes the gain in f_0 (sum rate) by increasing the power or computation constraint:

$$V(P_{max}, C_{max}) = \sup_{P_1,\dots,P_N,C_1,\dots,C_N} f_0 \tag{7.39}$$

For the limit of infinite computing power, the problem becomes the normal waterfilling problem with the analytic solution [BV04]:

$$\frac{P_i}{P_{max}} = \begin{cases} 0 & ;\ \nu_1 \ge \frac{g_i}{N} \\ \frac{1}{\nu_1} - \frac{N}{g_i} & ;\ \nu_1 < \frac{g_i}{N} \end{cases}$$

where the fill level $\frac{1}{\nu_1}$ is determined by the waterfilling algorithm.

Numerical illustration

Here the transmission schemes 1) to 6) are simulatively compared using the two example models in the following scenario: transmission over 100 parallel channels with independent Rayleigh-distributed squared gains ($\sigma = 1$), $P_{max} = 150$, $C_{max} = 50$ and $N = 1$. The optimization problems are solved using the active set algorithm with sequential quadratic programming. Resulting average sum rates for both models are shown in Fig. 7.19. Referring to the introductory questions, the following can be seen:

– pooling computation allocation to completely use the available computing power is beneficial (schemes 3-4).
– optimizing receiver computation allocation is beneficial (compare schemes 1-2).
– joint optimization performs best and shows a considerable potential gain in this scenario (scheme 6).
– the waterfilling transmit solution with receiver optimization afterwards performs worse than joint optimization – illustrating that the optimization problem is not separable (schemes 5-6).
– adaptive receiver computation allocation has a higher potential benefit when the transmission is not optimized (model 1, schemes 1-2 versus schemes 4-5).
– the waterfilling transmit solution can perform worse than equal transmit power allocation (model 2, schemes 2-5).

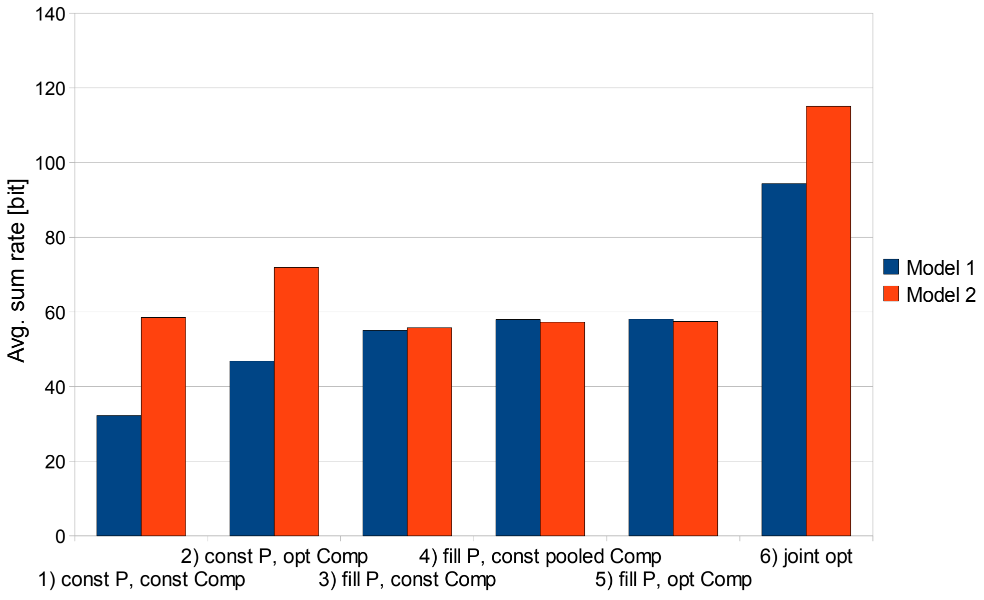


Figure 7.19: Expected sum rates in example scenario for different transmission/reception schemes and two example models of rate - SNR - computing power tradeoff.

7.4.3 Exploiting the Gain in Practical Scenarios

In modern cellular systems there is channel state information at the receiver (CSIR, provided by pilot symbol assisted channel estimation) and quantized channel state information at the transmitter (provided over a control feedback link). Now there is the need to additionally differentiate whether there is knowledge about the receiver's rate-SNR-complexity tradeoff at the receiver (we denote it as RSCR), or at the transmitter (RSCT). Other scenario differentiations are uplink versus downlink, reception from a single or multiple users or MIMO per-stream encoding.

Measuring and standardizing a practical receiver's rate-SNR-complexity tradeoff

Possible transmission rates are given as discrete set of transmission parameters (modulation, code rate etc.). For each choice of transmission mode and different SNR levels, the receiver is measured or simulated for all possible variations of its receive parameters (decoder iterations etc.). For each point in this discrete parameter space, the receiver's computational effort is stored (as percentage of its available computational power) if decoding is successful (defined as $BER < 10^{-3}$). The rate-SNR-complexity tradeoff can be stored in the receiver (RSCR) — as table look-up or as function parameters.

A selection of such rate-SNR-complexity tradeoffs can be standardized as 'receiver categories'. A receiver belongs to a category if it has more computational power (needs less percentage) in the complete parameter space than the reference one. To obtain RSCT, a terminal receiver's category is transmitted to the base station at system entry.

No RSCR and no RSCT

Normally a reference receiver algorithm with fixed receiver processing is assumed, no complexity tradeoff. The set of transmission parameters is chosen to have equal SNR spacing for this equal receiver processing. Fig. 7.20a illustrates such a a set of transmission modes with different modulation and code rate, having 1.5dB SNR spacing for unbiased MMSE equalization and 8 Turbo decoder iterations (4x4 MIMO, uncorr. Rayleigh).

Receiver, detection of early decoding success and computation pooling In modern receivers, at least the number of decoder iterations is variable. In the case of iterative demapping and/or iterative channel estimation, run-time optimization of the receiver component computation schedule e.g. according to [EMK06, ZLNA10] can also be used, possibly also switching of component algorithms. Without explicit knowledge of the tradeoff, the receiver can use computation pooling for parallel messages. One example is early detection of decoding success, which is normally possible because the transmitter adds a checksum before encoding – so the receiver can detect and drop erroneous packets after failed decoding. Decoding can be stopped as soon as the checksum fits after a forward error correction decoding iteration. Another example is transmission only on a frequency subband in OFDMA. Since using only a subband means transmitting less bits, the receiver has more time per bit for decoding and can run more iterations. So after successful decoding of one message, any left-over computing power (equivalently decoding time) can be used for other messages. This is applicable in multi-user uplink as well as in downlink with single-user MIMO and per-stream encoding.

RSCR, but no RSCT With RSCR, the receiver can optimize its processing according to the given transmission parameters. In the numerical illustration from Sec. 7.4.2 this corresponds to the gain from 2) in comparison to 1).

Uplink, receiver computation allocation One scenario is uplink reception at the base station from several terminals (different transmitters for parallel channels).

Downlink, single-user MIMO with per-stream encoding Another scenario is downlink reception of several MIMO streams at a terminal (same base station transmits several messages to this terminal over different channels).

RSCT, but no RSCR

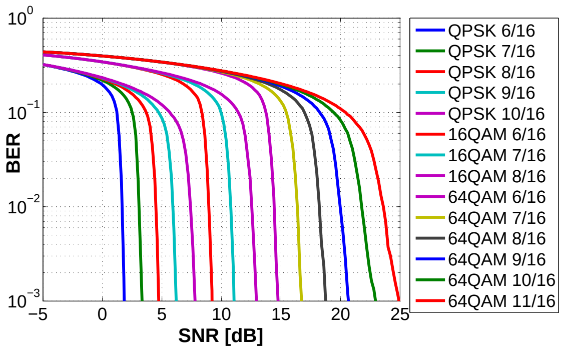
Transmitter, choose transmission parameter quantization for equal computation Here flexible receivers are considered, which have the same maximum computational power as the reference one with fixed processing. Compare again Fig. 7.20a: as the code-rate is varied, the actual receiver computational effort (measured per information bit) is not equal: for the shown rates between 6/16 and 11/16, Turbo decoding effort with same number of iterations changes with almost a factor of 2. Accordingly for the same transmit parameters and receive processing with equal computational effort (e.g. variable decoding iterations), the SNR spacings would be different. Fig. 7.20b illustrates variation of BER threshold of one of the transmission modes with receiver computational effort (also compare Fig. 7.17b). Transmit adaptation assumes equal distribution of computational power at the receiver — in the numerical example from Sec. 7.4.2 this corresponds to 3). The set of transmission modes should be chosen to have equal SNR distance and constant receiver computing effort in order to completely use the available sum computing power.

RSCR and RSCT

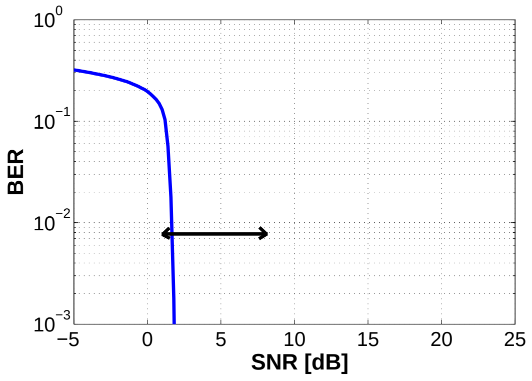
RSCR and RSCT together enable joint optimization corresponding to 6) in the numerical example (Sec. 7.4.2). Scenarios are

1. Uplink scheduling, transmit and receive processing jointly optimized by the base station's scheduler.
2. Accordingly in downlink, for single-user MIMO with per-stream encoding.

3. Downlink, receivers with different computational power. Optimal link adaptation (and scheduling) is different for receivers with small or high computational power.



(a) Example set of transmission modes (modulation and code rate) with equal receiver processing (non-equal computing effort) and SNR spacing of 1.5dB.



(b) BER threshold of transmission mode varies with receiver computing effort (also compare Fig. 7.17b).

Figure 7.20: Receiver computation as variable.

Some notes Concrete results depend of course on the actual scenario as well as the rate-SNR-computing trade-off (computational model). As in the case of adaptive power allocation only the possible gain (as difference between equal and adaptive distribution) tends to zero with growing available power, here the possible gain tends to zero with growing computational power. The target function for multi-user communication may be changed from sum rate maximization to optimization of a utility function considering fairness between users (Sec. 7.1). If link adaptation is done separately from (after) scheduling, it becomes similar to scheme 2) of the numerical illustration: after power control and scheduling, the SNRs are fixed – only determination of rate and computation allocation remain. Extra gain can be achieved for not fully used system bandwidth. In this case, adaptive computation allocation compared to fixed receiver processing can provide more flexible link adaptation by partly circumventing rate limitation due to tranmit power constraints of the individual transmitters. The potential gain is a motivation for joint scheduling, link adaptation and computation allocation – exploiting multiuser diversity (flexibly mapping transmitters to channels) for joint optimization gain.

7.5 Protocol Extensions for Multihop Relaying

Multihop relay nodes provide a means to quickly achieve the needed coverage of new systems without the high costs of fixed network connection of each access node. Another benefit of relaying is capacity improvement through higher SNRs. Multihop relaying means routing on the MAC layer. Sec.

7.5.1 considers necessary changes of frame structure and related signalling with the introduction of multihop relays. To limit the increase of signalling overhead Sec. 7.5.2 describes aggregation of signalling to minimize relaying control information. This entails hierarchical scheduling. Signalling for hierarchical fair scheduling is discussed in Sec. 7.5.3. Sec. 7.5.4 outlines benefits of traffic (de)multiplexing within the radio protocol stack, in case multiple network interfaces are available in a relay.

7.5.1 Frame Structure Extensions

Cellular systems operate in frequency division duplex (FDD) or time division duplex (TDD). Compared to FDD, TDD needs transmit/receive turnaround gaps — timing advance is not possible in TDD. The overhead due to these gaps becomes less important with smaller cell size (smaller round-trip delays). For relays, TDD has the advantage of cheaper radio frequency frontends. Relaying needs the inclusion of a TDD component in the frame structure, also for an FDD system. A side effect for signal processing is the impact of transmission times (non-reception) on channel tracking. Synchronization becomes hierarchical: in a two-hop scenario, the relay synchronizes to the base station, the terminal to the relay. An FDD relay must be able to receive and transmit in both frequency bands, differently from base stations and terminals: it behaves in one direction like a base station and in the other like a terminal. Switching these two relay modes (time duplexing) cannot be transparent to basestation and terminals (and other multihop nodes). It must either be predefined in the standard, or dynamically signalled, which needs inclusion in the protocol. One example protocol function which would need adaptation is synchronous HARQ retransmission [3GP09], where HARQ process numbers are derived implicitly from timing information. The possibility for signalling relay mode switching is described in [HHIZ08].

7.5.2 Aggregated Control Signalling

To reduce signalling overhead with multihop relays, signalling can be aggregated in the multihop nodes: terminal-relay connections are hidden within one relay - base station connection- If base station and relays are not to transmit on the same ressources, additional signalling for coordination of ressource usage becomes necessary [IZHH08]. Regarding the uplinks, there is no problem. The multihop relay can request uplink resources for both itself (transmission to base station) and the connected terminals (transmission to relay). For the downlink, there is a problem regarding resource allocations for transmissions from base station to relay versus resource allocations for transmissions from relay to terminals. To avoid that base station and relay transmit on the same downlink ressources, new messages for requests, grants and CQI feedback between relay and base station for the multihop relay downlink can be introduced

[IZHH08]. It is possible to use a delay, i.e. to grant resources for later subframes to allow for processing delay in the multihop relay. These messages can be used hierarchically, i.e. also for hop counts larger than 2. As normal bandwidth grants, allocations can be valid for several frames.

7.5.3 Distributed Fair Scheduling

If control information is aggregated in a multihop node for the upstream like described in Sec. 7.5.2, this implicitly means hierarchical ressource allocation: each multihop node independently schedules the ressources provided to it. Hierarchical (distributed) scheduling has an impact on the base station's ('global') scheduling criterion – it can no longer compute the utility based on user data rates. A number of user data flows is hidden behind a single relay data flow. A global scheduling criterion can be achieved nevertheless by signalling of local fairness information upwards in the scheduling hierarchy [Ibp07]: the utility computation is decomposed is the same way as the scheduling decision. In the case of max-min fairness, local minimum rates can be propagated upwards. In the case of proportional fairness, the product of local rates or the sum of logarithmic local rates can be propagated. The upstream scheduler may apply linearisation, i.e. assume constant utility per bit from the last scheduling interval [Ibp07].

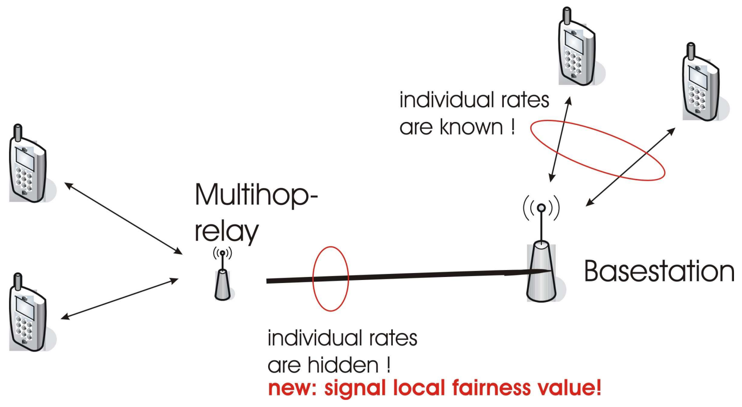


Figure 7.21: Exchanging the fairness value of local scheduling enables hierarchical fair scheduling [Ibp07].

7.5.4 Multi-Layer Routing

Multi-layer routing is an idea for quick network deployment (coverage) with reduced cost [Ibp08]. In the beginning of deployment, user numbers are small, leaving a large fraction of the radio resources unused. At first, a few full basestations and many multihop relays can be deployed, Later, a relay can be equipped with an additional network connection, either a fixed line or a line of sight radio link to the base station (e.g. with a dish for higher frequencies). This corresponds to a partial upgrade from relay to base station, saving relaying overhead over the air. Relay nodes can be upgraded to full base stations by providing adequate lines step by step depending on the capacity requirements of the individual cell. The result is a base station / relay hybrid, with a smooth transition depending on network side date rate. Relay overhead reduction is achieved by splitting the radio protocol stack traffic (data and control information) and partially transmitting it indirectly (not over the air) over IP between relay and base station. This traffic split is a combined MAC/IP routing for load balancing, which differentiates it from ’multi-channel’ routing (which completely works on the IP layer) [Ibp08]. An analogous partial downgrade from base station to relay is also possible, in case of cell bandwidth extension or upgrade to a new standard. In this case, existing sites and fixed line connections are reused, which normally are comparably slow (e.g. DSL) and by themselves would be insufficient for full upgrade. Traffic (de)multiplexing can be adaptive to traffic classes and transport delay-sensitive traffic over the faster connection. Traffic can also partially skip the base station and be transmitted directly between a relay with fixed line and the gateway.

Chapter 8

SDR Testbed

Many algorithms and ideas described in the previous chapters have been implemented and benchmarked in the testbed described in this chapter. In the other direction, parameters described in this chapter have been used for optimization in chapter 6. Sec. 8.2 describes the hardware platform used in the testbed, Sec. 8.3 describes the software platform. Implemented functionality and results are detailed in 8.4. The PCIe form factor demonstrator has been presented at [Hei09], a demonstration using Playstation 3 and USRP has been set up in the HHI lab.

8.1 Design Decisions

The choice of a standard bus enables usage of boards which are produced in high volumes (and are therefore comparatively cheap) and makes the system upgradeable for future needs. The demonstrator consists of off-the-shelf products in the common PCI Express form factor, which provides competitive high performance. A workstation mainboard is used as PCIe backplane, the platform is easily extendable using PCIe plug-in cards.

Comparison to FPGA-centric design A demonstrator platform based on FPGAs (with DSPs for channel estimation and computation of MIMO stream separation matrices) has been presented in [JFH+05]. Here, a processor-centric platform has been chosen because software development as compared to hardware development generally offers a higher productivity and flexibility. The high computational requirements of signal processing can be well met with the Cell processor which is available as PCIe coprocessor board. High-throughput network layer and medium access control layer protocol processing can be done on an IXP network processor, which is also available as PCIe coprocessor board.

Comparison to AdvancedTCA form factor A testbed based on the AdvancedTCA (ATCA) form factor is presented in [JS07]. An ATCA backplane offers more slots and can be used with different fabrics like PCI Express, Ethernet or Infiniband. ATCA has not been chosen here because it would have been considerably more expensive.

8.2 Hardware Platform

A block diagram of the main components is shown in Fig. 8.2. A workstation mainboard [Asu] is used as PCIe backplane. PCIe 2.0 is a (switched) serial bus with 2.5GBit/s per lane, offering a raw data rate of 2GBit/s bidirectional per lane [BAS04]. Lanes can be aggregated for higher throughput.
The four plug-in cards used are:

- Cell accelerator board [Mer]
- Network processor board (IXP2350) [IP 05]
- FPGA board (Virtex5) [Xil]
- (optionally) 10GBit Ethernet network interface card [Myr]

A base station should have only one cable connecting it to the network. If using the 10GBit card for base band sample exchange in a distributed physical layer (’coordinated multi-point’, ’base station co-operation’), the demonstrator platform uses two cables to be able to handle the high network load. A network processor like IXP28xx would be able to handle the complete load, but unfortunately at the time of purchasing, it was not available on a PCIe board.

8.2.1 PCIe Backplane

The mainboard used is an Asus L1N64-SLI WS [Asu]. It has four highspeed PCIe slots: 2 times x16 (16 lanes, i.e. 32GBit/s bidirectional) and 2 times x8 (16GBit/s). In the demonstrator platform, the two host processors (dual core Athlon 64 FX each) on the mainboard are only used for management and to provide boot images from the local hard disc to the Cell and network processors. A more recent mainboard alternative woud be [Asu09] — which has 7 highspeed PCIe slots.

8.2.2 Cell Processor

Base band signal processing and (de)coding are done completely in software on a Cell processor. With up to 200 GFLOP/s the Cell processor [IBM06, IBM07a] offers considerably higher computational power than currently available (multi-core) DSPs. It consists of a Power architecture core (PPE), eight

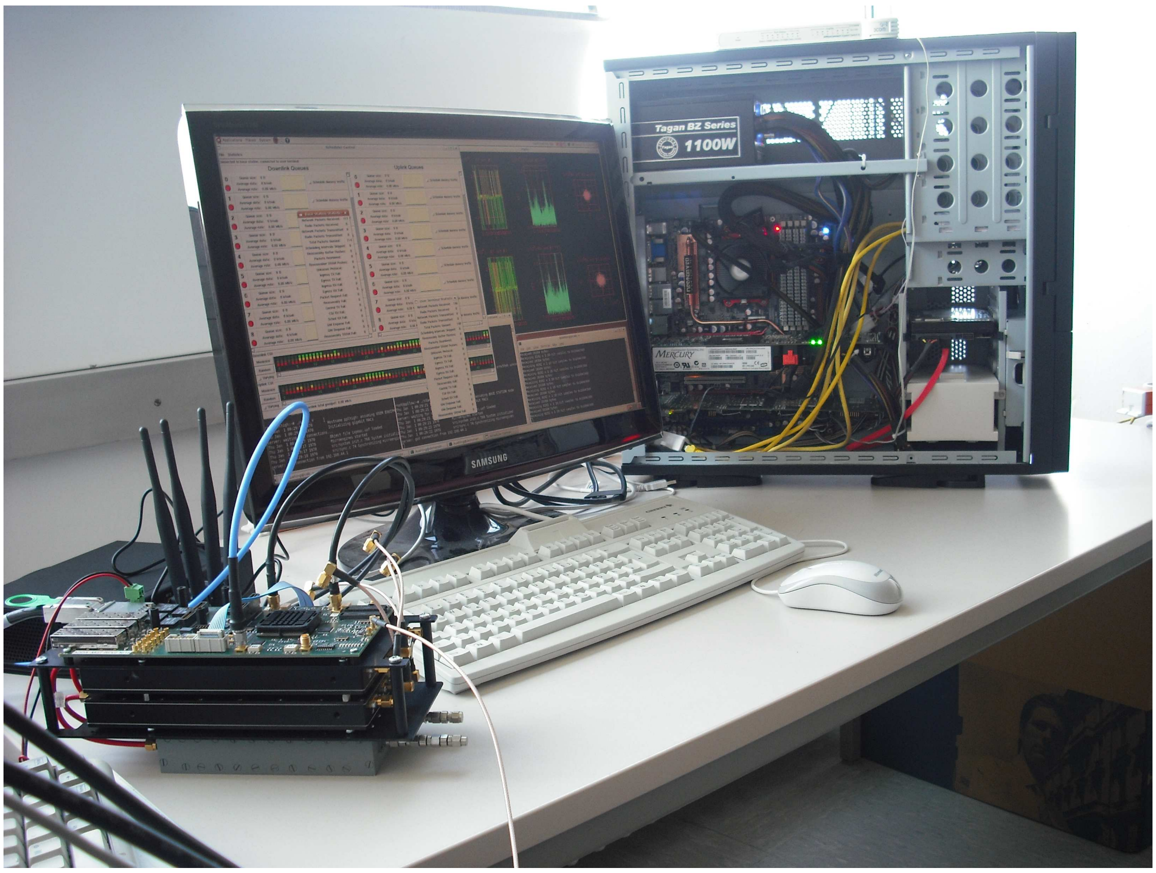


Figure 8.1: SDR demonstrator with two radio frequency frontends in the foreground [Hei09].

co-processors (SPE) optimized for single-precision floating point arithmetic, and on-chip memory and I/O controllers, connected via the Element Interconnect Bus (EIB), which provides a maximum total bandwidth of 192 bytes per CPU cycle [Sca09]. The PPE features a simple dual-issue in-order execution unit. The SPEs also do dual-issue in-order execution but use an instruction set optimized for multimedia and signal processing applications, each operating on a local store (LS) consisting of 256 kB of on-chip SRAM accessible with a latency of only 6 cycles. Larger data-sets can be processed by manually issuing DMA commands for copying blocks of data from/to RAM, somewhat resembling a software controlled cache. SPEs have 128 general purpose registers, each 16 byte wide. Most instructions use these in a single instruction multiple data (SIMD) fashion, with single-precision floating point operations treating each register as a vector of 4 values to be processed in parallel. Each SPE can issue one arithmetic instruction in parallel with one load/store instruction per cycle. Maximum FLOP count is achieved by the single-cycle multiply-and-add instruction, performing 8 FLOP per SPE and cylce. [KDH$^+$05, IBM06, IBM07a, IBM07c] The Cell structure is illustrated in Fig. 8.3c. The Cell on the Cell Accelerator Board (CAB) is clocked with 2,8 GHz, all eight SPEs are programmable. The CAB is shown in Fig. 8.3a. Cell software can also be developed and run on a Playstation 3 (PS3, Fig. 8.3b). The Cell on the

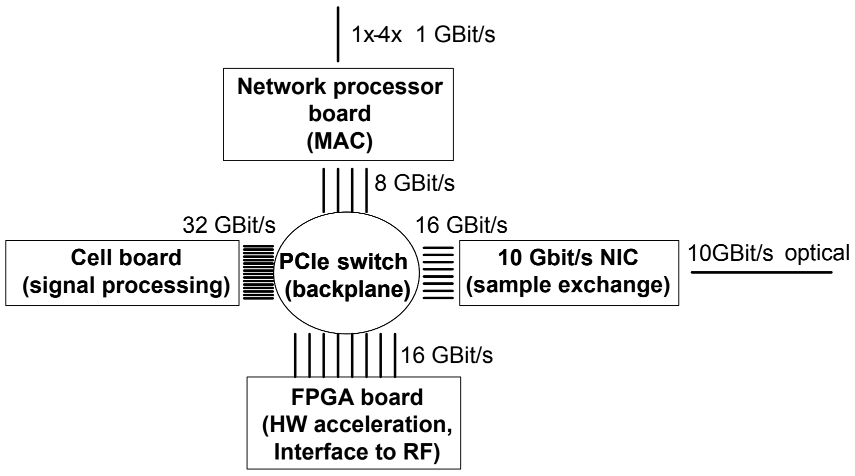
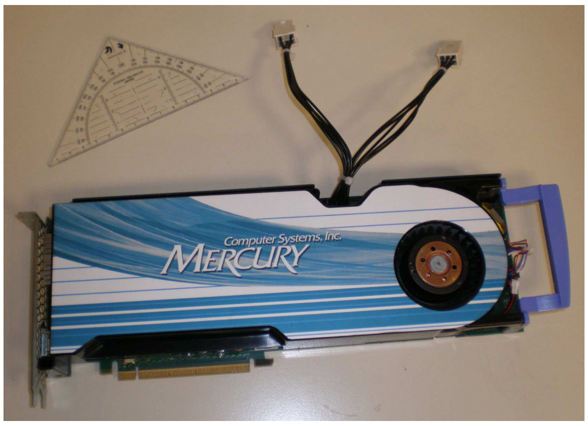

1x-4x 1 GBit/s
Network processor board (MAC)
8 GBit/s
32 GBit/s
16 GBit/s
Cell board (signal processing)
PCIe switch (backplane)
10 Gbit/s NIC (sample exchange)
10GBit/s optical
16 GBit/s
FPGA board (HW acceleration, Interface to RF)

Figure 8.2: System overview with main components [IKK+08].

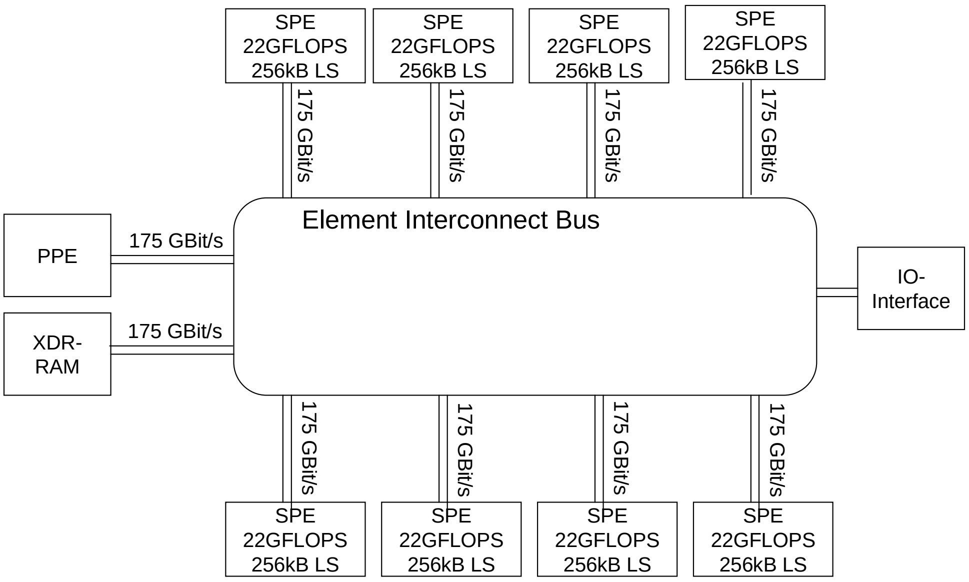
PS3 is clocked with 3,2 GHz, six SPEs are programmable. The PS3 does not offer PCIe slots (only USB and Ethernet). The physical layer is functionally decomposed into signal processing modules running without operating system under hard real-time constraints on the SPEs, while control and management is conveniently implemented on the power architecture core, benefitting from OS functionality.

8.2.3 IXP Network Processor

Network processors (NPUs) are flexibly programmable devices (normally in the C language) which have a specific architecture for efficient examination and manipulation of packet headers. They are used in routers, switches, firewalls, intrusion detection/prevention devices and network monitoring systems [Wik]. The IXP2350 network processor combines a general purpose XScale processor with a set of four simple, 8-times multi-threaded RISC processors called microengines (MEs) [Int05, JK03, Car03]. The microengines run small programs that operate on the majority of packets (‘fast path’), while the XScale is used for exception handling and system maintenance (‘slow path’). The microengines have an 8K instruction program store each (40bit instructions). Each microengine has 256 32bit general purpose registers, which can be accessed in thread-local or in absolute mode (global for the MEs threads). Each ME also has 256 32bit transfer registers for off-chip SRAM, and 256 32bit transfer registers for off-chip DRAM. In addition, there are 128 next-neighbour registers in each ME for communication with the adjacent ME. Two timer registers are available per thread [JK03]. A PCI interface allows the XScale core and the microengines to initiate DMA transfers across the external PCI bus. Other per ME features are context addressable memory (CAM, 16 entries, 32bit tag, 9bit return value, 4bit state) and a CRC unit. For MAC and RLC protocol processing the ’Double Espresso’ board [IP 05] is used. The board contains two IXP2350 network processors, clocked at 900MHz and each capable of processing 2 GBit/s traffic. A block diagram of the board is shown in Fig. 8.4b. The board has an x4 PCI Express connector,


(a) Cell Accelerator Board.

(b) Playstation 3.



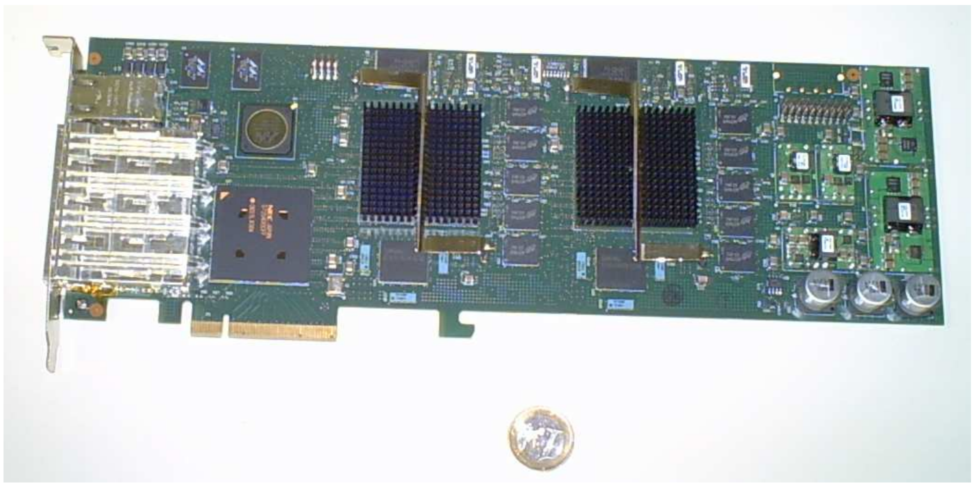
(c) Basic structure and capabilities of a Cell CPU running at 2.8GHz [KDH+05].

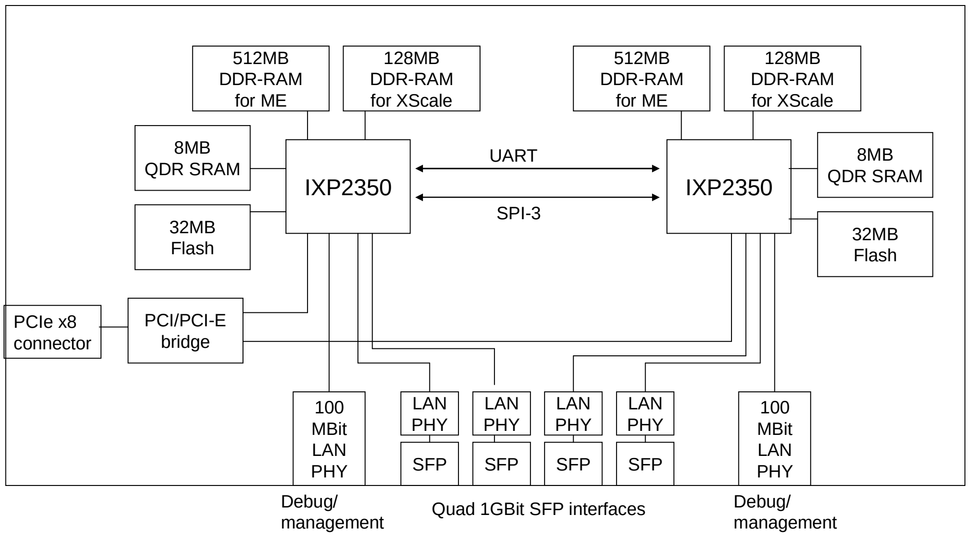
Figure 8.3: Cell CPU and development boards.

four 1 GBit/s Ethernet interfaces (SFP) and an additional pair of 100 Mbps Ethernet ports (used to boot from an NFS server). Each IXP2350 on the board has 128 MB of DRAM for the XScale. The two sets of microengines both have their own 512 MB DRAM as well as an 8 MB SRAM. The IXP2350 only supports PCI (64bit, 66MHz; 533 Mbps bandwidth), so the board contains a PCIe to PCI bridge.

8.2.4 RF Frontends

Bus options for radio frequency (RF) frontend connection include PCIe-over-cable, USB2, Ethernet and proprietary LVDS signalling. The Cell and the frontend device exchange base band samples (16bit integer). Two different frontends are used.


(a) ’Double Espresso’ board.



(b) Block diagram of the network processor board [IP 05].

Figure 8.4: IXP NPU and development board.

2.6 GHz frontend supporting 20MHz bandwidth. The frontend device contains a Virtex5 FPGA for interfacing and digital up-down conversion. The frontend has a PCIe-over-cable connector (and an LVDS connector). It was connected to the demonstrator using an x8 PCIe-over-cable adapter (PCIe plug-in card from [One]). The PCIe endpoint is implemented in the frontend’s Virtex5 FPGA (x8 PCIe) and FPGA memory is mapped into the PCIe address space. A Direct Memory Access (DMA) controller is also implemented in the FPGA, the registers are mapped into PCIe address space.

ISM band frontend with USB interface This is the ’Universal Software Radio Peripheral’ [Ett] with USB interface. It was used with the demonstrator (PCIe form factor) and also with the PS3 (with 5MHz bandwidth).

Other options Interfacing to RF hardware with digital base band interface can also be done using the ML-555 PCIe FPGA board [Xil]. It has an LVDS connector which could be used to connect proprietary RF hardware. Also several commercial channel emulators support a digital base band interface with LVDS. The ML-555 hosts a Xilinx Virtex5 LXT FPGA and has an x8 PCIe connector, so the same PCIe interface can be provided to the software as with the 2.6 GHz frontend. This card was used to implement an (AWGN) base band channel emulator with the same PCIe interface (Sec. 8.4.5).

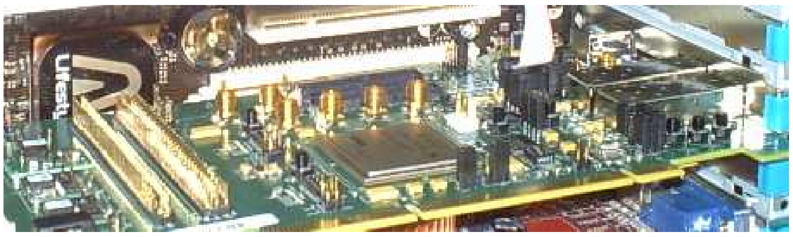


Figure 8.5: ML555 FPGA PCIe board for connecting proprietary radio frequency frontends over LVDS and for base band channel emulation.

8.2.5 Busses

An overview of the demonstrator's bus topology is given in Fig. 8.6. To measure actually achievable bus bandwidths, a test setup with CAB and the ML-555 board has been set up. The test setup and available raw data rates are shown in Fig. 8.7, with the corresponding measured performance numbers on the left. For data transfers utilizing the PCIe bus, write transactions initiated by the data source are preferable over read transactions initiated by the sink. This is due to the asymmetry of PCIe writes versus reads, where reads consist of a two-stage request-response transaction, whereas writes are simply transmitted as unconfirmed messages. For this reason, only results for writes are presented. In the target application with 20Mhz bandwidth and two antennas, the system continuously receives and transmits baseband samples at a rate of 2 Gbits per second and direction (62.5 Msamples/second I/Q parts $\times$ 2 antennas $\times$ 16 bits per sample). DMA has to be employed to achieve transfer rates this high. The measured throughput from Cell (SPE) to FPGA with 4 lanes is 4.5GBit/s (56% raw bandwidth), and with 8 lanes 7.5GBit/s (47% raw bandwidth). For write transactions in the opposite direction (FPGA to Cell), the DMA controller implemented in the FPGA is used. Since the target application's traffic would almost completely saturate 2 lanes, each capable of transmitting 2 Gbit/s, the minimum requirement is a 4-lane-connection. The PCIe interface in the FPGA is implemented using 64kB ring buffers made of blockRAMs for the transmit and receive data, and registers for read/write pointers, status and control registers, test pattern generation control and configuration. All three regions are memory-mapped by host and Cell processor. Towards the digital up/down conversion (or the base band channel emulator), the ringbuffers have 64bit FIFO interfaces transmitting I and Q components of both antennas per clock and direction.

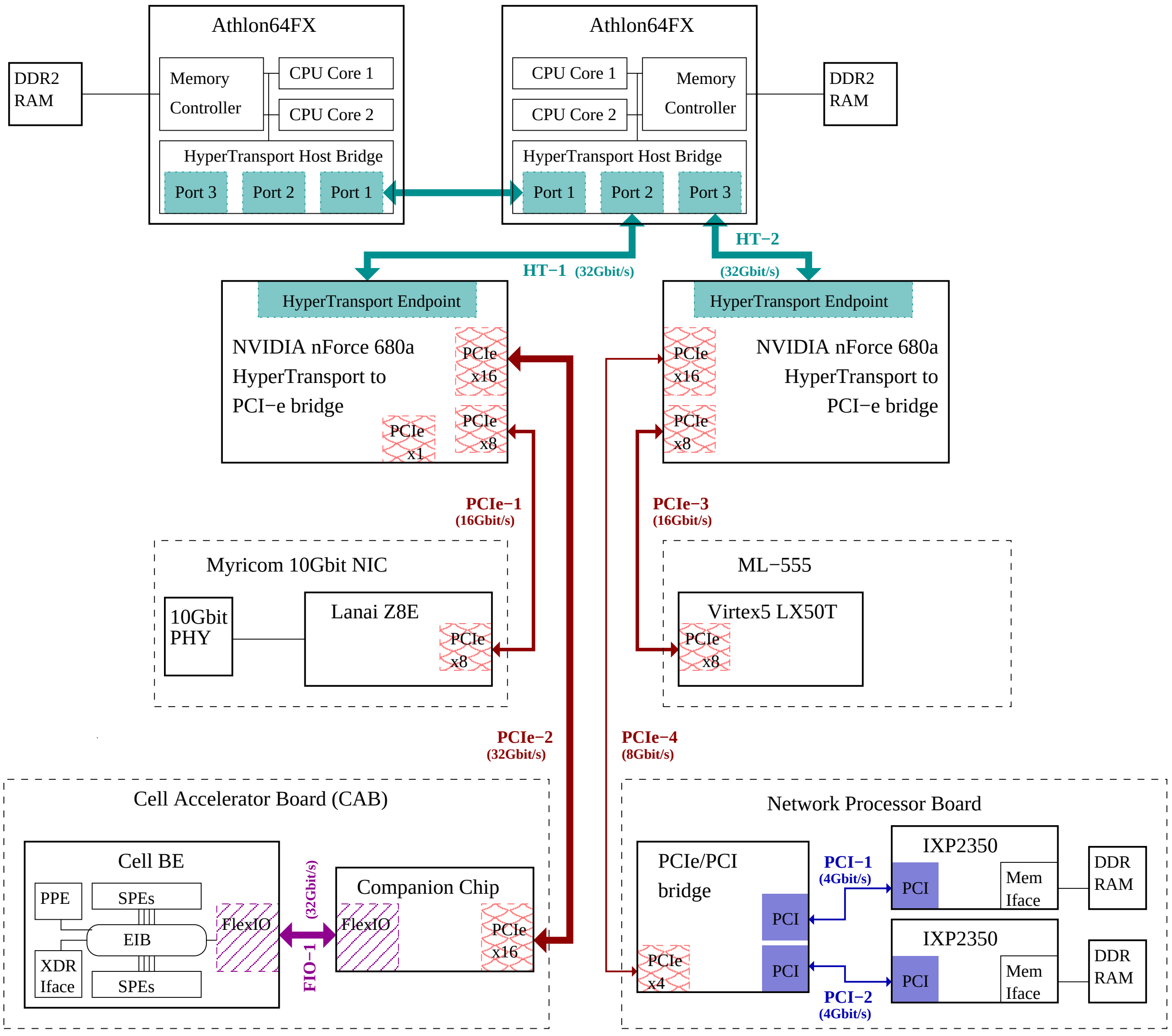


Figure 8.6: Bus topology and available raw data bandwidths [IKK+08].

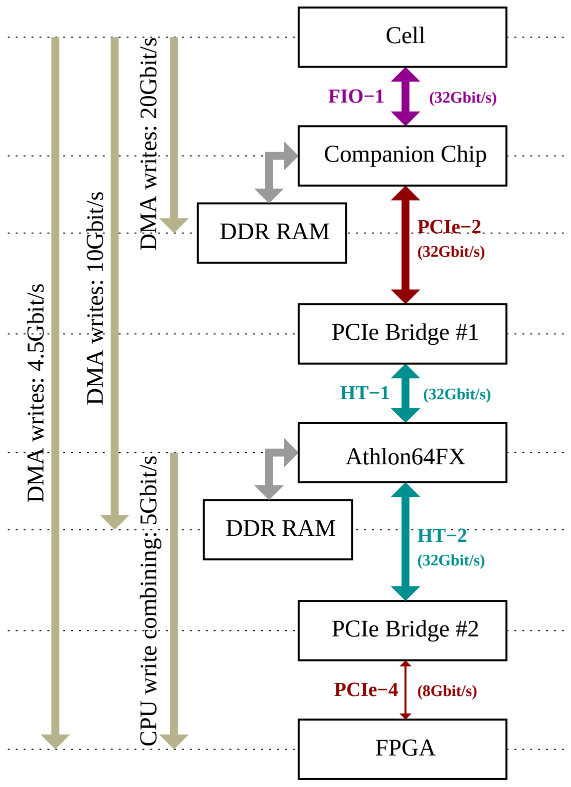


Figure 8.7: Results of performance measurements of data transfers between platform components [IKK+08].

8.3 Software Platform

8.3.1 Linux’ on Host, Cell PPE and NPU XScale

The host is running a Linux 2.6.22 kernel, and provides an NFS server so that the Cell PPE and NPU XScale can boot from the local disc. The Cell PPE (on CAB as well as PS3) is also running a 2.6.22 kernel, and gcc version 4.3 is used. The PPE loads and starts code on the SPEs using the *libspe* library [IBM07a]. The XScale, being an ARM derivative, can also be programmed using standard GNU tools under Linux. Development was done on a standard desktop machine using a self-compiled cross-toolchain [HIK09]. Compilation of the toolchain was aided by the kernel source code provided by [IP 05]. The kernel is a patched 2.4.20 kernel that includes many modules for the Double Espresso board such as Ethernet drivers and memory mapping assignments that enable the XScale to access memory on the microengines’ bus. Additional libraries downloadable from [ixa] offer microengine communication and maintenance functions from within Linux userspace programs. The components map local memories into PCIe address space, so that they can be accessed by other components over PCIe. PCIe address offsets are set by bus enumeration at host boot. Communication between components is exclusively performed using DMA block transfers to and from the recipient’s RAM, cache or I/O areas. A ring-buffer scheme (or multi-buffering) ensures that participants can perform data processing concurrently with transfer of data. Since all communication performed is strictly peer-to-peer (no broadcasts), no sophisticated synchronization schemes need to be implemented. The 2 notifications that need to be handled for a one-directional pipe would be request-to-send and ready-to-read. These can either be communicated via atomic writes and polled reads to locations in RAM, or using DMA writes to event notification registers for hardware components that support that mode of synchronization (such as FPGA and Cell SPE).

8.3.2 Cell SPE Programming

The SPE is supported by the GNU C compiler (gcc). The *libspe* library provides a special instruction set and vector data types for SPE SIMD computation [IBM07b]. SPE programs are perceived as threads, with DMA commands to and from system memory using the same virtual memory addresses as the parent process by which the SPE was started. The SPEs do not access the EIB directly, they send DMA transfer requests to their Memory Flow Controllers (MFC, one per SPE). A single DMA transfer transports up to 16kB. Scatter/gather DMA with DMA request lists is supported. An MFC is accessible through the SPE's registers, which are memory-mapped within the effective address space – so the MFCs can also be used by the PPE. DMA is not only possible between an SPE's LS and RAM, but also between LSs of different SPEs (SPE-SPE DMA): therefore the effective addresses of LS have to be known, which can be exchanged by the PPE. Time measurement (e.g. for throughput measurement) is enabled by an SPE's decrementer register. Atomic operations and mutexes can be used over the *libsync* library. The physical layer implementation was compiled using Cell SDK 2.1 with gcc 4.3-20070713 as replacement for the included SPE C compiler. Several math libraries are provided by the Cell SDK. SPE intrinsics, i.e. basic math and logical operations for fixed- and floating-point SIMD vectors are defined in *intrinsics.h*. Since C-compilers are notoriously bad at extracting such parallelism from program code, these vector intrinsics have to be explicitly used to achieve the SPE's full computational potential. The SIMD math library (*simdmath.h*) provides advanced mathematical operations like trigonometry and logarithm on floating-point SIMD vectors. The MASSV library is similar to the SIMD math library, but operates on arrays. Several functions of *libvector* operate on four vectors at once, *libmatrix* only works on matrices of 4x4 floats, Other math libraries are *liblarge_matrix* (for matrices of any size), the standard basic linear algegra subprogram library *libblas* – of which all routines are implemented for PPE, but currently only a few for the SPE [Sca09] – and *libfft*.

For optimized implementation of the physical layer algorithms, the SDK's math libraries were judged not flexible enough. Using only the SIMD vector intrinsics, an own library was implemented. Signal processing in this library is based on a SIMD vector of four complex floats (data type `c4`, real and complex 4x32bit vectors). This library provides operations with vectors of these SIMD vectors, with variable length (e.g. `c4x4` for 16 complex numbers; operations expanded using preprocessor macros) [KIJ08]. Necessity of this type of matrix-based processing can be understood looking at instruction latencies. SPE load, store and all floating point instructions have a latency of 6 cycles, meaning that although one instruction can be issued per cycle, the results of each instruction are only available to other instructions after 6 cycles have passed. If a result is accessed earlier, execution stalls for the remaining number of cycles [IBM07a]. The C-Compiler tries to reorder instructions for most efficient execution, but depending on the algorithm, stalls are unavoidable. Fig. 8.8a shows how one iteration of a dot product routine already takes 8 cycles to execute. This is unfortunate, since most of the necessary signal processing algorithms are expressed in terms of dot products. Operating on large matrices, one

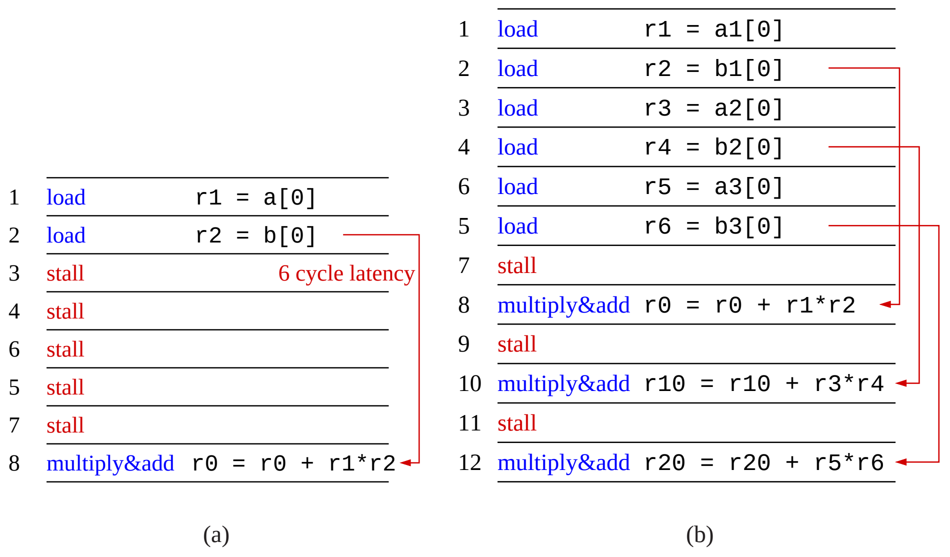


Figure 8.8: (a) Long latency of 6 cycles leads to stalls for one iteration of a dot product. (b) The 3-way parallel dot product reduces percentage of stalled cycles due to more parallelism in the data flow [KIJ08].

can usually extract enough parallelism from an algorithm to make full use of SIMD instructions while avoiding any stalls. [KBD07] shows how a block matrix based linear equation solver for large matrices is implemented on the Cell processor, using the SPEs to operate on matrices of fixed size 64×64, achieving up to 175 GFLOP/s on a single Cell CPU. For several instances of small problem size, parallelization is more efficient over the problems (solving several at once), e.g. MIMO stream separation on different subcarriers. Parallelism is then created external to the algorithm, making the algorithm process N data sets in one run. N need not necessarily correspond to the widths of data elements processed by the SIMD instructions. Adding more than just the SIMD parallelism greatly helps to reduce processor stalls, as shown by the 3-way parallel dot product in Fig. 8.8b. This can be seen as a logical extension of the SIMD concept. Instead of generating just one instruction for a 4-way parallel multiply, it is preferred to have the compiler generate 3 instructions for a $3 \cdot 4$ parallel operation. Using standard C syntax, a 3-tuple structure and corresponding inline functions can be created to operate on these data items.

The resulting programming model can be called ’pseudo-SIMD’ [KIJ08], since from the programmer’s point of view, she just programs for a very wide SIMD architecture. Using C preprocessor macros, an algorithm implementation can be made independent from the underlying pseudo-SIMD data width and the programmer can choose the optimal level of parallelism at compile-time. Fig. 8.9 shows how the parallelization grade affects performance of the 12×12 MIMO channel matrix inversion code from [KIJ08], running on a single SPE. The throughput has its maximum at a parallelism of $4 \cdot 4$, processing 16 channel matrices in one run. At higher parallelism, the compiler starts generating inefficient code due to limited number of registers and probably limits in the compiler’s abilities to analyse the increasingly complex data flow.

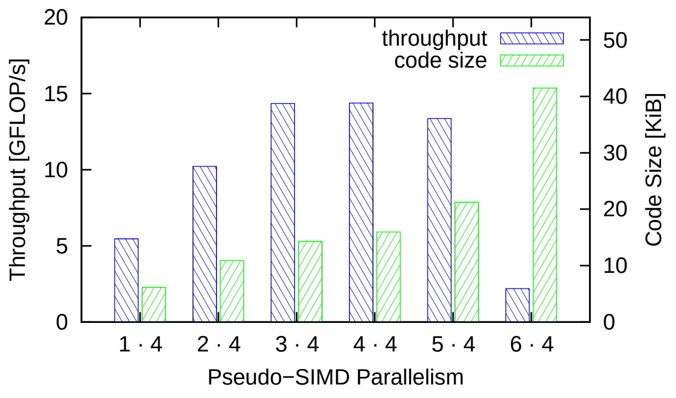


Figure 8.9: Single-SPE throughput of 12×12 MMSE pseudo inverse depends on pseudo-SIMD parallelization factor chosen at compile-time [KIJ08].

8.3.3 NPU Microengine Programming

The microengines are programmed using a proprietary software development kit which can be downloaded without cost from [ixa]. The SDK includes an assembler and a C compiler for the MEs. It also includes an IDE with architecture tool (defining packet processing stages and tasks) and simulator with packet generator. MEs’ code is loaded and started over the XScale using the *resource manager* library. Typical data plane functionality includes packet classification, segmentation, reassembly, packing, addressing and queueing. For standard functions like packet receive and transmit, a *microblocks library* with optimized microcode is available. The MEs transfer data to and from scratch memory, SRAM, DRAM, Media Switch Fabric (MSF) and PCI. Access times for the different memory types from MEs in IXP2xxx processors are listed in [Car03]: local memory (inside ME, 640 x 32bit) has a latency of 4 cycles, scratch memory and message SRAM (16kB and 128kB resp., on-chip, global to MEs and XScale) are accessed with 80 cycles, SRAM (off-chip) access needs 130 cycles and DRAM (off-chip) access 300 cycles. The programmer can explicitly decide where to put data with a type modifier. A simple memory test application has been developed that measures the average access times for scratch memory, SRAM and DRAM under varying bus loads. The access time for any memory was measured to be roughly constant with respect to reference count, i.e. the number of long- or quadwords transferred. The time increases proportionally to the number of threads accessing a given memory regardless of the distribution of threads among the microengines. A summary of read access times for a constant reference count of eight is given in Figure 8.10. Write access times are similar. The latencies are used in Sec. 8.4.4 for performance estimation and choice of necessary number of threads. Data plane functionality can be implemented a pipeline of threads (threads work on the same packet) or in parallel threads (threads work on different packets). Thread execution is non-preemptive. When a thread yields the ME, a hardware arbiter selects the next thread to run among the non-blocked threads. There are two compiler modes: explicit partitioning (EP, the programmer determines which threads on a ME executes which code) and autopartitioning (AP, automatic ME assignment). In AP mode, there are C language extensions e.g. for determination of packet processing stages, inter-PPS communication, path annotation (e.g. critical path) and storage class declarations (DRAM, SRAM etc). There are several possibilities for inter-thread communication. Rings (FIFOs) in scratch memory or SRAM can be used, which can be written by any producer/consumer including the XScale with atomic operations. Scratch memory supports 16 rings with atomic *put* and *get* operations. Another possibility is that an ME writes to the adjacent ME's next-neighbour registers. For threads inside an ME, a *reflect* can be used to write into transfer registers of another thread on the same ME. Or local memory can be used, global to the threads of an ME. The EP compiler mode provides signal intrinsics between threads on the same or different MEs. Free lists of buffer pointers are normally kept in SRAM.

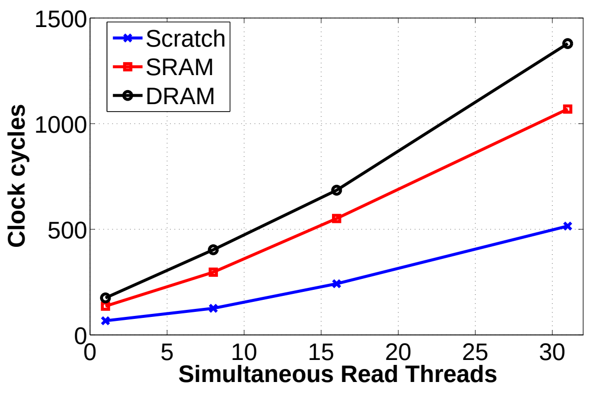


Figure 8.10: Read access time with increasing bus load for reference count (number of long- or quadwords transferred) of eight [HIK09].

8.3.4 TCL Scripting

For more comfortable development, testing and debugging, TCL (Tool Command Language, [OJ09]) is used. A TCL interpreter is running on the PPE (and one on the Host). The demonstrator is started by logging into the PPE and starting a TCL script. Code loading into the SPEs and starting the protocol application on the NPU are wrapped in TCL. The SPE signal processing modules are selected in the TCL script, and the FIFO adresses are exchanged. The protocol stack mode is set to base station, terminal or multi-terminal emulation (Sec. 8.4.4). TCL scripts are also used for testing and debugging and for GUIs (Sec. 8.4.5). The instanciation of SPE signal processing modules as TCL script bears resemblance to the receiver description language from chapter 6.

8.4 Implemented Functionality

8.4.1 Modem

This subsection describes Cell implementations of (uncoded) Modem functionality, i.e. MMSE MIMO demapping, QRD-M MIMO demappping, FFTs and channel estimation.

MMSE MIMO Demapper based on Greville Algorithm

For MMSE MIMO demapping, both the Greville-based algorithm from Sec. 3.2.3 and the Cholesky based algorithm been implemented and compared as optimized programs on the SPE. To make full use of the parallel MAC operations, the Greville-based implementation computes 4 equalization matrices in parallel. Completely unrolling the innermost loop of the algorithm allows it to outperform the Cholesky based implementation by up to a factor of 3.7. Comparable unrolling is not possible for the Cholesky based algorithm since all loops are of variable length. However, the Cholesky based method can be unrolled along the outer loop when operating on many matrices. In that case peak performance is achieved when working on 16 matrices in parallel, which is still substantially slower than the Greville-based algorithm [KI08]. The results for quadratic matrices are illustrated in Fig. 8.11. For 12x12 matrices, the Cholesky-based code has a size of 1536 bytes and achieves a utilization of 2.11 MAC/cycle (of 4 MAC/-cycle), the Greville-based code has a size of 3084 bytes and achieves a utilization of 2.53 MAC/cycle.

List-QRD-M MIMO Demapper

Here, SPE implementation and benchmarks of the algorithm from Sec. 3.2.3 are described. The implementation assumes separable QPSK or 16QAM modulation sets and an equal number of transmit and receive antennas. For QPSK modulation up to 16x16 antennas and for 16QAM up to 8x8 are supported. The modulation can be different for different PRBs (assuming LTE format). Any value for parameter M is supported, and a priori LLRs are used. Set partitioning is not applied. For LLR generation the Max-

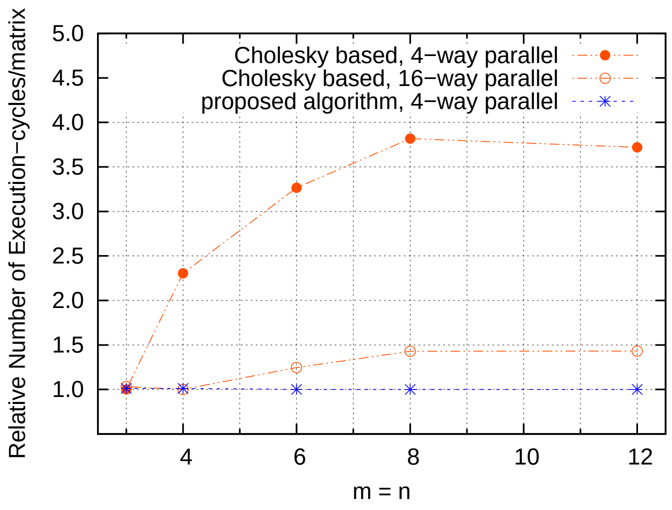
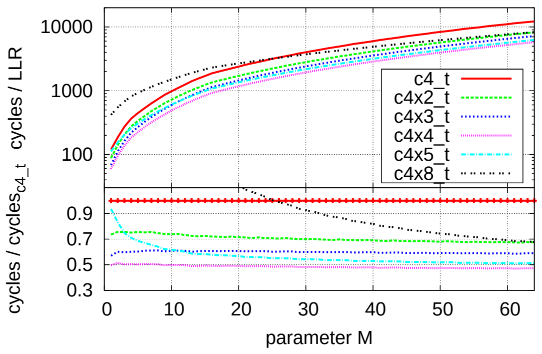
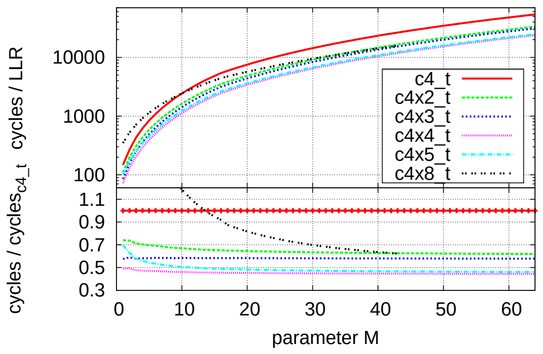


Figure 8.11: Relative number of execution cycles per calculated matrix. Number of cycles is normalized for every *n* to the best-performing algorithm [KI08].

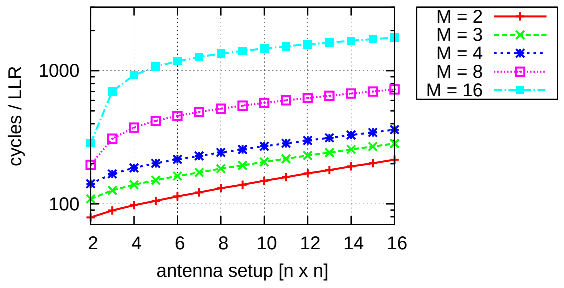
Log approximation is used. If the candidate list does not contain a bit (counter-)hypothesis, clipping is applied as in [Zim07]. Implementation uses the library for vectors of complex SIMD vectors (Sec. 8.3.2). List-QRD-M MIMO demapping consists of QR-Decomposition, M-algorithm tree search and (clipping) Max-Log LLR generation. The implementation assumes low mobility and performs QR-decomposition only every 5th received symbol vector. Three loops are performed per (vector of) transformed received symbol vector(s). The outer loop is over the tree layers (transmit antennas), the middle loop is over the surviving candidates of the previous layer (maximal M), and the inner loop over all possible modulation symbols of this layer (node expansion). Sorting the metrics in descending order is performed using the selection sort algorithm. The benchmarks in Fig. 8.12a and Fig. 8.12b compare the necessary cycles per LLR for QPSK and 16QAM for different values of M and for tuple types `c4_t` to `c4x8_t`. The highest throughput is again achieved with `c4x4_t` (16 complex symbols) — throughput compared to using `c4_t` is doubled. Fig. 8.12c and Fig. 8.12d show the complexity increase with larger number of antennas (using `c4x4_t`). Since the search tree for 16QAM is 4 times wider than for QPSK (16-ary versus 4-ary), 4 times as many metrics have to be computed and sorted — without set partitioning, complexity for 16QAM is up to 4 times higher compared to QPSK. Throughput for 4x4 on one SPE on the CAB (2.8GHz) is shown in dependence on M for both modulations in Fig. 8.12e. 10 Mbit/s per SPU are achieved for QPSK for $M = 6$ and for 16QAM for $M = 4$.



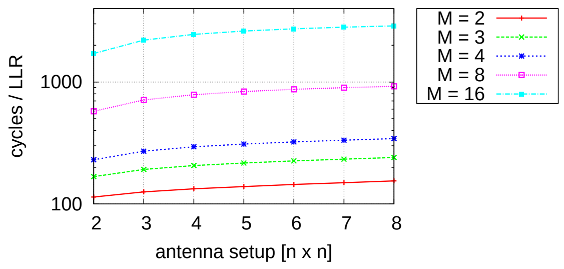
(a) Complexity increase with M for 4x4 QPSK.



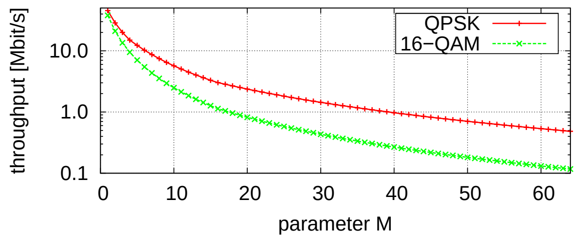
(b) Complexity increase with M for 4x4 16QAM.



(c) Complexity increase with number of antennas for QPSK.



(d) Complexity increase with number of antennas for 16QAM.



(e) Throughput on one SPE for 4x4 QPSK and 16QAM in dependence on M.

Figure 8.12: List-QRD-M MIMO demapper benchmarks on Cell SPE [WKI].

Fourier Transforms, Channel Estimation and PRB-(De)Mapping

FFT / IFFT are implemented using the radix-4 decimation-in-frequency algorithm. The highly regular code structure achieves a high processor utilization: the FFT code achieves around 3.4 MAC/cycle on an SPE, IFFT achieves 3.3 MAC/cycle. For length-2048 FFT on the PS3 (with 3.2 GHz) this corresponds to 118807 FFT/s per SPE. 2048-FFT computation for 20MHz 2x2 MIMO utilizes an SPE to 20%. Channel estimation assumes the LTE 2x2 downlink pilot pattern. For higher number of transmit antennas, this pattern is extended with the same pilot density. The implemented channel estimation algorithm follows [SJ06]: 1D static Wiener filtering and SNR estimation with subspace method. The interpolation matrix is chosen according to estimated SNR from a precomputed set of matrices. Usage of the SPE's LS for

Program Component	**Size [KiB]**	**LS usage**
data buffers	187.5	73.2%
free space (available for stack, heap)	30.2	11.8%
channel inversion code	15.9	6.2%
C runtime library	10.5	4.1%
constant matrices W, u for interpolation and SINR estimation	6.2	2.4%
channel interpolation and SINR estimation code	4.4	1.7%
DMA communication code	1.3	0.5%

Table 8.1: LS memory utilization of SPE for 12x12 MIMO channel estimation and equalization [KIJ08].

uncoded modem benchmark	cycles/slot	MBit/s
SISO, 1024-FFT, no equalization	249422	292
SISO, 1024-FFT, with channel estimation and equalization	347206	210
MIMO 2x2, 512-FFT, with channel estimation and MIMO MMSE equalization	365823	199

Table 8.2: Time domain loopback through emulated AWGN channel on one SPE of PS3, using 64QAM.

channel estimation and (Cholesky-based) MIMO stream separation for 12x12 MIMO is shown in Tab. 8.1. The code for symbol mapping, PRB mapping and pilot insertion achieves 683,9 Msymbols/s for 64QAM on a PS3 (4% utilization of one SPE on PS3 for 20MHz 2x2). PRB demapping and 64 QAM per-stream soft demapping achieves 214,7 Msymbols/s (14% utilization of one SPE on PS3 for 20MHz 2x2). Uncoded MMSE MIMO modem benchmark (time domain loopback through base band channel emulated in SPE) results are shown in Tab. 8.2.

8.4.2 Error Correction

Error correction decoding is a computationally very expensive part of wireless receivers. There are several papers on software FEC implementations for SDR. [VS01] presents a UMTS turbo decoder (two concatenated 8-state convolutional decoders) on a 933MHz Pentium 3 processor. It uses single-precision arithmetic and achieves a throughput of 366kBit/s per iteration, running up to 14 iterations. Memory usage is 200kByte. [LMM+06] presents an implementation of the UMTS turbo code achieving 2MBit/s on a 400MHz 32bit DSP (using SIMD instructions). Other software implementations of this decoder achieve 1.8MBit/s on a Starcore [KMGW03] and 1.48MBit/s on an Xtensa based processor [GTW03]. [GZC+09] presents a WiMAX Turbo decoder implementation on the Cell, which achieves 1.4 Mbps on an SPE running at 3.2 GHz. [FSS11] describes an LDPC decoder implementation on the Cell. [FAFF02] presents a comparison between different decoder algorithms implemented in software, regarding the needed processor cycles per decoded bit on X86 and PowerPC general purpose processors.

This subsection gives details about SPE implementations of a 64-state soft-decision Viterbi decoder (like used e.g. in IEEE 802.11a) and an 8-state BCJR and Turbo decoder (like used e.g. in LTE). The implementation runs on one SPE. Parallelization is done in a way that different SPEs concurrently process different packets. The decoder uses only registers and the local store. Over the Cell Element Interconnect Bus (EIB) only LLRs as input and decoded bits or LLRs respectively as output are transferred.

Convolutional encoder

The discrete convolution can be written in time domain in matrix form or in (shift-) transform domain with generator polynomial. For encoding in hardware, a shift register with tapped binary delay elements and an exclusive or (XOR) combiner like in Fig. 8.13 is used. The stream of information bits is entering from the left into the shift register. The shift register has $L-1$ binary delay elements and additional taps before, between and after the delay elements. Altogether there are L taps. The generator polynomials declare which taps are connected to the XOR-combiner. The value of the least significant bit (LSB) of the generator polynomial number is representing the connection of the last delay element. In Fig. 8.13 the LSB of both generator polynomials are one. The upper XOR-combiner represents the first generator polynomial 171_8. At the output the two resulting bits are concatenated. The shift register is initialized with zeros at all binary delay elements. After the last information bit entered, the shift register may be flushed with $L-1$ zeros to bring the register to a predefined end state with generating $2 \cdot (L-1)$ tail-bits. Software implementation might use a register and shift operations, but higher throughput necessitates using iterative table look-ups for partial bit sequences of certain length.

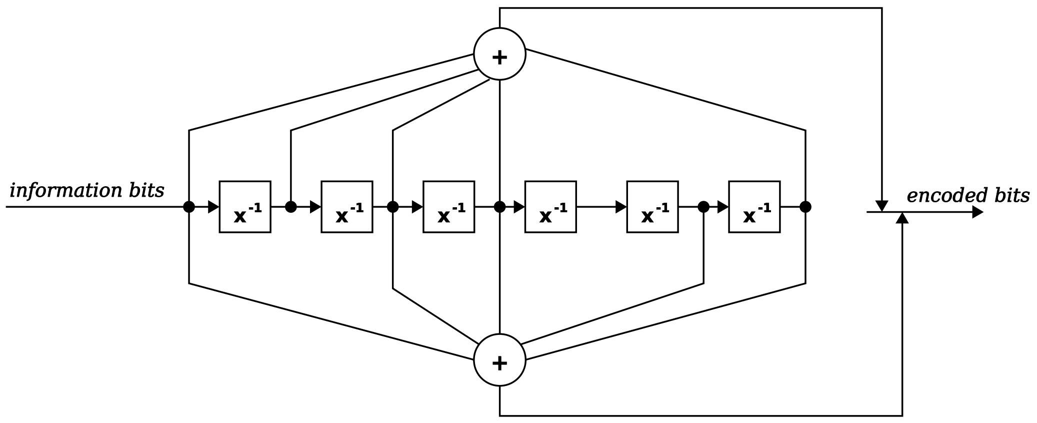


Figure 8.13: Encoder shift register for 802.11a generator polynomial (171_8;133_8), code rate 1/2, constraint length 7 [WKI08].

Viterbi Decoder

The Viterbi algorithm finds the minimum weight path through the code trellis (folded code tree). Implementation is for a rate 1/2 code with constraint length 7 (trellis with 64 states) and uses soft decisions with 16bit integer values for the metrics. The algorithm consists of two phases: in the first phase, path metrics through the trellis and information about the most likely predecessor of each node are computed. In the second phase, the minimum-weight path is traced back beginning from the (either predefined using tail-bits or most-likely) end node to obtain the information sequence. If there is not enough memory available to hold the trace back information for the complete packet, the two phases are performed on parts of the packet (truncated Viterbi), which potentially leads to a slight accuracy degradation [LC04]. As example decoder parameters, the generator polynomials of the 802.11a standard, 171_8 and 133_8, are used (binary representation is 1111001 and 1011011). In the implementation they define the shuffle mask for branch metric generation and the shuffle masks for Add-Compare-Select (ACS) implementation. The

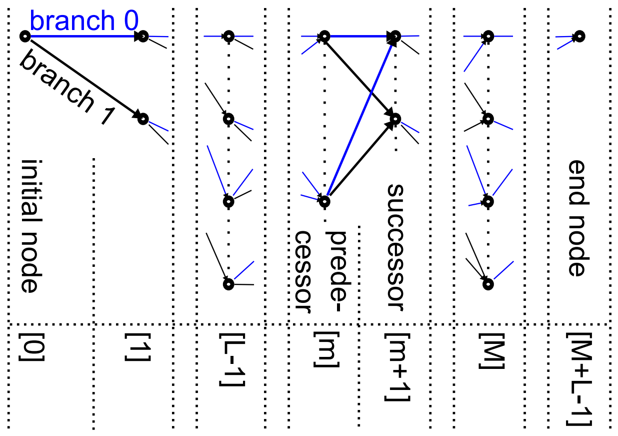


Figure 8.14: Convolution trellis with packet size M and constraint length L [WKI08]

trellis contains two sorts of information: on the one hand the path metrics describing the distance between the received coded sequence and paths through the trellis (code words), and on the other hand the most likely predecessor of any node in the trellis. The latter can be thought of as a separate predecessor trellis and constitutes the trace back information. The path metrics are computed level-by-level (state transitions) as sums of branch metrics.

Compute branch metrics: the squared Euclidean distance (regarding bit positions as dimensions) between the received code symbol **r** and every possible symbol **b** (branch word, Fig. 8.15) for code rate $1/N$ (here: N bits branch word) is:

$$\mathbf{d}_E = \sum_{n=0}^{N-1} (\mathbf{r_n} - \mathbf{b_n})^2 = \sum_{n=0}^{N-1} \left(\mathbf{r_n^2} - 2\mathbf{r_n}\mathbf{b_n} + \mathbf{b_n^2}\right) \tag{8.1}$$

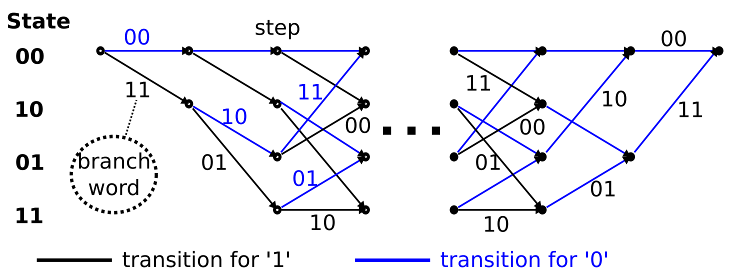


Figure 8.15: Four state trellis (generator polynomial 7;5, code rate 1/2) [WKI08].

Considering the constant value (energy) of the squared terms, it is sufficient to evaluate only the middle (dot product) terms [LC04]:

$$\mathbf{d} = \sum_{n=0}^{N-1} (-\mathbf{r_n b_n}) \tag{8.2}$$

For code rate 1/2 this becomes:

$$\mathbf{d} = -\mathbf{r_0 b_0} - \mathbf{r_1 b_1} \tag{8.3}$$

Thus, the branch metric can be computed with an Add operation. With code rate 1/2 there are only four possible branch metrics, where two of them are just inverses of the other two. These four branch values have to be calculated per time instant. Afterwards they are distributed to form the branch word. The structure of this mapping to the branch word can be precomputed. The implementation uses a shuffle mask as parameter for the SPE instruction `spu_shuffle` to permute vectors. With 16bit metric values, eight values can be stored in one SIMD vector (8*16bit=128bit). For high SPE utilization, 16 received symbols *r* are computed in one call (the information word contains an integer number of bytes). The branch metric values are computed by equation 8.3. Afterwards an unrolled loop is used to perform the following steps: distribute branch metric with precomputed shuffle mask, perform ACS butterfly operation to compute path metric, and store bit of most likely predecessor.

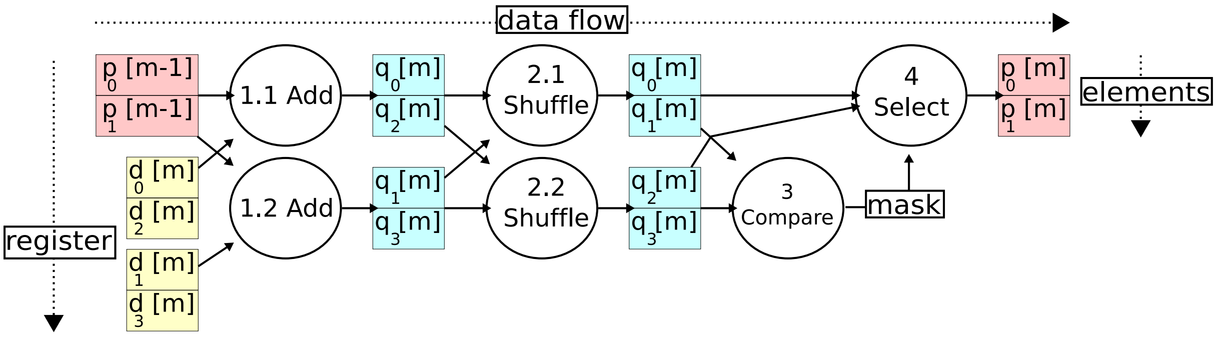


Figure 8.16: Implementation of ACS butterfly operation by Add-Shuffle-Compare-Select instructions. Calculation of path metric p of time instant m by sum of path metric $p[m-1]$ and branch metric $d[m]$ [WKI08].

Compute path metrics, Add-Compare-Select operation: the path metrics of the successor nodes are computed by summing the actual branch metric and the path metric of the survivor path. The necessary operations for each level are therefore: *Add* branch metrics to path metrics, *Compare* the two metrics of the arriving paths in each node and *Select* the better one (survivor path). The path metrics are computed on the fly and kept in registers: there is only one vector of path metrics necessary which describes the actual computed time instant. The ACS butterfly operation [LMM+06] is implemented as an *Add*, *Shuffle*, *Compare*, *Select* operation. *Shuffle* is a permute instruction and used for permuting the path metrics in enumeration order. This allows for 4x SIMD parallel processing of the ACS butterfly operation. Path metrics are kept 8x parallel in the 128 bit registers. ACS implementation with SPE intrinsics is illustrated in (Fig. 8.16): the branch metric is added with (`spu_add(p,d)`) to the path metric of the predecessor node. Since every node has two outgoing branches, two different branch metrics are added to the predecessor path metric. They are stored in two different vectors. The two vectors are permuted with (`spu_shuffle(`p_{left},p_{right}`,pattern)`) to order them according to their branch number s_b (enumeration order) and for the following constraint to be fulfilled: the element of first returned vector and corresponding element of second returned vector have the same state number $s = s_b mod(2^{L-1})$. `spu_shuffle` has to be performed twice, because it returns just one vector. Both vectors are compared with each other by `spu_cmpgt(`p_{left},p_{right}`)`, which returns a compare mask. Bits of the returned vector are set to one if the corresponding element in p_{left} is greater than the element in p_{right}. `spu_sel(`p_{left},p_{right}`,compare mask)` returns the elements with smaller path metric. This vector represents the path metric of the actual time instant and replaces the path metric of the previous time instant. For the 64-state trellis, path metrics for one time instant are stored in 8 128bit SIMD vectors. This leads to high parallelization gain by reduced latencies: these eight vectors are independent and so is the ACS operation on them. Loop unrolling enables the compiler to exploit this gain. Of course loop unrolling also blows up the generated code and so the used memory in the local store.

Update trace back information (predecessor trellis): The result of the *Select* operation (survived predecessor node) has to be stored in a predecessor trellis to later enable trace back. The information to be stored is one bit for each node (since every node has just two predecessors). The predecessor trellis is built in local store. 64 bits are used for all 64 states, so that in a 128bit vector two time instants of the trellis are stored. This is the only data structure in LS, all other variables are temporary and kept in registers. The most likely predecessor node was already computed by the compare intrinsic `spu_cmpgt`. Elements with a zero indicate that the node with the smaller state number $\lfloor s[m]/2 \rfloor$ is the survivor node. Otherwise $\left(\lfloor s[m]/2 \rfloor + 2^{L-2}\right)$ is the state number of the survivor node. `spu_gather(compare mask)` extracts the LSB of every element and return them concatenated in one integer element. This has to be

Program Component	**Size [KiB]**	**LS usage**
data buffer for predecessor trellis	48.3	18.8%
Viterbi decoder machine code	13.3	5.2%
C runtime library	5.2	2.0%
path metric shuffle masks	0.4	0.2%
free space	215.7	73.8%

Table 8.3: Local Store memory utilization of implemented decoder (802.11a polynomial and MLD for packet size of 6144 Bit) [WKI08].

performed for all compare masks. The final vector with the information about the survivor nodes results from merging the computed vectors to one SIMD vector. Two time instants are stored in one SIMD vector (two time instants, 64bit*2=128bit).

Path trace back: the decoded bits (information bits) are obtained as path through the predecessor trellis. Trace back starts either when the trellis is built completely up to the end of the packet, or it starts earlier due to limited (trace back information) memory or limited tolerable processing delay in stream processing. Trace back from the end of the packet starts in the defined end node when tail-bits are used. Trace back starting somewhere in the middle of a packet starts from the node with the best path metric so far (truncated Viterbi). This is suboptimal, but the trace back path converges to the optimal path after a certain length. Therefore, the first decoded (traced back) bits are not used, and the subtrellises for stream decoding overlap. It is common to trace back for around five times the constraint length before using the obtained information bits [LC04]. At the end of every decoder function call, the trellis memory is checked. If the memory is full trace back is performed. Obtaining the node with the smallest path metric is a compute-expensive operation. It can not be parallelized in vector arithmetics. The available trellis memory can be defined as parameter, but should fit in the local store. Since the local store is comparably large, also large packet sizes with over 10kB can be supported with MLD performance.

Implementation results: the performance of the decoder implementation depends on the size of memory used in the SPU local store. Usage of the LS shown in Tab. 8.3. Due to loop unrolling and inline function usage, 13,3KiB are occupied by the decoder code. This code size is of course scalable. The allocated memory for the predecessor trellis can vary between 3KiB and 200KiB, which is influencing the speed of the application and the MLD performance for different packet sizes. Fig. 8.17 gives a detailed view on the dependence between predecessor trellis size and performance. At small local storage, 8 Mbit/s performance are lost, but already at around 20KiB memory for the trellis (2048 data bits per trace back), almost the maximum throughput of 33 Mbit/s is achieved (with 2.8GHz, on CAB). Assum-

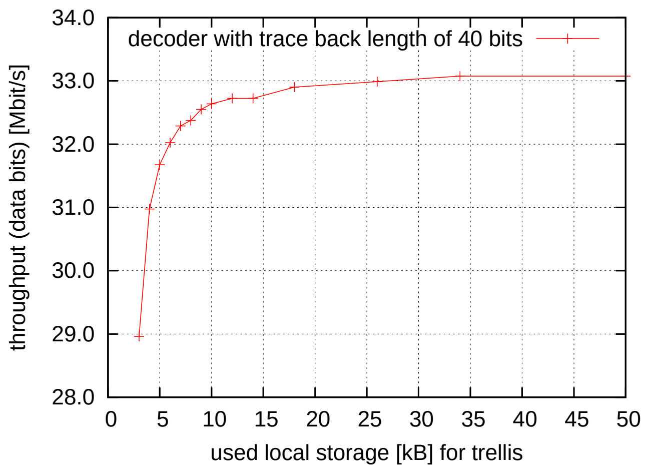


Figure 8.17: Throughput of decoder depending on trellis memory usage. (generator polynomials $171_8; 133_8$). Benchmark on one SPE on CAB [WKI08].

ing optimal parallelization without latencies and 16bit metric values, branch metric computation takes 16 cycles per decoded bit, ACS-butterfly takes $2 \cdot 16 + 2 \cdot 8$, and predecessor trellis operations $8 + 4 + 2 + 1$. So a minimum number of 79 cycles per decoded bit are required [WKI08]. Some operations like shuffle can be performed by both of the independendt SPE pipelines (`spu_sl` and `spu_slqw` instructions). So with the additional hypothesis of optimal distribution of the operation on both SPU pipelines, 40 cycles per decoded bit is the theoretical maximum performance on one SPU. The presented implementation achieves 85 cycles per decoded bit by 50KiB local storage and 40 KiB trace back length, which corresponds to 47% of the maximum. In [FAFF02] the hand optimized assembly Viterbi decoder takes 108 cycles/bit on a Pentium III with SSE optimization (128bit wide). Performance of the presented implementation could be further increased, for example by reducing the path metric number format to 8bit. This may necessitate changes in normalization. Currently the path metrics are normalized in certain distances by subtracting the path metric of state 0 from the branch metric.

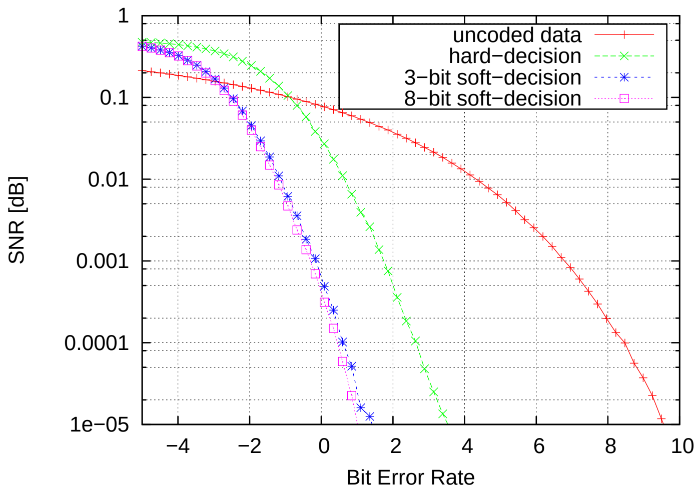


Figure 8.18: BER Performance of BPSK in AWGN (polynomial $(171_8; 133_8)$) [WKI08]

Turbo Decoder

An 8-state Max-Log BCJR decoder and QPP Turbo (De-)Interleaver are implemented, with LTE code parameters. The BCJR algorithm formulates the probability of state transition in the trellis from state s' to s at time instant l given received sequence $\mathbf{r}$ using Bayes' rule [LC04]:

$$\begin{aligned} p(s', s, \mathbf{r}) &= p(s', s, \mathbf{r}_{t<l}, \mathbf{r}_{t>l}) \\ &= p(\mathbf{r}_{t>l}|s)p(s, \mathbf{r}_l|s')p(s', \mathbf{r}_{t<l}) \\ &= \beta_{l+1}(s)\gamma_l(s', s)\alpha_l(s') \end{aligned} \tag{8.4}$$

with forward path metric α, backward path metric β and branch metric γ. The decoder traverses the code trellis both in forward and backward direction, computing the forward and backward path metrics using Add-Compare-Select (ACS) operations. The presented implementation again uses 16 bit integer arithmetic, allowing to compute metrics for the 8 trellis states in parallel by using the 128 bit wide SIMD operations ($8 \cdot 16$bit= 128bit). The ACS operation using Add, Shuffle, Compare and Select SIMD instructions is illustrated in Fig. 8.19. The implemented algorithm has two phases. In the first phase, a loop concurrently computes forward path metrics for the first half of the packet and backward path metrics for the last half (using a forward moving window on the trellis for α, and a backward moving window for β computation). Normalization is performed by subtracting the path metric of state 0 from all states' metrics in certain distances. At the end of the loop, forward and backward traversal meet in the middle of the packet. The maximum size LTE packet consists of 6144 information bits, which needs $6144 * 8 * 16bit = 98kB$ memory for path metrics in LS (48kB for α and 48kB for β). In the second phase, a loop continues the bidirectional traversals and computes APP-LLRs. LLR reconstruction uses Compare, Select, Shuffle and Subtract SIMD instructions.

With the same generator polynomials, the BCJR is around 3-times as complex as the Viterbi algorithm (assuming about equal computational effort for forward metric computation, backward metric computation and APP-LLR computation [LC04]). When reconstructing also the parity bit APP-LLRs, it is roughly 4-times as complex compared to Viterbi.

The implementation achieves 23 cycles/uncLLR on one SPE, which on the CAB corresponds to 140 MuncLLR/s.

The QPP interleaver implementation exploits the the fact that QPP interleavers are vectorizable [Nim08]. It uses a window size of 8 LLRs, matching the 128-bit operand size of SIMD instructions. To output one interleaved window, 8 source windows are loaded, using offsets computed from the QPP polymomial, implementing the inter-window permutation. Intra-window permutation is then performed using Shuffle and Select instructions using the constant intra-window permutation pattern [IKWB09]. The (de-)interleaver implementation achieves 2.6 cycles/LLR.

With 8 iterations, the resulting Turbo decoder achieves 17MuncBit/s on one SPE on the CAB (compare Tab. 8.4).

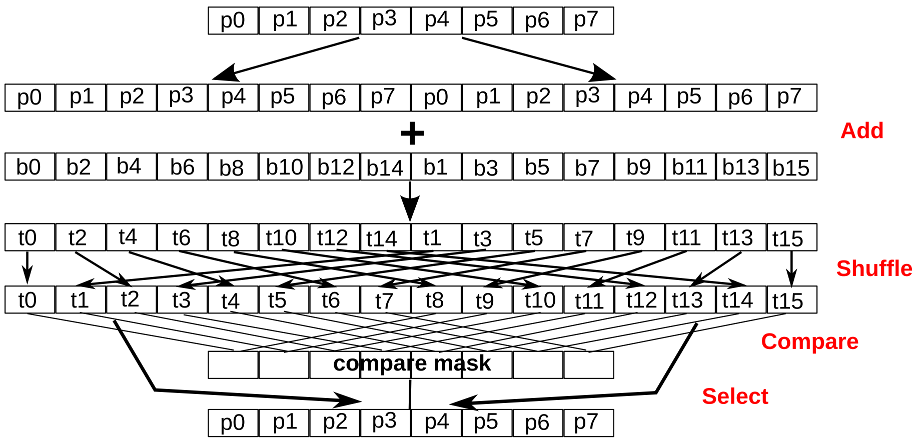

p0 p1 p2 p3 p4 p5 p6 p7
p0 p1 p2 p3 p4 p5 p6 p7 p0 p1 p2 p3 p4 p5 p6 p7
+
Add
b0 b2 b4 b6 b8 b10 b12 b14 b1 b3 b5 b7 b9 b11 b13 b15
t0 t2 t4 t6 t8 t10 t12 t14 t1 t3 t5 t7 t9 t11 t13 t15
Shuffle
t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15
Compare
compare mask
Select
p0 p1 p2 p3 p4 p5 p6 p7

Figure 8.19: ACS butterfly with SPE instrinsics [Ibi08].

8.4.3 Hybrid Iterative Reception

An iterative receiver is obtained by concatenating algorithms from the previous subsections: MMSE channel estimation (1D Wiener Filter), MIMO MMSE stream separation (Greville-based) and demapping, QRD-M MIMO demapping and Turbo decoding.

To achieve realtime throughput for a broadband configuration, a 'mini' Turbo receiver with (only) one outer iteration, a small number of turbo decoder iterations and a small M is used.

In the first demapping step, there is no apriori information from the decoder and therefore MMSE demapping is performed (as proposed in [Zim07]) — which has a very low complexity and performs well at low SNR. The LLRs obtained by (Max-Log) soft demapping are then improved by two Turbo decoder iterations. After that, another demapping step is run using breadth first tree search with the M-algorithm (with M=3), taking the obtained information from the decoder as apriori information into account. After the M-demapper, four turbo decoder iterations are run.

The granularity of loop unrolling and parallelization in most algorithms is one physical resource block (12 subcarriers times 7 OFDM symbols). Multi-user MIMO with different modulation levels on the same resources is supported.

Performance of this receiver is illustrated for QPSK, 16QAM and 64QAM transmission over 4x4 uncorrelated Rayleigh fading (for perfect synchronization and channel estimation) in Fig. 8.20. Performance would of course increase with larger *M*, larger number of Turbo decoder iterations and larger

Function	Benchmark result
2048-FFT	118807 transform/s (3,4 MAC/-cycle)
Channel, SNR, CFO estimation, MMSE equalization, CFO compensation	142,6 Msymbol/s (78.52 MAC/symbol)
Resource block demapping, soft demodulation (for 64QAM)	214,7 Msymbols/s
maxlogBCJR decoder	22.8 cycles/uncLLR, 140.0 MuncLLR/s
QPP (de)interleaver	2.6 cycles/LLR, 1210.2 MLLR/s
QRD-M algorithm, 4x4 QPSK M=3	82.4 cycles/LLR; 38.75 MLLR/s
QRD-M algorithm, 4x4 QPSK M=8	259.1 cycles/LLR; 12.32 MLL-R/s
QRD-M algorithm, 4x4 16QAM M=3 (no set partitioning)	129.9 cycles/LLR; 24.58 MLL-R/s
QRD-M algorithm, 4x4 16QAM M=8 (no set partitioning)	616,7 cycles/LLR; 5.18 MLLR/s

Table 8.4: Benchmark results on one SPU on CAB [IKWB09].

number of outer iterations.

In the presented implementation, the QRD-M detector for QPSK with M=3 is computationally almost as expensive as 2 Turbo decoder iterations. The computational effort for 16QAM with M=8 would correspond to 12 turbo decoder iterations.

On the other hand especially the M-demapper code offers room for improvement: it supports variable number of antennas, variable *M* and variable modulation size. The demapper code could be accelerated by hard coding a special case (like 4x4 QPSK, M=3), which would allow for optimized register usage. The complexity of the M demapper could also be reduced for higher-level modulation by using modulation set partitioning (with a slight accuracy degradation).

Processing 10 MHz in this configuration requires at least 8 SPEs running at 2.8GHz, processing 20MHz requires around 13.

8.4.4 Medium Access Control and Radio Link Control

The protocol stack functionality implemented on the NPU allows for adaptive multi-user transmission with fair scheduling (adaptive to channel qualities and bandwidth demands). Towards the physical layer, a physical downlink shared channel (PDSCH), physical uplink shared channel (PUSCH), physical downlink control channel (PDCCH), and physical uplink control channel (PUCCH) are currently present. On the MAC layer, transport blocks (data) are transferred between MAC entities via the downlink shared channel (DL-SCH) and uplink shared channel (UL-SCH). Finally, each user has (at least) one logical

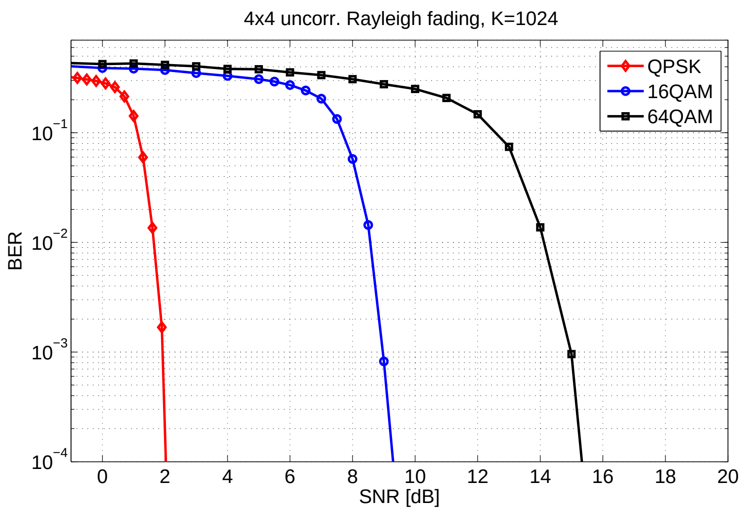


Figure 8.20: Accuracy of hybrid iterative receiver over 4x4 uncorr. Rayleigh channel with perfect channel estimation [IKWB09].

channel, the dedicated traffic channel (DTCH), in both the uplink and the downlink to carry user traffic between RLC entities. The control channels (PDCCH and PUCCH) are used for scheduling requests and grants and channel quality feedback information.

The control plane schedules user data and processes the control channels. The scheduler adapts to channel conditions based on CQI feedback data from the user terminals. The management plane allows clients to connect to the live protocol stack via TCP/IP, e.g. to view the measured data rate for each user.

Data plane packets are assigned headers with logical channel identifier (connection identifier), ARQ sequence number for repeat requests and reordering, as well as length and framing bits for fragmentation and packing/concatenation (to form transport blocks of adequate length chosen by the scheduler, and to allow data reconstruction).

The software is implemented in three branches: one is the base station protocol stack (including e.g. the MAC layer scheduler), the second one is the terminal protocol stack, the third is a multi-terminal realtime emulator for testing of the base station implementation.

The software further has two modes: in the normal mode, transmit/receive queues (with in-band control for the PHY) in DRAM are exported to PCIe address space and can be written by the Cell. In standalone testing mode, instead of the PHY connection the physical channels are tunneled over Ethernet beween MAC entities (physical channels are then separated by different ethertype field entries in the Ethernet tunnel header).

The data plane is functionally decomposed into a pipeline of microengine threads. Control and management plane are implemented (multi-threaded) in Linux user space on the XScale.

Block	ME	Worst Case Line Rate	Req. Parallel Threads
Network RX	0	550 Mbps	2
QM Enqueue	0	504 Mbps	2
Segmenter/QM Dequeue	1/0	401 Mbps	3
PHY TX	1	369 Mbps	4
PHY RX	2	550 Mbps	2
Reordering	2	360 Mbps	2
Reassembly	3	213 Mbps	1
Network TX	3	369 Mbps	4

Table 8.5: Estimated worst case line rate for functional blocks [HIK09].

Design and Implementation

Performance considerations The multithreaded architecture of the microengines allows to divide the data plane into a set of independent tasks. For example, one stage receives IP packets, the next stage inserts them into queues, and a third stage forms packets of adequate size for transmission based on a scheduling decision (segmentation, concatenation, fragmentation). To calculate the processing time available to each stage, first the packet arrival rate is defined. Towards the network side, Ethernet payload size can vary from 46 bytes to 1500 bytes, leading to a total of 84 to 1538 bytes per frame including preamble and interpacket gap. With 1 GBit line rate, maximum arrival rate with respect to packet size varies between $1/670ns$ and $1/12\mu s$. This corresponds to a maximum number of 603 or 11070 cycles per stage respectively [HIK09].

The second major performance consideration is that of memory bandwidth. The microengines have shared access to three types of memory on the Double Espresso board with varying capacities and access times (compare Fig. 8.10 on page 161). 512MB of DRAM are intended to hold packet buffers, 8MB of SRAM store packet handles and other frequently used data, and 16KB of fast scratch memory primarily support interthread communication.

Data Plane Functional Blocks and Thread Assignment The data plane's tasks are broken down into a set of processing stages to be implemented on the microengines. An estimate of the worst-case line rate for each block is given in Table 8.5 based on memory access time measurements, and the required parallel thread count is mainly determined by dividing 1000 Mbps by the worst-case line rate. The blocks communicate by scratch rings (circular FIFOs) except where noted. Threads are distributed as evenly as possible among the microengines and as dictated by use of the next neighbor bus, which allows efficient transfer of data from microengine N to microengine $N + 1$.

Ethernet Drivers The gigabit Ethernet port used for base station communication with the network requires RX/TX drivers which transfer network data to and from the microengines. The source code for a simple pair of drivers is presented in [JK03] and was adapted: in particular the TX driver was modified to deal with the quadword alignment present in the DRAM where packet data is stored, enabling the transmitter to efficiently access user packet data segmented at byte boundaries. In standalone mode, a second gigabit Ethernet port is also used.

Queue Manager User data sourced from the Ethernet receiver needs to be buffered prior to scheduling. The queue manager block also performs packet classification according to IP address and DiffServ type field for QoS. The queue manager maintains a packet queue in SRAM for each logical channel, and provides packets to the segmenter when requested. Separate threads are implemented for enqueue and dequeue operations, and a table of current queue sizes is maintained in shared memory for access by the scheduler. Enqueue and dequeue requests are transported by scratch rings, while dequeue replies are delivered to the segmenter by the next neighbor bus to reduce load on the scratch buses. Because the SRAM controller handles a limited number of simultaneous FIFOs, the CAM of the microengines is used as a cache to track which queues are currently active and to swap them in and out of the controller as needed.

Segmenter Segmentation (also concatenation and fragmentation) is performed on user packets based on a table of transport block sizes maintained by the control plane. The table is double buffered in shared scratch memory, allowing the control plane to write one buffer while the data plane reads the other. A microengine timer triggers the construction of a transport block for each user with schedulable data once per TTI (one millisecond). For each user, the segmenter requests packets from the queue manager until it has sufficient data to build a PDU. An Ethernet and MAC/RLC header is written and sent to the transmitter, followed by the packets (SDUs) to be included in the PDU. The final packet is segmented if necessary to fulfill the size requirement, with remaining data buffered by the segmenter for the user's next PDU.

Reordering As data is received, it is passed to the reordering block to correct for any out-of-order PDUs. A 512 PDU reception buffer for each user is maintained in SRAM, and the reordering timer is implemented using a microengine timer. Because most PDUs are expected to arrive in order, the block is optimized for that case, and the worst-case line rate of 360 Mbps given in Table 8.5 is correspondingly improbable. For this reason, only one or two threads are needed for gigabit line rate. This block also performs demultiplexing of data and control channels (packets).

Reassembly Reordered PDUs have their SDUs reassembled into IP packets, undoing the operation of the segmenter. If SDUs are segmented across multiple PDUs, they must be buffered by the reassembler block until all SDUs belonging to the packet are received. The reassembler maintains a set of FIFO queues for incoming PDUs quite similar to the queue manager. However, user packets at higher data rates will have been segmented into only a few SDUs. For this reason, the reassembler buffers packet handles in local memory before resorting to the much more expensive SRAM FIFO access, giving the block a worst-case rate of nearly 2 Gbps for largely unsegmented data. Highly segmented packets requiring the SRAM FIFO will have been scheduled at a lower rate, making the slower memory access irrelevant.

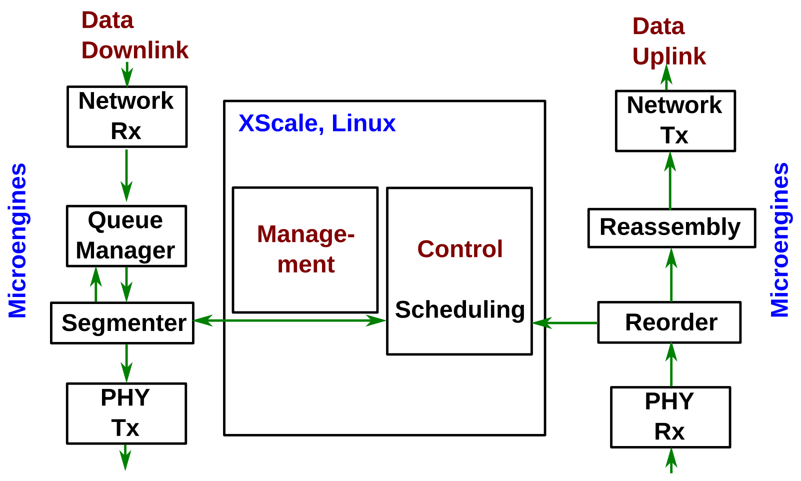


Figure 8.21: Block diagram of downlink and uplink protocol processing [HIK09].

Control and Management Plane The control and management plane are implemented as a Linux userspace application on the XScale processor with separate threads for scheduling, housekeeping and management. A custom kernel module receives timing interrupts from the microengines and signals the control plane when a new scheduling decision is to be made.

The scheduling algorithm currently implemented for OFDMA scheduling is a simple proportional fairness heuristic, evenly dividing the physical resource blocks of the system among users with schedulable data. The average channel quality in frequency direction for a user's PRBs determines the modulation index, which then gives the number of bits per PRB based on a lookup table. The size of the user's transport block is thus determined and passed to the segmenter. Uplink scheduling takes place through scheduling requests and grants via control channels. The XScale's control plane is responsible for constructing scheduling/CQI control packets, which are then transmitted and received by the microengines through a high priority scratch ring. For testing purposes, the control plane offers to manually override

CQI values for uplink and downlink and to schedule dummy traffic (using the client GUI over network).

The Linux kernel currently used here limits the control plane to responding to scheduling interrupts every 10 ms. A scheduling granularity of 1 ms (1 TTI) is supported by making decisions for ten TTI at once (into the future) — with the control plane latency still being 10 ms.

Implementation Results

Data plane goodput The data plane was set to multi-terminal emulation and tunnel mode on one side of the tunnel, and the GUI was used to manually adjust the transport block size for each user. On the other side of the tunnel the base station stack was running in tunnel mode. Traffic was generated and goodput measured using iperf [ipe], a bandwidth measuring tool, in TCP mode. It was verified that the goodput of any connection can be independently adjusted (tested with 480 kbps to 15.2 Mbps) by varying the transport block size (from 60 to 1900 bytes, one block per connection and TTI). Up to fifty simultaneous connections were active while performing this test. The maximum speed of the data plane was tested by setting the transport block size to 1900 for all users and starting multiple instances of iperf. The highest measured goodput of the data plane is approximately 550 Mbps. Above 500 Mbps, the data rate begins to fluctuate significantly.

Microengine utilization and memory bus load For a processor load, the data plane was simulated in Developer Workbench and performance statistics gathered for a period of five TTIs, or five milliseconds. The threads involved in segmentation (ME0/ME1) were sent IP packets at gigabit line rate using the packet generator, and the tunnel packets transmitted from the segmenter were then logged. These logged packets were then fed to the threads involved in reassembly (ME2/ME3), simulating tunnel traffic from five TTIs. To represent the worst case in terms of processing time per packet, 60-byte user packets were used in the simulation. These were formed into 1500-byte PDUs, and 20% of the tunneled PDUs were set to arrive out of order, thus also testing the reordering block. Figure 8.22 shows the thread execution time. Most of the threads demonstrate relatively little activity with the exception of the segmenter and reassembler. The total load on each microengine is illustrated in Fig. 8.23. With the load below about 50% in all cases, the microengines show plenty of potential for increasing data plane speed and incorporating additional functions. The memory bus usage further backs this claim. Fig. 8.23 also shows the bus utilization for the three main types of data plane memory. In all cases the usage is under 20%, again indicating a possibility for expansion.

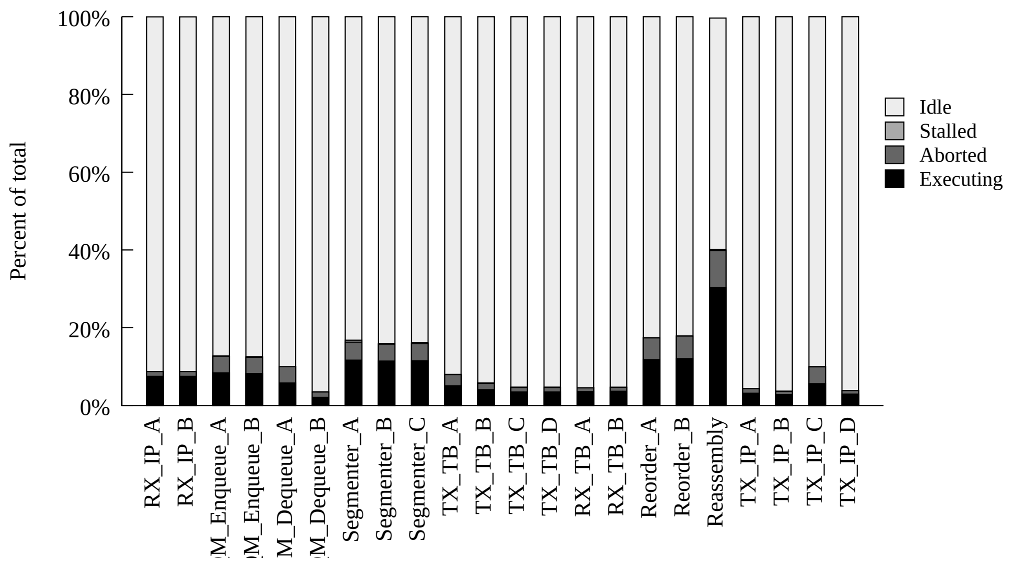


Figure 8.22: Microengine thread execution time for 60-byte IP packets and 1500-byte transport blocks [HIK09].

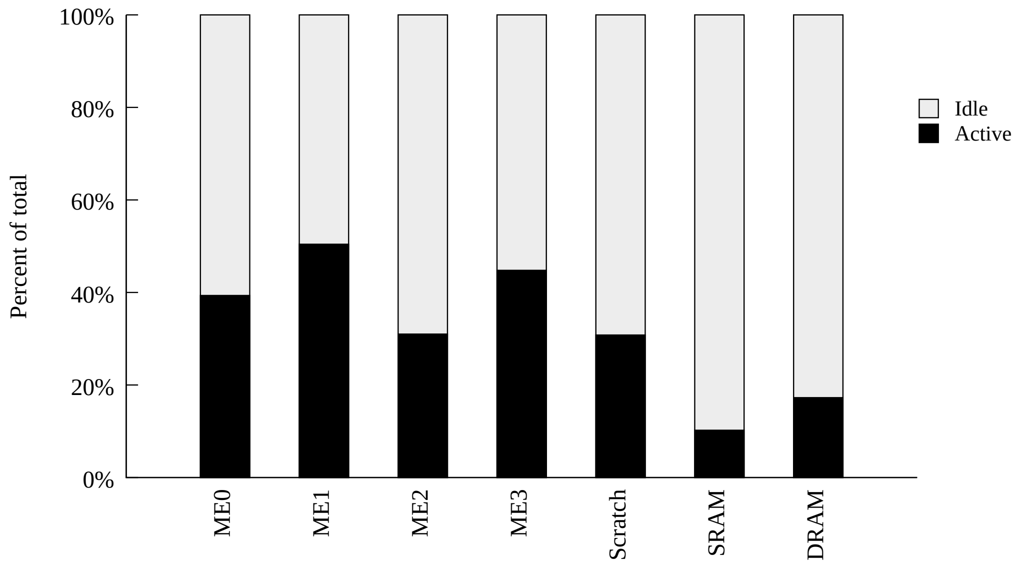


Figure 8.23: Microengine and memory bus load [HIK09].

8.4.5 Testing and Visualization

Rather than dealing with software test cases, this subsection describes some demonstrator test setups.

Base station protocol test, MAC/RLC GUI A setup for base station protocol test in one host is illustrated in Fig. 8.24. One NPU on the 'Double Espresso' board is running the base station stack and the other is running multi-terminal emulation, both in tunnel mode. An Ethernet switch is also contained in the host chassis and connected in the middle of the tunnel. Tunnel traffic carries the broadcast destination address, so that the host can sniff all packets on a third port.

For protocol stack visualization, status and control, a server is running on the XScale. It provides queue fillings and packet counts, channel feedback values and throughput measurements. A client was written in TCL/TK and run on the host. It allows to manually override scheduling parameters (CQI values in uplink and downlink and ressource allocations per logical channel) and also to schedule dummy traffic. For an impression of the GUI compare Fig. 8.1 on page 151, where also the switch is visible. To visualize scheduler operation, traffic is generated with iperf on two external computers (assigning multiple IP addresses to its NIC for multiple connections).

Using the GUI CQI controls, it can be verified that the data rate of the connections can be varied by changing the channel conditions. By increasing the number of users with schedulable data either through actual traffic generation or by manually overriding the queue status, it can be observed that the goodput of the connections varies according to the number of scheduled users.

Correct operation of all physical channels (including the control channels) is visualized using the protocol analyzer software Wireshark [wir] on the host for the sniffed tunnel traffic with adequate filtering.

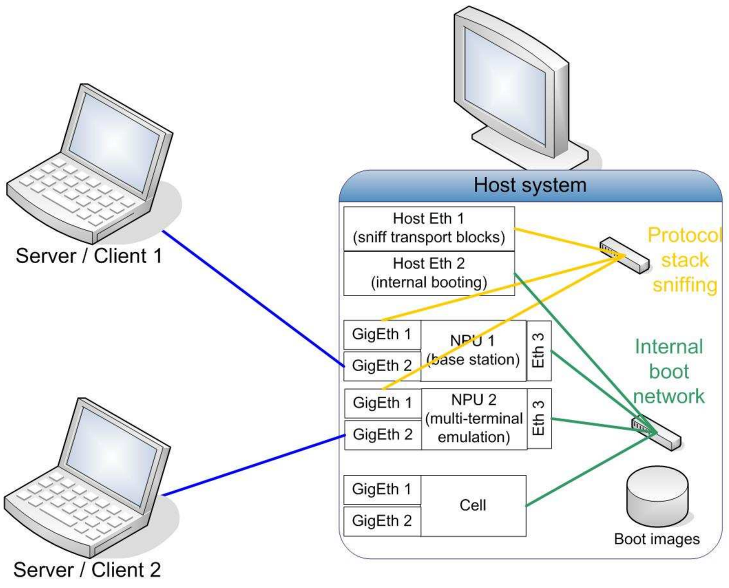


Figure 8.24: Base station protocol test in one host.

PS3 only test, PHY GUI To demonstrate only Cell software (PPE and SPE) without PCIe components, the PS3 can be used. It can either transmit and receive over the USRP frontend with USB connection, or a physical layer loopback through a base band channel emulated in software can be used for defined channel conditions. A dummy MAC layer is implemented on the PPE supporting one queue (one logical channel). A physical layer visualization GUI was written in TCL and using the *plplot* library. It is run on the PPE (the window can of course also be exported over the network). The GUI includes a time domain scope, sprectrum scope, scatter plot and LLR distribution plot. The PHY GUI allows for loopback test to vary modulation, number of turbo decoder iterations and noise power of a base band channel emulator (run in an SPE). To visualize the effect of PHY PER on a video codec, test videos were streamed using vlc [Vid] server and client.

Software channel emulation A realtime base band AWGN channel emulator was implemented on the SPE (single-precision float, 4xSIMD). It first generates uniform distributed random numbers from a 128bit register using *shift* and *xor* operations. The uniform distributed numbers are then mapped to Gaussian distributed numbers using the inverse distribution function — which is interpolated with line segments. Interpolation with 8 line segments is illustrated in Fig. 8.25. Gneration of the PRBS on an SPE on the CAB achieves 38909 MBit/s, and of the complex Gaussian noise 315Msample/s [Ibi08]. The SNR is adjustable, and the software channel is used for visualization with the PHY GUI or to measure BER/PER curves.

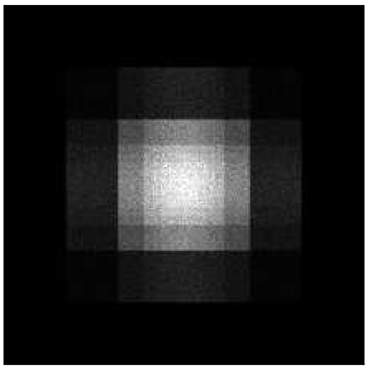


Figure 8.25: WGN density plot of realtime Software emulated channel, inversion method with 8 line segments [Ibi08].

FPGA channel emulation Since an FPGA is available in the platform anyway, it was also used as base band channel emulator – using the same PCIe interface as the RF frontend. It allows for the same tests and visualization as the software channel on SPE, but also includes the number conversion, PCIe bus, FPGA DMA controller and FIFO flow control into the loopback. The interface number format is 16 bit fixed-point, with symbol rate $n\cdot$ 125MHz (with n being variable, in SISO case $n = 2$). The implementation was pipelined to achieve the 125 MHz. Internally the emulator uses a 32 bit fixed-point representation of both data and noise for calculations [EKKI]. The interface offers 4 lanes PCIe (v1.1), with DMA controller in the FPGA. Signal and noise gain are adjustable (with SNR range more than 30dB) over registers mapped into PCIe adress space.

For WGN generation, first uniformly distributed numbers are produced by means of linear feedback shift registers (LFSRs, Galois type, see fig. 8.27), which are in turn transformed into Gaussian distributed samples using the inversion method:

$$U \sim \text{unif.} \quad \Rightarrow \quad F^{-1}(U) \sim F \tag{8.5}$$

If U is uniformly distributed, then $F^{-1}(U)$ has F as its CDF. Implementation uses a 128bit LFSR and linear interpolation with 14 line segments of the inverse CDF (which results in a step-wise constant interpolation of the PDF, since the PDF is the CDF's derivate). The difference between target and approximating function is an overall error of 5.2% and variance error of 1.8%. Doubling the number of interpolation line segments roughly halves the error. Instantiation including DMA transfer logic and FIFO registers for the PCIe interface on the Xilinx Virtex 5 FPGA of the ML-555 board (more specifically an xc5vlx50t) consumes roughly 30% of its slices and 20% of the DSP48 blocks (multipliers for gains and Gauss transformation). The resource consumption on the target FPGA (xc5vsx95t) in the radio frontend uses less than half its slices.

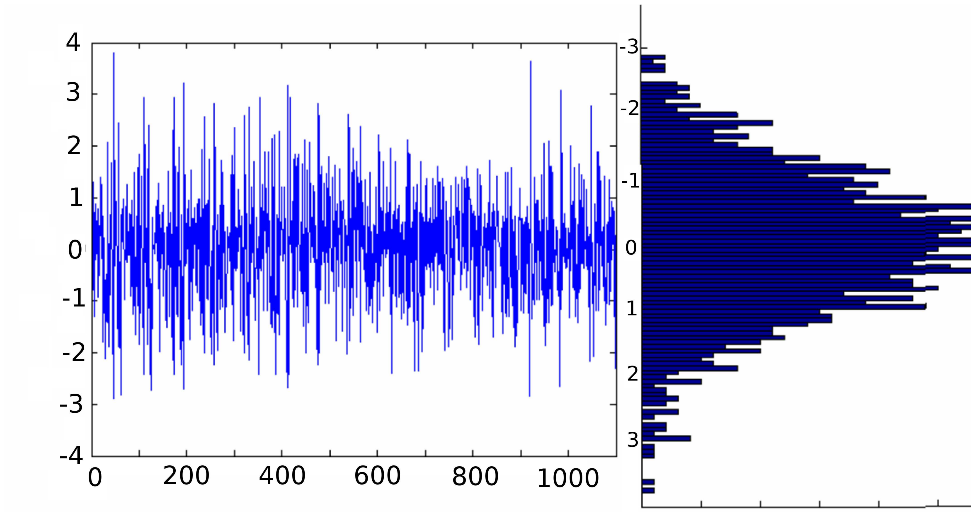


Figure 8.26: Channel emulator noise samples and their probability density [EKKI].

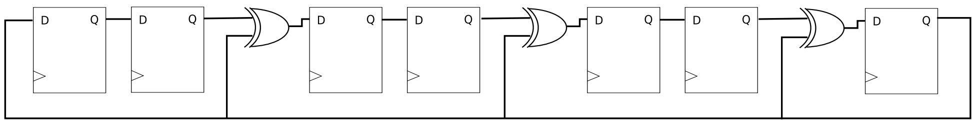


Figure 8.27: 7 bit Linear Feedback Shift Register (clock connections not shown) [EKKI]. Implementation uses 128bit LFSR.

8.5 Suitability for Further Developments

8.5.1 Distributed Physical Layer

Motivation The approach is to increase the number of base station antennas serving a user by also using the antennas from neighbouring cells. Intercell interference is countered by multi-cell joint detection (and possibly joint transmission) or intercell interference cancellation (compare Sec. 1.5). The approach is particularly interesting for cell-edge users, where the received energy almost equally splits between the neighbouring cells. High-throughput low-delay connections between base stations are assumed. Distributed physical layer processing can be interpreted as an extension of remote antenna heads or of the soft handover applied in 3G systems. Remote antenna heads offer to connect antennas to base stations over a longer distance, where digital base band samples are transmitted over the interface [obs, cpr]. Distributed physical layer processing is also called cooperative transmission/reception, network MIMO or coordintated multi-point transmission/reception. Joint processing of multiple base stations' and/or relays' signals is in principle applicable in uplink and downlink.

Signal processing issues Signal processing issues with this approach concern channel estimation and synchonization.

The MIMO pilot grid in a sector is designed to have no intra-cell interference on pilot symbols (compare Sec. 1.4). For intercell channel estimation with the same grid, interference cancellation has to be performed on pilot symbols (this relates to MIMO APP channel estimation, compare Sec. 3.1.3). The alternative of adapting the pilot grid to the intercell setup (containing transmit antennas from neighbouring cells) runs into the pilot overhead problem: when increasing the number of MIMO transmit antennas with constant pilot density per antenna, the pilot symbol overhead (counting pilot symbols and zero symbols at other antenna's pilot locations) grows quadratically, while the sum of available symbols (pilots + data) only grows linearly. This leads to the tradeoff of pilot overhead versus potential multiplexing gain (or here interference cancellation gain) – limiting the number of MIMO transmit antennas.

For synchronization, propagation delay differences from dislocated transmitters cause timing offsets. Timing advance (like in normal uplink, Sec. 1.4) can only be applied with respect to one receiver. The OFDM cyclic prefix length is chosen to cope with delay spread — which is now increased by propagation delay difference. With symbol rate T_S and speed of light c, one sample on the air has length c/T_S, and a distance of s translates into $s/(cT_S)$ samples delay. For OFDM signal processing at the receiver, the cooperation area is limited by the sum of delay spread (regarding one of the transmitters) and propagation delay difference: intersymbol interference is inherent if this sum exceeds CP length [IM08]. This leads to the tradeoff of CP overhead versus potential multiplexing gain (or here interference cancellation gain). Compare Fig. 8.28 and Fig. 8.29 for an illustration.

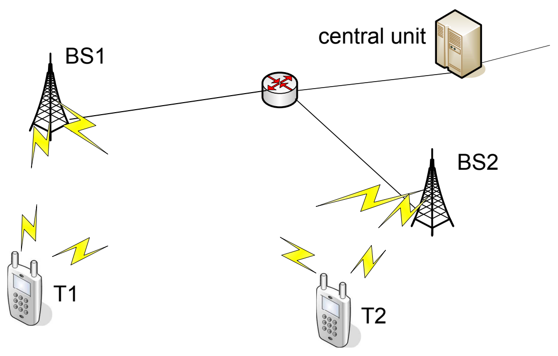


Figure 8.28: Example scenario of 2x2 cooperative MIMO-OFDM: two base stations with one antenna each (BS1 and BS2) are jointly processed as a virtual station with two antennas in uplink and downlink. A terminal T1 with two antennas receives both data streams, but has to cope with delay differences [IM08].

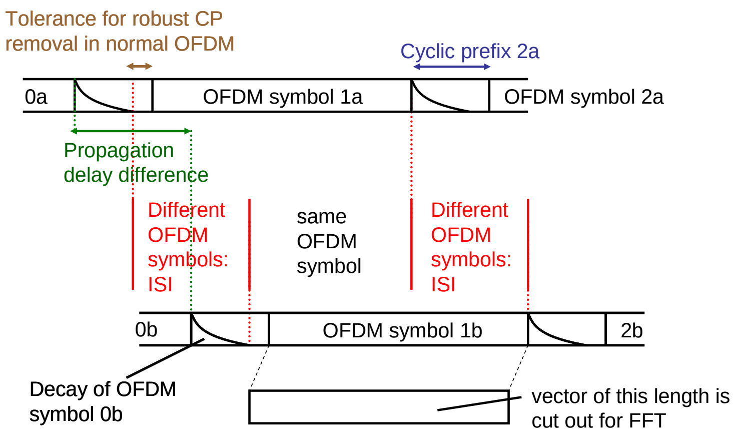

Tolerance for robust CP removal in normal OFDM
Cyclic prefix 2a
0a
OFDM symbol 1a
OFDM symbol 2a
Propagation delay difference
Different OFDM symbols: ISI
same OFDM symbol
Different OFDM symbols: ISI
0b
OFDM symbol 1b
2b
Decay of OFDM symbol 0b
vector of this length is cut out for FFT

Figure 8.29: Superposed receive sample streams of two transmit antennas from different transmit stations. In this case the cyclic prefix is long enough to guard against channel delay spread, but not long enough to guard against delay differences. The result is intersymbol interference between the data streams of the transmit antennas [IM08].

System architecture Since network-wide central processing is not feasible, the question of choice of cooperation areas is raised. Joint processing wants to include the desired signal(s) and the strongest respective interferers. The base station grid with its antenna directions influences receive power from different transmitters. The areas of joint processing can be either chosen statically (predefined) or dynamically. [MC11] selects clusters of cooperating base stations based on received signal strengths. Static predefined cooperation areas are described e.g. in [IJ08, WY10], discussing the possiblity for joint selection of cooperating stations, antenna directions and MIMO pilot grid. Dynamic selection of possibly overlapping cooperation areas is discussed e.g. in [IHJ08, RCP10].

Network requirements The network requirements between base station connections in terms of throughput and delay largely increase for base band sample exchange, in dependence on the chosen system architecture. A very rough estimation for static cooperation areas is given in [IJ08]. For 'distributed joint processing', there is a choice between iterative sample transport and redundant computation in different stations. Available network infrastructure may require the system architecture to work scaleably with variable bandwidth. A multicast sample exchange protocol for dynamic overlapping cooperations areas is drafted in [IHJ08].

Computational requirements Another consequence from the chosen system architecture is an increased computational requirment for the base stations: computational power has to increase to perform multicell joint detection or sequential intercell interference cancellation. A very rough estimation for static cooperation areas assuming linear MIMO stream separation is given in [IJ08].

Static cooperation areas Central processing (compared to distributed processing) for cooperation areas minimizes network and computational load. Static cooperation areas for piecewise central processing are illustrated in Fig. 8.30, based on a setup with 120 degree sectorization. The same base station positions are used as in the standard hexagonal grid, but rotating sectorization by 90 degrees (Fig. 8.30). The resulting hexagonal cells are 3 times larger in area. Before, there were 3 cells (sectors) per base station, with the mapping there is one cell per base station. In the new hexagonal grid, a terminal normally receives strong signals from three base stations. The antennas offering most performance gain when cooperating thus belong to three sectors each from a different base station. Each location in the system can be mapped to such three sectors of three different base stations. Another mapping without rotating the base stations and with the same old cell size can be done by introducing more base stations (Fig. 8.32). The same joint processing approach can be used. The whole area covered by the system is partitioned into disjunct areas, each consisting of three sectors belonging to three different base stations. Joint signal processing is applied to each such three-sector-area. The resulting partitioning is depicted in Fig. 8.31. The processing for a service area is located in one base station. In Fig. 8.31 the processing for service area *m* is located in base station *m*. Between the base station and its two remote sectors (hosted by two neighbouring base stations) base band samples are exchanged for uplink and downlink. Between neighbouring service areas the well-known approaches of interference mitigation can of course still be applied. Now there are two types of Sectors: one base station still handles data, protocol and signal processing for three sectors, but for three different ones, of which two are remote. Signal processing for the three sectors can now be done jointly to reduce inter-cell interference (before, the three local sectors were separated by sectorized antennas). Each base station now offers two sectors and their network connection for base band sample exchange to other base stations and jointly processes signals from the third and two remote sectors.

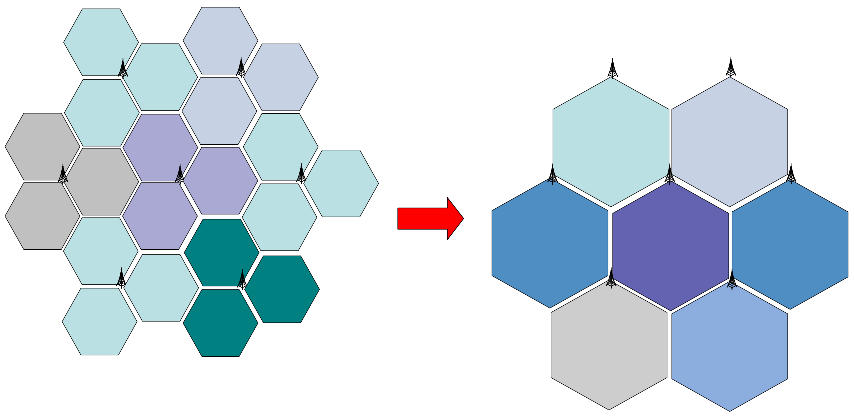


Figure 8.30: The current hexagonal grid (left) can be modified into a new hexagonal grid with exactly the same base station positions (right) by rotating sectorization 90 degrees, suited for piecewise joint processing [IJ08].

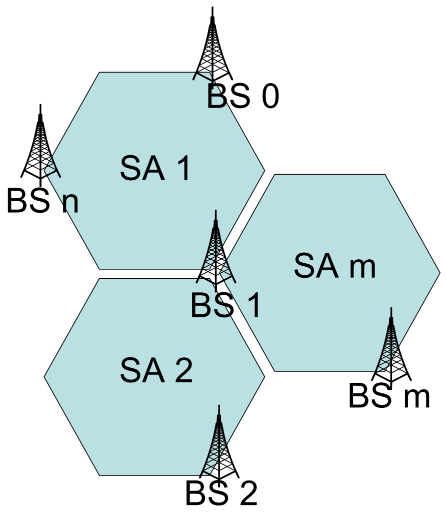


Figure 8.31: Mapping joint signal processing areas into the hexagonal grid: signals in area 'SA m' are jointly processed by base station 'BS m'. Each base station processes samples from one local and two remote sectors, and offers the other two local sectors for remote processing [IJ08].

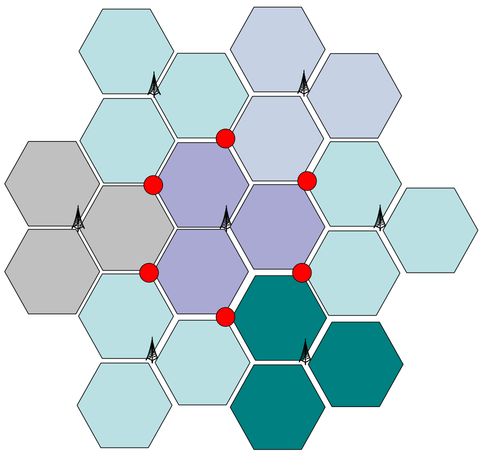


Figure 8.32: This architecture can also be implemented with the current cell size and without rotating sectorization, by introducing more base stations (red dots) [IJ08].

Dynamic cooperation areas Dynamic cooperation areas can be formed in uplink, when base stations dynamically request neighbouring base stations's sample streams to aid the own computation. Sample streams can possibly be requested for certain subbands (in frequency domain). Protocol-wise this can be seen as dynamic join / leave of multicast groups [IHJ08]. Computation in this architecture is partly redundant (if sample streams are mutually requested), because there is no actual central unit. Overlapping areas of joint detection are formed, where each base station is a virtual central unit. This architecture is scaleable in terms of network bandwidth (heterogeneous backhaul) and computational power. Groups of cooperating stations can dynamically form according to the actual interference situation.

Testbed suitability For high-rate base band sample exchange with the demonstrator platform, a PCIe 10BGit Ethernet interface card can be used, e.g. the Myri10G card [Myr] (compare Fig. 8.2 on page 152). It offers a fiber connection with pluggable XFP interface, has an x8 PCIe connector and is supported by Linux. Samples can be streamed over UDP/IP. With interrupt coalescence (not for every received packet an interrupt) and checksum offload from the host into the NIC, more than 9.9GBit/s througput were measured. An example 12x12 configuration for channel estimation and stream separation on the Cell (e.g. for three sectors with 4 antennas) is covered by Fig. 8.11 on page 162 and in [IKK+08].

Abbreviations

3GPP	3rd Generation Partnership Project.
APP	A Posteriori Probability
CSI	Channel State Information.
FDD	Frequency Division Duplex
FIR	Finite Impulse Response.
ICI	Intercarrier Interference
ISI	Intersymbol Interference.
LS	Least Squares
LTE	Long Term Evolution.
LLR	Log Likelihood Ratio
MAP	Maximum A Posteriori
MIMO	Multiple Input Multiple Output.
MISO	Multiple Input Single Output.
ML	Maximum Likelihood
MMSE	Minimum Mean Square Error.
MSE	Mean Square Error.
OFDM	Orthogonal Frequency Division Multiplexing.
PDF	Probability Density Function.
QAM	Quadrature Amplitude Modulation.
SER	Symbol Error Rate.
SIC	Successive Interference Cancellation.
SINR	Signal to Interference plus Noise Ratio.
SISO	Single Input Single Output.
SNR	Signal to Noise Ratio.
TDD	Time Division Duplex.
WiMAX	Worldwide Interoperability for Microwave Access.
WLAN	Wireless Local Area Network.

Publication List

- A. Ibing and H. Boche. On Predicting Convergence of Iterative MIMO Detection-Decoding with Concatenated Codes. *IEEE Trans. Veh. Techn.*, 59(8):4134-4139, 2010.
- A. Ibing, D. Kühling, and H. Boche. Reliability-Based Hybrid MMSE/Subspace-Max-Log-APP MIMO Detector. *IEEE Commun. Lett.*, May 2010.
- A. Ibing and H. Boche. Automatic Joint Optimization of Iterative MIMO-OFDM Receiver Algorithms on a Meta Level. *IEEE Int. Conf. Commun. 2011*, in press.
- A. Ibing, P. Otto and H. Boche. On Channel Correlation Based Scheduling and Signalling for MIMO-OFDMA Downlink. *IEEE 73rd Vehicular Technology Conf., 2011*, in press.
- K. Khan, A. Ibing and D. Dahlhaus. Joint Model for Fine Synchronization and Adaptive LMMSE Channel Estimation in Uplink OFDMA. *7th Int. Symp. Wireless Communication Systems, 2010*.
- H. Li and A. Ibing. On Complexity-Reduced Implementation of Multi-Dimensional Wiener Interpolation Filtering. *IEEE 72nd Vehicular Technology Conf., 2010*, in press.
- A. Ibing, D. Kühling, and H. Boche. On the Relation of MIMO APP Detection and SIMO Maximum Ratio Combining. *IEEE Global Commun. Conf. 2009*.
- A. Ibing, D. Kühling, D. Wieruch, and H. Boche. Software Defined Hybrid MMSE/QRD-M Turbo Receiver for LTE Advanced Uplink on a Cell Processor. *IEEE Int. Conf. Commun. 2009*.
- X. Zhao and A. Ibing. Performance Evaluation of a Low-Complexity LTE Base Station Receiver. *7th Int. Workshop Multicarrier Syst. Solutions, 2009*.
- C. Holmes, A. Ibing, and D. Kühling. On Network Processor based Protocol Stack Implementation for 4G Base Stations. *Wireless Telecommun. Symp. 2009*.
- A. Ibing, D. Kühling, M. Kuszak, V. Jungnickel, and C. Helmolt. Flexible Demonstrator Platform for Cooperative Joint Transmission and Detection in Next Generation Wireless MIMO-OFDM Networks. *4th Int. Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities, 2008*.
- D. Kühling, A. Ibing, and V. Jungnickel. 12x12 MIMO-OFDM Realtime Implementation for 3GPP LTE+ on a Cell Processor, *14th European Wireless Conf., 2008*.
- D. Kühling and A. Ibing. A Novel Low-Complexity Algorithm for Linear MMSE MIMO Receivers. *Int. Symp. Wireless Pervasive Computing, 2008*.
- K. Manolakis, Andreas Ibing, and Volker Jungnickel. Performance Evaluation of a 3GPP LTE Terminal Receiver. *14th European Wireless Conf., 2008*

- A. Ibing and K. Manolakis. MMSE Channel Estimation and Time Synchronisation Tracking for Cooperative MIMO-OFDM with Propagation Delay Differences. *5th Int. Symp. Wireless Communication Systems, 2008*.
- D. Wieruch, D. Kühling, and A. Ibing. 250MBit/s 64-State Soft-Decision Viterbi Decoder on a Cell CPU. *Mikroelektroniktagung 2008*.
- A. Ibing, X. Zhao and D. Kühling. Discontinuous Time Tracking for OFDMA Uplink. *13th Int. OFDM Workshop, 2008*
- A. Ibing and V. Jungnickel. Joint transmission and detection in hexagonal grid for 3GPP LTE. *Int. Conf. Information Networking, 2008*.
- D. Kühling, S. Schiffermüller, and Andreas Ibing. On Efficient Computation of MMSE Equalizer Matrix for MIMO-OFDM Systems. *Workshop on Commercial MIMO-Components and -Systems 2007*.
- A. Ibing and H. Boche. Fair OFDMA Scheduling Algorithm using iterative local search with k-opt switches. *IEEE Wireless Commun. Networking Conf., 2008*.
- A. Ibing, Y. Hadisusanto, and V. Jungnickel. Scalable Network Multicast for Cooperative Base Stations. *2nd Int. Conf. Commun. Syst. Software and Middleware, 2008*.
- A. Ibing. Software Defined Radio Testbed. *invited talk at BMBF Statusseminar Mobilkommunikation 2008*
- A. Ibing and H. Boche. Fairness vs. Efficiency: Comparison of Game Theoretic Criteria for OFDMA Scheduling. *Asilomar Conf. Signals, Systems and Computers, 2007*.
- V. Jungnickel, M. Schellmann, A. Forck, H. Gäbler, S. Wahls, A. Ibing, K. Manolakis, T. Haustein, W. Zirwas, J. Eichinger, E. Schulz, C. Juchems, F. Luhn, and R. Zavrtak. Demonstration of virtual MIMO in the Uplink. *IET Smart Antennas and Cooperative Communications Seminar, 2007*.
- A. Ibing and V. Jungnickel. On Hardware Implementation of Multiuser Multiplexing for SC-FDMA. *IEEE 66th Vehicular Technology Conf., 2007*.
- M. Schellmann, K. Manolakis, A. Ibing, and M. Kuszak. Impact of the Preamble Bandwidth on the Synchronization Performance. *12th Int. OFDM Workshop, 2007*
- T. Farnham, A. Gefflaut, A. Ibing, P. Mähönen, D. Melpignano, J. Riihijärvi, and M. Sooriyabandara. Toward Open and Unified Link-Layer API. *IST Mobile and Wireless Summit 2005*.
- C. Hoymann, A. Ibing, and I. Forkel. MAC Layer Concepts to Support Space Division Multiple Access in Wireless Metropolitan Area Networks (IEEE 802.16a). *Wireless World Research Forum (WWRF) 2003*

- Weiterleitungsknoten und Endgerät für ein FDD-Kommunikationsnetzwerk und Verfahren diese zu betreiben. *European patent application EP 2 245 761 A1*, 16.01.2008. applicants: Nokia Siemens Networks, Fraunhofer Gesellschaft; inventors: R. Halfmann, T. Haustein, A. Ibing, W. Zirwas.
- Wireless telecommunication system including a base station, relay node and method for global fair scheduling. *European patent EP 2 066 084 B1; international application WO 2009/068413 A1*; 27.11.2007, holders: Nokia Siemens Networks, Fraunhofer Gesellschaft; inventors: R. Halfmann, T. Haustein, A. Ibing and W. Zirwas,
- Verfahren zum Steuern einer Relaisstation in einem Mobilfunknetzwerk. *German patent application DE 10 2008 009 087 A1*, 14.02.2008. applicants: Nokia Siemens Networks, Fraunhofer-Gesellschaft; inventors: A. Ibing, W. Zirwas, T. Haustein, R. Halfmann.
- Verfahren und Vorrichtung zur Datenübertragung, 05.02.2008, *German patent DE 10 2008 007 497*; holder: Fraunhofer Gesellschaft; inventors: W. Zirwas, J. Eichinger, A. Ibing, L. Thiele, V. Jungnickel and T. Haustein.
- Vorrichtung und Verfahren zum Empfangen eines Sendesignals. *German patent DE 10 2008 057 059 B3*. 13.11.2008. holder: Fraunhofer Gesellschaft; inventor: A. Ibing.
- Ein Detektor zur Detektion eines Eingangswortes. *German patent application DE 10 2009 041 175.5*, 11.09.2009. applicant: Fraunhofer-Gesellschaft; inventor: A. Ibing.
- Method and apparatus for frequency division multiple access transmission and reception. *European patent application EP 07010071.4-1237*, 21.05.2007; applicants: Nokia Siemens Networks, Fraunhofer Gesellschaft; inventors: T. Haustein, A. Ibing and V. Jungnickel.
- Device for and method of transmitting data packets. *European patent application EP 07015536.1-1237*, 07.08.2007; applicant: Nokia Siemens Networks; inventors: J. Eichinger, T. Haustein, A. Ibing, R. Inderst, K. Migge and W. Zirwas.
- Vorrichtung und Verfahren zur Datenübertragung per Mobilfunk, 05.07.2010, *German patent application DE 10 2010 027 434.8*; applicant: P. Otto; inventors: A. Ibing and P. Otto.
- Receiver. *US patent application US 12/923.534*, 27.09.2010, applicant: TU Berlin; inventors: A. Ibing and H. Boche.
- Ad hoc mobile devices and and hoc networks. *US patent application US 12/941.861*, 08.11.2010; applicant: TU Berlin; inventors: A. Ibing and H. Boche.
- Receiver. *US patent application US 12/843.721*, 26.07.2010. applicant: TU Berlin; inventors: A. Ibing and H. Boche

Bibliography

[3GP06] 3GPP. *TSG RAN: TR25.814v7.1.0 Physical layer aspects for Evolved Universal Terrestrial Radio Access*, Sept 2006. online http://www.3gpp.org/ftp/Specs/html-info/25814.htm.

[3GP07] 3GPP. *3GPP TSG RAN: TS36.211v1.2.0 Physical Channels and Modulation*, June 2007. online http://www.3gpp.org/ftp/Specs/html-info/36211.htm.

[3GP09] 3GPP. *3GPP TSG RAN: TS36.212v8.7.0 Multiplexing and Channel Coding*, May 2009. online http://www.3gpp.org/ftp/Specs/html-info/36212.htm.

[3GP10a] 3GPP. *3GPP TSG RAN: TS36.213v9.2.0 Physical Channels and Modulation*, June 2010. online http://www.3gpp.org/ftp/Specs/html-info/36213.htm.

[3GP10b] 3GPP. *3GPP TSG RAN: TS36.300v.10.0.0 E-UTRA and E-UTRAN; Overall Description; Stage 2*, June 2010. online http://www.3gpp.org/ftp/Specs/html-info/36300.htm.

[ABI03] T. N. E. Greville A. Ben-Israel. *Generalized Inveres*. Springer, 2nd edition, 2003.

[ABSS97] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54:317–331, 1997.

[AEVZ02] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Trans. Inf. Theory*, 48:2201–2214, 2002.

[AJ04] C. Athaudage and A. Jayalath. Delay-spread estimation using cyclic prefix in wireless OFDM systems. *IEE Proc. Commun.*, 151(6):559–566, December 2004.

[AS06] Y. Altun and A. Smola. Unifying divergence minimization and statistical inference via convex duality. In *Proc. Conf. Learning Theory*, 2006.

[Asu] Asus. *L1N64-SLI WS*. online http://www.asus.com/products4.aspx?modelmenu=2&model=1530&l1=3&l2=136&l3=486&l4=0 (visited in 2008).

[Asu09] Asus. *P6T7 WS SuperComputer Mainboard Manual*, 2009. online http://usa.asus.com/Product.aspx?P_ID=9ca8hJfGz483noLk (last visited on 24.01.2011).

[Aue09] G. Auer. 3d pilot aided channel estimation. In *IEEE Wireless Commun. Networking Conf.*, 2009.

[AW02] V. Abhayawardhana and L. Wassell. Common phase error correction with feedback for OFDM in wireless communication. In *IEEE Global Commun. Conf.*, 2002.

[BAS04] R. Budruk, D. Anderson, and T. Shanley. *PCI Express System Architecture*. Mindshare, 2004.

[BCJR74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theory*, 20:284–287, March 1974.

[BD87] P. Brockwell and R. Davis. *Time Series: Theory and Methods*. Springer, 1987.

[Bea03] M. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, May 2003.

[Bel63] P. Bello. Characterization of randomly time-variant linear channels. *IEEE Trans. Commun. Syst.*, 11(4):360–393, 1963.

[BHM+05] K. Berkel, F. Heinle, P. Meuwissen, K. Moerman, and M. Weiss. Vector processing as an enabler for software-defined radio in handheld devices. *EURASIP J. Appl. Signal Process.*, 16, 2005.

[Bis06] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BMDP98] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara. Soft-input soft-output modules for the construction and distributed iterative decoding of code networks. *European Trans. Telecommun.*, 9(2):155–172, 1998.

[Bon04] T. Bonald. A score-based opportunistic scheduler for fading radio channels. In *European Wireless Conf.*, 2004.

[BR70] F. Bauer and C. Reinsch. Inversion of positive definite matrices by the Gauss-Jordan methods. In J. Wilkinson and C. Reinsch, editors, *Handbook for Automatic Computation Vol. 2: Linear Algebra*, pages 45–49. Springer, 1970.

[BRG05] F. Brännström, L. Rasmussen, and A. Grant. Convergence analysis and optimal scheduling for multiple concatenated codes. *IEEE Trans. Inf. Theory*, pages 3354–3364, Sept. 2005.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BW07] H. Boche and M. Wiczanowski. Optimization-theoretic analysis of stability-optimal transmission policy for multiple antenna multiple access channel. *IEEE Trans. Signal Process.*, 55(6):2688–2702, June 2007.

[Car03] B. Carlson. *Intel Internet Exchange Architecture and Applications*. Intel Press, 2003.

[CCB95] P. Chow, J. Cioffi, and J. Bingham. A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels. *IEEE Trans. Commun.*, 43(2-4):773–775, 1995.

[CJ08] V. Cadambe and S. Jafar. Interference alignment and degrees of freedom of the k-user interference channel. *IEEE Trans. Inf. Theory*, 54(8):3425–3441, 2008.

[CJS$^+$10] J. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *Ann. Int. Conf. Mobile Comput. Networking*, 2010.

[CK80] I. Csiszar and J. Körner. Many coding theorems follow from an elementary combinatorial lemma. In *Czechoslovak-Soviet-Hungarian Sem. Inf. Theory*, pages 25–44, 1980.

[CK81] I. Csiszar and J. Körner. Graph decomposition: a new key to coding theorems. *IEEE Trans. Inf. Theory*, 27:5–20, 1981.

[CN95] I. Csiszar and P. Narayan. Channel capacity for a given decoding metric. *IEEE Trans. Inf. Theory*, 41(1):35–43, Jan. 1995.

[CNB$^+$08] J. Cheng, A. Nimbalker, Y. Blankenship, B. Classon, and T. Blankenship. Analysis of circular buffer rate matching for LTE turbo code. In *IEEE 68th Veh. Tech. Conf.*, 2008.

[Cos83] M. H. Costa. Writing on dirty paper. *IEEE Trans. Inf. Theory*, 29(3):439–441, May 1983.

[cpr] Common public radio interface. online http://www.cpri.info (last visited on 15.02.2011).

[CT65] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19:297–301, 1965.

[Czy96] A. Czylwik. Adaptive OFDM for wideband radio channels. In *IEEE Global Commun. Conf.*, pages 713–718, 1996.

[dJW02] Y. de Jong and T. Willink. Iterative tree search detection for MIMO wireless systems. In *IEEE 56th Veh. Tech. Conf.*, 2002.

[DPSB07] E. Dahlman, S. Parkvall, J. Sköld, and P. Beming. *3G Evolution: HSPA and LTE for Mobile Broadband*. Academic Press, 2007.

[Eat07] J. W. Eaton. *GNU Octave Manual*. Network Theory Limited, 3rd edition, 2007. online `http://www.network-theory.co.uk/docs/octave/`.

[EKKI] V. Eckert, M. Kuszak, D. Kühling, and A. Ibing. In-platform baseband channel emulator for early stage realtime testing. unpublished.

[EMK06] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Conf. on Uncertainty in Artificial Intelligence*, July 2006.

[ESB+98] O. Edfors, M. Sandell, J. Beek, S. Wilson, and P. Börjesson. OFDM channel estimation by singular value decomposition. *IEEE Trans. Commun.*, 46(7), 1998.

[Ett] Ettus Research. *Universal Software Radio Peripheral*. online `http://www.ettusresearch.com/downloads/ettus_ds_usrp_v7.pdf` (last visited on 25.01.2011).

[FAFF02] J. Feldman, I. Abou-Faycal, and M. Frigo. A fast maximum-likelihood decoder for convolutional codes. *IEEE 56th Veh. Tech. Conf.*, 2002.

[Fle00] B. Fleury. First- and second-order characterization of direction dispersion and space selectivity in the radio channel. *IEEE Trans. Inf. Theory*, 46(6):2027–2044, 2000.

[FMP+07] A. Forenza, M. McKay, A. Pandharipande, R. Heath, and I. Collings. Adaptive MIMO transmission for exploiting the capacity of spatially correlated channels. *IEEE Trans. Veh. Techn.*, 56(2), March 2007.

[FP85] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comput.*, 44:463–471, 1985.

[Fre07] Freescale. *Modular AdvancedMC Platform for Broadband/LTE Base Stations*, 2007. Document Number: LTEWIMAXFS.

[FSS11] G. Falcao, L. Sousa, and V. Silva. Massively LDPC decoding on multicore architectures. *IEEE Trans. Parallel Distrib. Syst.*, 22, Feb. 2011.

[Fu05] M. Fu. Stochastic analysis of turbo decoding. *IEEE Trans. Inf. Theory*, 51(1):81–100, Jan. 2005.

[Gal68] R. Gallager. *Information Theory and Reliable Communication*. Wiley, 1968.

[Gal07] G. Del Galdo. *Geometry-based Channel Modelling for Multi-User MIMO Systems and Applications*. PhD thesis, TU Ilmenau, 2007.

[GH11] Q. Guo and D. Huang. EM-based joint channel estimation and detection for frequency selective channels using Gaussian message passing. *IEEE Trans. Signal Process.*, 59(8):4030–4035, 2011.

[GIM+06] J. Glossner, D. Iancu, M. Moudgill, G. Nacer, S. Jinturkar, and M.l Schulte. The Sandbridge SB3011 SDR platform. In *Int. Forum Application-Specific Multi-Processor SOC*, 2006.

[GLMZ07] G. Giannakis, Z. Liu, X. Ma, and S. Zhou. *Space-Time Coding for Broadband Wireless Communications*. Wiley, 2007.

[Goo58] I. Good. The interaction algorithm and practical Fourier analysis. *J. R. Statist. Soc.*, 20(2):361–372, 1958.

[GTW03] F. Gilbert, M. H. Thul, and N. Wehn. Communication centric architectures for turbo-decoding on embedded multiprocessors. In *Design, Automation and Test Europe*, pages 356–461, March 2003.

[GZC+09] H. Guo, J. Zhao, J. Chen, X. Chen, and J. Wang. High performance turbo decoder on CELL BE for WiMAX system. In *IEEE WCSP*, 2009.

[Hag02] J. Hagenauer. The turbo principle in mobile communications. In *IEEE Int. Symp. Inform. Theory Appl.*, 2002.

[HAGO05] V. Hassel, M. Alouini, D. Gesbert, and G. Oien. Exploiting multiuser diversity using multiple feedback thresholds. In *IEEE 61st Veh. Tech. Conf.*, 2005.

[Har07] M. Harman. The current state and future of search based software engineering. In *Future of Software Engineering (FoSE)*, 2007.

[Hay01] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 2001.

[Hei09] Heinrich-Hertz Institut. *Software defined radio testbed*, June 2009. online http://www.hhi.fraunhofer.de/en/departments/wireless-communication-and-networks/events/icc-2009/software-defined-radio/ (last visited on 24.08.2011).

[HH] D. Hughes-Hartogs. *Ensemble Modem Structure for Imperfect Transmission Media*. U.S. Patents Nos. 4,679,227 (July 1987), 4,731,816 (March 1988), and 4,833,706 (May 1989).

[HHA07] K. Huang, R. Heath, and J. Andrews. Space division multiple access with a sum feedback rate constraint. *IEEE Trans. Signal Process.*, 55(7):3879–3891, July 2007.

[HHIZ08] R. Halfmann, T. Haustein, A. Ibing, and W. Zirwas. Weiterleitungsknoten und Endgerät für ein FDD-Kommunikationsnetzwerk und Verfahren diese zu betreiben. German patent application 102008004690.6, Jan. 2008. applicant: Nokia Siemens Networks.

[HIK09] C. Holmes, A. Ibing, and D. Kühling. On network processor based protocol stack implementation for 4G base stations. In *Wireless Telecommun. Symp.*, 2009.

[HJL05] Z. Han, Z. Ji, and K. Liu. Fair multiuser channel allocation for OFDMA networks using Nash bargaining solutions and coalitions. *IEEE Trans. Commun.*, 53(8), Aug. 2005.

[HKMS04] K. Higuchi, H. Kawai, N. Maeda, and M. Sawahashi. Adaptive selection of surviving symbol replica candidates based on maximum reliability in QRDM-MLD for OFCDM MIMO multiplexing. In *IEEE Global Commun. Conf.*, 2004.

[HKR97] P. Hoeher, S. Kaiser, and P. Robertson. Two-dimensional pilot-symbol-aided channel estimation by wiener filtering. In *Int. Conf. Acoustics, Speech and Signal Process.*, pages 1845–1848, April 1997.

[HLR$^+$08] B. Hu, I. Land, L. Rasmussen, R. Piton, and B. Fleury. A divergence minimization approach to joint multiuser decoding for coded CDMA. *IEEE J. Sel. Areas Commun.*, 2008.

[HSM05] J. Hou, P. Siegel, and L. Milstein. Design of multi-input, multi-output systems based on low-density parity-check codes. *IEEE Trans. Commun.*, 53(4):601–611, April 2005.

[HtB03] B. Hochwald and S. ten Brink. Achieving near-capacity on a multiple-antenna channel. *IEEE Trans. Commun.*, 51(3):389–399, March 2003.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[IB07] A. Ibing and H. Boche. Fairness vs. efficiency: Comparison of game theoretic criteria for OFDMA scheduling. In *Asilomar Conf. Signals, Systems and Comput.*, 2007.

[IB08] A. Ibing and H. Boche. Fair OFDMA scheduling algorithm using iterative local search with k-opt switches. In *IEEE Wireless Commun. Networking Conf.*, 2008.

[IB10] A. Ibing and H. Boche. On predicting convergence of iterative MIMO detection-decoding with concatenated codes. *IEEE Trans. Veh. Techn.*, 59(8):4134–4139, 2010.

[IB11] A. Ibing and H. Boche. Automatic joint optimization of iterative MIMO-OFDM receiver algorithms on a meta level. In *IEEE Int. Conf. Commun.*, 2011. in press.

[Ibi08] A. Ibing. Software defined radio testbed. invited talk at BMBF Statusseminar Mobilkommunikation, 2008.

[IBM06] IBM, Sony, Toshiba. *Cell Broadband Engine Architecture (v1.01)*, October 2006. http://www.ibm.com/chips/techlib/techlib.nsf/techdocs/1AEEE1270EA2776387257060006E61BA [Online; accessed 31-Aug-2007].

[IBM07a] IBM, Sony, Toshiba. *Cell Broadband Engine Programming Handbook (v1.1)*, April 2007. online http://www.ibm.com/chips/techlib/techlib.nsf/techdocs/9F820A5FFA3ECE8C8725716A0062585F (accessed 31-Aug-2007).

[IBM07b] IBM, Sony, Toshiba. *PPU & SPU C/C++ Language Extension Specification (v2.4)*, August 2007. http://www.ibm.com/chips/techlib/techlib.nsf/techdocs/30B3520C93F437AB87257060006FFE5E [Online; accessed 31-Aug-2007].

[IBM07c] IBM, Sony, Toshiba. *SPU Instruction Set Architecture (v1.2)*, January 2007. http://www.ibm.com/chips/techlib/techlib.nsf/techdocs/76CA6C7304210F3987257060006F2C44 [Online; accessed 31-Aug-2007].

[Ibp07] Wireless telecommunication system including a base station, relay node and method for global fair scheduling. European patent EP 2 066 084 B1; international application WO 2009/068413 A1, 2007. holders: Nokia Siemens Networks, Fraunhofer Gesellschaft; inventors: R. Halfmann, T. Haustein, A. Ibing and W. Zirwas.

[Ibp08] Verfahren und Vorrichtung zur Datenübertragung. German patent applications DE 10 2008 007 497 A1, Feb. 2008. applicant: Fraunhofer Gesellschaft; inventors: W. Zirwas, J. Eichinger, A. Ibing, L. Thiele, V. Jungnickel and T. Haustein.

[IHJ08] A. Ibing, Y. Hadisusanto, and V. Jungnickel. Scalable network multicast for cooperative base stations. In *2nd Int. Conf. Commun. Syst. Software and Middleware*, Jan. 2008.

[IJ08] A. Ibing and V. Jungnickel. Joint transmission and detection in hexagonal grid for 3GPP LTE. In *International Conference on Information Networking (ICOIN)*, Jan. 2008.

[IKB09] A. Ibing, D. Kühling, and H. Boche. On the relation of MIMO APP detection and SIMO maximum ratio combining. In *IEEE Global Commun. Commun. Conf.*, 2009.

[IKB10] A. Ibing, D. Kühling, and H. Boche. Reliability-based hybrid MMSE/subspace-max-log-APP MIMO detector. *IEEE Commun. Lett.*, 14(5):387–389, 2010.

[IKK$^+$08] A. Ibing, D. Kühling, M. Kuszak, V. Jungnickel, and C. Helmolt. Flexible demonstrator platform for cooperative joint transmission and detection in next generation wireless MIMO-OFDM networks. In *4th Int. Conf. Testbeds and Research Infrastructures for the Develop. of Networks and Communities*, 2008.

[IKWB09] A. Ibing, D. Kühling, D. Wieruch, and H. Boche. Software defined hybrid MMSE/QRD-M turbo receiver for LTE Advanced uplink on a Cell processor. In *IEEE Int. Conf. Commun.*, 2009.

[IM08] A. Ibing and K. Manolakis. MMSE channel estimation and time synchronisation tracking for cooperative MIMO-OFDM with propagation delay differences. In *5th Int. Symp. Wireless Communication Systems*, 2008.

[Int05] Intel. *IXP2350 Network Processor; product brief*, 2005.

[IOB11] A. Ibing, P. Otto, and H. Boche. On channel correlation based scheduling and signalling for MIMO-OFDMA downlink. In *IEEE 73rd Vehicular Technology Conf.*, 2011. in press.

[IP 05] IP Fabrics, Inc. *Double Espresso Hardware User's Manual*, 2005. online http://www.ipfabrics.com/products/de.php.

[ipa10a] Ad hoc mobile devices and and hoc networks. US patent application US 12/941.861, Nov. 2010. applicant: TU Berlin; inventors: A. Ibing and H. Boche.

[ipa10b] Receiver. US patent application US 12/843.721, July 2010. applicant: TU Berlin; inventors: A. Ibing and H. Boche.

[ipa10c] Receiver. US patent application US 12/923.534, Sept. 2010. applicant: TU Berlin; inventors: A. Ibing and H. Boche.

[ipe] *iperf*. online http://sourceforge.net/projects/iperf (visited on 14. Jan. 2009).

[ixa] *IXA SDK*. online http://www.netronome.com/pages/intel-sdk-downloads (visited on 14. Jan. 2009).

[IZHH08] A. Ibing, W. Zirwas, T. Haustein, and R. Halfmann. Verfahren zum Steuern einer Relaisstation in einem Mobilfunknetzwerk. German patent application DE102008009087A1, 2008. applicant: Fraunhofer-Gesellschaft.

[IZK08] A. Ibing, X. Zhao, and D. Kühling. Discontinuous timing advance tracking for OFDMA uplink. In *International OFDM Workshop*, 2008.

[JA71] F. Jelinek and J. Anderson. Instrumentable tree encoding of information sources. *IEEE Trans. Inf. Theory*, 17:118, 1971.

[JFH$^+$05] V. Jungnickel, A. Forck, T. Haustein, S. Schiffermüller, C. Helmolt, F. Luhn, M. Pollock, C. Juchems, M. Lampe, W. Zirwas, J. Eichinger, and E. Schulz. 1Gbit/s MIMO-OFDM transmission experiments. In *IEEE 62nd Veh. Tech. Conf.*, 2005.

[JGA09] M. Jordan, X. Gong, and G. Ascheid. Conversion of the spatio-temporal correlation from uplink to downlink in FDD systems. In *IEEE Wireless Commun. Networking Conf.*, 2009.

[JK03] Erik J. Johnson and Aaron R. Kunze. *IXP2400/2800 Programming*. Intel Press, 2003.

[JL07] K. Jeong and J. Lee. Low complexity channel tracking for adaptive MMSE channel estimation in OFDM. In *Conf. Int. Sci. Syst.*, 2007.

[JN02] F. Jensen and T. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2nd edition, 2002.

[JPP00] A. Jalali, R. Padovani, and R. Pankai. Data throughput of CDMA HDR a high efficiency-high data rate personal communication wireless system. In *IEEE 51st Veh. Tech. Conf.*, pages 1854–1858, 2000.

[JS07] B. Johansson and T. Sundin. *LTE test bed*. Ericsson Review, 2007. online http://www.ericsson.com/ericsson/corpinfo/publications/review/2007_01/files/2_lte_web.pdf (last visited on 21.01.2011).

[JSAM07] M. Jordan, L. Schmitt, G. Ascheid, and H. Meyr. Prediction of downlink SNR for opportunistic beamforming. In *IEEE Wireless Commun. Networking Conf.*, 2007.

[K07] D. Kühling. Design and realtime implementation of MAC-layer hardware and software components for a 3GPP LTE base station. Master's thesis, Technische Universität Berlin, Dec. 2007.

[KB06] M. Khalighi and J. Boutros. Semi-blind channel estimation using the EM algorithm in iterative MIMO APP detectors. *IEEE Trans. Wireless Commun.*, 5(11):3165–3173, 2006.

[KBD07] J. Kurzak, A. Buttari, and J. Dongarra. LAPACK working note 184: Solving systems of linear equations on the Cell processor using Cholesky factorization. Technical Report UT-CS-07-596, University of Tennessee, May 2007. online http://eprints.ma.man.ac.uk/821/.

[KBH06] M. Khalighi, J. Boutros, and J. Helard. Data-aided channel estimation for Turbo-PIC MIMO detectors. *IEEE Commun. Lett.*, 10(5):350–352, 2006.

[KDH$^+$05] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the cell multiprocessor. *IBM J. Res. Dev.*, 49(4/5):589–604, 2005.

[KFL01] F. Kschischang, B. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2), Feb. 2001.

[KH00] T. Keller and L. Hanzo. Adaptive multicarrier modulation: A convenient framework for time-frequency processing in wireless communications. *Proc. IEEE*, 88(5), May 2000.

[KH05] H. Kim and Y. Han. A proportional fair scheduling for multicarrier transmission systems. *IEEE Commun. Lett.*, 9(3), March 2005.

[KI08] D. Kühling and A. Ibing. A novel low-complexity algorithm for linear MMSE MIMO receivers. In *Int. Symp. Wireless Pervasive Computing*, 2008.

[KID10] K. Khan, A. Ibing, and D. Dahlhaus. Joint model for fine synchronization and adaptive

LMMSE channel estimation in uplink OFDMA. In *7th Int. Symp. Wireless Communication Systems*, 2010.

[KIJ08] D. Kühling, A. Ibing, and V. Jungnickel. 12x12 MIMO-OFDM realtime implementation for 3GPP LTE+ on a Cell processor. In *European Wireless Conf.*, 2008.

[KL08] S. Kim and Y. Lee. 3-dimensional MMSE channel estimation in multi-antenna OFDM systems. In *IEEE Int. Conf. Digital Telecommun.*, 2008.

[KMGW03] F. Kienle, H. Michel, F. Gilbert, and N. Wehn. Efficient MAP-algorithm implementation on programmable architectures. *Advances Radio Sci.*, 1:259–263, 2003.

[KMT98] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *J. Operational Research Soc.*, 1998.

[KRJ00] B. Krongold, K. Ramchandran, and D. Jones. Computationally efficient optimal power allocation algorithms for multicarrier communication systems. *IEEE Trans. Commun.*, 48(1), 2000.

[KSI07] D. Kühling, S. Schiffermüller, and A. Ibing. On efficient computation of MMSE equalizer matrix for MIMO-OFDM systems. In *Workshop on Commercial MIMO-Components and -Systems*, Sep 2007.

[KWA+09] T. Kempf, S. Wallentowitz, G. Ascheid, R. Leupers, and H. Meyr. A workbench for analytical and simulation based design space exploration of software defined radios. In *Int. Conf. VLSI Design*, 2009.

[Lai08] K. Lai. Reduced-complexity MIMO detection using adaptive set partitioning. In *IEEE Wireless Commun. Networking Conf.*, 2008.

[LAMC11] I. Lai, G. Ascheid, H. Meyr, and T. Chiueh. Efficient channel-adaptive MIMO detection using just-acceptable error rate. *IEEE Trans. Wireless Communications*, 10(1), Jan. 2011.

[LC04] S. Lin and D. Costello. *Error Control Coding*. Pearson, 2004.

[LHL+08] D. Love, R. Heath, V. Lau, D. Gesbert, B. Rao, and M. Andrews. An overview of limited feedback in wireless communication systems. *IEEE J. Sel. Areas Commun.*, 26(8), Oct. 2008.

[LI] H. Li and A. Ibing. On probability density modelling for a posteriori OFDM channel estimation. unpublished.

[Li00] Y. Li. Pilot-symbol-aided channel estimation for ofdm in wireless systems. *IEEE Trans. Veh. Technol.*, 49:1207–1215, Jul. 2000.

[Li10a] H. Li. Turbo channel estimation for coded MIMO-OFDMA. Master's thesis, Technische Universität Berlin, Sept. 2010.

[LI10b] H. Li and A. Ibing. On complexity-reduced implementation of multi-dimensional Wiener interpolation filtering. In *IEEE 72nd Vehicular Technology Conf.*, 2010.

[LJ08] E. Larsson and J. Jalden. Fixed-complexity soft MIMO detection via partial marginalization. *IEEE Trans. Signal Process.*, 56(8):3397–3407, 2008.

[LL09] D. Lin and T. Lim. A variational inference framework for soft-in soft-out detection in multiple-access channels. *IEEE Trans. Inf. Theory*, 55(5):2345–2346, May 2009.

[LLL82] A. Lenstra, H. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, (261):515–534, 1982.

[LLN$^+$09] C. Liao, I. Lai, K. Nikitopoulos, F. Borlenghi, D. Kammler, M. Witte, D. Zhang, T. Chiueh, G. Ascheid, and H. Meyr. Combining orthogonalized partial metrics: Efficient enumeration for soft-input sphere decoder. In *Ann. IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun.*, 2009.

[LM87] E. Lee and D. Messerschmitt. Synchronous data flow. *Proc. IEEE*, 75(9), 1987.

[LMM$^+$06] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, and K. Flautner. Design and implementation of turbo decoders for software defined radio. In *IEEE Workshop Signal Process. Syst. Design and Implementation*, pages 22–27, Oct. 2006.

[LS88] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Roy. Statistical Soc.*, 50(2):157–224, 1988.

[Mac04] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2nd edition, 2004.

[MC11] J. Moon and D. Cho. Efficient cell-clustering algorithm for inter-cluster interference mitigation in network MIMO systems. *IEEE Commun. Lett.*, Jan. 2011.

[Mer] Mercury Computer Systems Inc. *Cell Accelerator Board*. online http://www.mc.com/products/boards/accelerator_board2.aspx (last visited on 15.02.2011).

[MH02a] J. Moon and S. Hong. Adaptive code-rate and modulation for multi-user OFDM system in wireless communications. In *IEEE 56th Veh. Tech. Conf.*, 2002.

[MH02b] M. Münster and L. Hanzo. MMSE channel prediction assisted symbol-by-symbol adaptive OFDM. In *IEEE Int. Conf. Commun.*, pages 416–420, 2002.

[MIJ08] K. Manolakis, A. Ibing, and V. Jungnickel. Performance evaluation of a 3GPP LTE terminal receiver. In *European Wireless Conf.*, 2008.

[Min05] T. Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005. online ftp://ftp.research.microsoft.com/pub/tr/TR-2005-173.pdf (last visited on 15.02.2011).

[MK09] B. Mielczarek and W. Krzymien. Adaptive CSI prediction in linear multi-user MIMO systems. In *IEEE 70th Veh. Tech. Conf.*, 2009.

[MKF$^+$09] C. Manchon, G. Kirkelund, B. Fleury, P. Mogensen, L. Deneire, T. Sorensen, and C. Rom. Interference cancellation based on divergence minimization for MIMO-OFDM receivers. In *IEEE Global Commun. Conf.*, 2009.

[MMC98] R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE J. Sel. Areas Commun.*, 16(2), Feb. 1998.

[MMD91] R. Mazumdar, L. Mason, and C. Douligeris. Fairness in network optimal flow control: Optimality of product forms. *IEEE Trans. Commun.*, 39(5), May 1991.

[Mod89] R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16:233–248, 1989.

[MS83] R. Mersereau and T. Speake. The processing of periodically sampled multidimensional signals. *IEEE Trans. Acoust., Speech, Signal Process.*, 1983.

[Myr] Myricom. *10GBase-R*. online http://www.myri.com/Myri-10G/NIC/10G-PCIE-8A-R.html (visited in 2008).

[MZB00] H. Minn, M. Zeng, and V. K. Bhargava. On timing offset estimation for OFDM systems. *IEEE Commun. Lett.*, pages 242–244, July 2000.

[Nim08] Nimbalker et al. ARP and QPP interleavers for LTE turbo coding. In *IEEE Wireless Commun. Networking Conf.*, pages 1032–1037. IEEE, 31 2008-April 3 2008.

[NVI07] NVIDIA. *NVIDIA Tesla GPU Computing Technical Brief*, 2007. online `http://www.nvidia.com/docs/IO/43395/Compute_Tech_Brief_v1-0-0_final__Dec07.pdf` (visited on 10.11.2010).

[obs] Open base station architecture initiative, 'Reference point 3 specification version 4.0'. online `http://www.obsai.org` (last visited on 15.02.2011).

[OJ09] J. Ousterhout and K. Jones. *TCL and the TK Toolkit*. Addison-Wesley, 2009.

[One] One Stop Systems. *PCIe Cable Adapters*. online `http://www.onestopsystems.com/components.php` (last visited on 26.01.2011).

[ONSS08] D. Obradovic, C. Na, L. Scheiterer, and A. Szabe. EM-based semi-blind channel estimation method for MIMO-OFDM communication systems. *Neurocomputing*, 71:2388–2398, 2008.

[ORP05] B. Özbek, D. Ruyet, and C. Panazio. Pilot-symbol-aided iterative channel estimation for OFDM-based systems. In *European Signal Process. Conf.*, 2005.

[OS09] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 3rd edition, 2009.

[Par] Pareto efficiency. Wikipedia. online `http://en.wikipedia.org/w/index.php?title=Pareto_efficiency&oldid=376222647`.

[Par62] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33:1065–1076, 1962.

[Pat02] M. Patzold. *Mobile Fading Channels*. Wiley, 2002.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd edition edition, 1988.

[PGNB04] A. Paulraj, D. Gore, R. Nabar, and H. Bolcskei. An overview of MIMO communications - a key to gigabit wireless. *Proc. IEEE*, 92(2):198–218, February 2004.

[PS82] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[PS07] H. Park and M. Schaar. Fairness strategies for multi-user multimedia applications in competitive environments using the Kalai-Smorodinsky bargaining solution. In *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2007.

[PSP10] E. Panayirci, H. Senol, and V. Poor. Joint channel estimation, equalization, and data detection for OFDM systems in the presence of very high mobility. *IEEE Trans. Signal Process.*, 58(8):4225–4238, 2010.

[Ram07] U. Ramacher. Software-defined radio prospects for multistandard mobile phones. *IEEE Computer*, 40(10), 2007.

[RBO04] D. Ruyet, T. Bertozzi, and Berna Özbek. Breadth first algorithms for APP detectors over MIMO channels. In *IEEE Int. Conf. Commun.*, 2004.

[RC00] W. Rhee and J. Cioffi. Increase in capacity of multiuser OFDM system using dynamic subchannel allocation. In *IEEE 51st Veh. Tech. Conf.*, 2000.

[RCP10] S. Ramprashad, G. Caire, and H. Papadopoulos. A joint scheduling and cell clustering scheme for MU-MIMO downlink with limited coordination. In *IEEE Int. Conf. Commun.*, 2010.

[RHV07] V. Ramon, C. Herzet, and L. Vandendorpe. A semi-analytical method for predicting the performance and convergence behaviour of a multiuser turbo-equalizer/demapper. *IEEE Trans. Signal Process.*, 55(3), 2007.

[RU08] T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.

[RVH95] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain. In *IEEE Int. Conf. Commun.*, 1995.

[SA03] M. Sternad and D. Aronsson. Channel estimation and prediction for adaptive OFDM downlink. In *IEEE 58th Veh. Tech. Conf.*, pages 1283–1287, 2003.

[SAL99] J. Su, A. Annamalai, and W. Lu. Optimization of power allocation in a multicell DS/CDMA system with heterogeneous traffic. In *IEEE Int. Conf. Commun.*, pages 1136–1140, June 1999.

[SB07] M. Schubert and H. Boche. Properties and operational characterization of proportionally fair resource allocation. In *IEEE Int. Workshop Signal Process. Advances in Wireless Commun.*, June 2007.

[SB10] C. Studer and H. Bolcskei. Soft-input soft-output single tree-search sphere decoding. *IEEE Trans. Inf. Theory*, 56(10), Oct. 2010.

[SBB95] M. Sandell, J. Beek, and P. Borjesson. Timing and frequency synchronization in OFDM systems using the cyclic prefix. In *Proc. Int. Symp. Synchronization*, 1995.

[SC97] T. Schmidl and D. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Trans. Commun.*, 1997.

[Sca09] M. Scarpino. *Programming the Cell Processor*. Prentice Hall, 2009.

[SCS$^+$08] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: A many-core x86 architecture for visual computing. *ACM Trans. Graphics*, 27(3), 2008.

[SDU06] S. Sand, A. Dammann, and A. Usmani. Iterative OFDM receiver with channel estimation. In *Int. Symp. Wireless Personal Multimedia Commun.*, 2006.

[SE94] C. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Programming*, 66:181–191, 1994.

[SF11] D. Shutin and B. Fleury. Sparse variational Bayesian SAGE algorithm with application to the estimation of multipath wireless channels. *IEEE Trans. Signal Process.*, 59(8):3609–3623, 2011.

[SFFM01] M. Speth, S. Fechtel, G. Fock, and H. Meyr. Optimum receiver design for OFDM-based broadband transmission - part ii: a case study. *IEEE Trans. Commun.*, 49(4):571–578, April 2001.

[SHFS04] V. Srivastava, C. Ho, P. Fung, and S. Sun. Robust MMSE channel estimation in OFDM systems with practical timing synchronization. In *IEEE WCNC*, 2004.

[Sim90] S. Simmons. Breadth-first trellis decoding with adaptive effort. *IEEE Trans. Inf. Theory*, 38:3–12, 1990.

[SJ06] S. Schiffermuller and V. Jungnickel. Practical channel interpolation for OFDMA. In *IEEE Global Commun. Conf.*, 2006.

[SJS03] F. Sanzi, S. Jelting, and J. Speidel. A comparative study of iterative channel estimators for mobile OFDM systems. *IEEE Trans. Wireless Commun.*, 2(5):849–859, Sept. 2003.

[SK08] S. Stefanatos and A. Kattsegelos. Joint data detection and channel tracking for OFDM systems with phase noise. *IEEE Trans. Signal Process.*, 56(9):4230–4243, 2008.

[SL03] W. Song and J. Lim. Pilot-symbol aided channel estimation for OFDM with fast fading channels. *IEEE Trans. Broadcast.*, 49(4):398–402, December 2003.

[SZ03] T. Schenk and A. Zelst. Frequency synchronization for MIMO OFDM wireless LAN systems. In *IEEE 58th Veh. Tech. Conf.*, 2003.

[Tel99] E. Telatar. Capacity of multi-antenna Gaussian channels. *European Trans. Telecommun.*, 10(6):585–596, 1999.

[ten01] S. tenBrink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Trans. Commun.*, 49(10):1727–1737, October 2001.

[Tex07] Texas Instruments. *TMS320TCI6488 DSP Platform Product Bulletin*, 2007.

[Tho94] W. Thomson. *Handbook of Game Theory*, volume 2, chapter 35, Cooperative Models of Bargaining, pages 1237–1283. Elsevier Science, 1994.

[Til09] Tilera. *TILExpress-20GTM Card Product Brief*, 2009. online http://www.tilera.com/products/platforms/TILExpress-20G_card (visited on 10.11.2010).

[UYLW09] Y. Ueng, C. Yeh, M. Lin, and C. Wang. Turbo coded multiple-antenna systems for near-capacity performance. *IEEE J. Sel. Areas Commun.*, 27(6):954–964, Aug. 2009.

[VB99] E. Viterbo and J. Boutros. A universal lattice code decoder for fading channels. *IEEE Trans. Inf. Theory*, 45(5):1639–1642, 1999.

[Vid] Videolan. *VLC media player*. online http://www.videolan.org/vlc/ (last visited on 08.02.2011).

[Vog02] J. Vogt. *Beiträge zur effizienten Decodierung von Turbo-Codes*. PhD thesis, TU Dresden, Germany, May 2002.

[VS01] M. C. Valenti and J. Sun. The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios. *Int. J. Wireless Inf. Networks*, 8(4), October 2001.

[W05̈] D. Wübben. *Effiziente Detektionsverfahren für Multilayer-MIMO-Systeme*. PhD thesis, Universität Bremen, Dez. 2005.

[WBA$^+$10] E. Witte, F. Borlenghi, G. Ascheid, R. Leupers, and H. Meyr. A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding. *IEEE Trans. Circuits Syst.*, 57(9), Sept. 2010.

[WCLM99] C. Wong, R. Cheng, K. Letaief, and R. Murch. Multiuser OFDM with adaptive subcarrier, bit, and power allocation. *IEEE J. Sel. Areas Commun.*, 17(10), October 1999.

[WDS10] J. White, B. Doughtery, and D. Schmidt. ASCENT: An algorithmic technique for designing hardware and software in tandem. *IEEE Trans. Software Engineering*, 36(6):838–851, 2010.

[WE08] I. Wong and B. Evans. Optimal downlink OFDMA resource allocation with linear complexity to maximize ergodic rates. *IEEE Trans. Wireless Commun.*, 7(2), Feb. 2008.

[WF01] Y. Weiss and W. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, Oct. 2001.

[WG04] R. Wang and G. Giannakis. Approaching MIMO channel capacity with reduced-complexity soft sphere decoding. In *IEEE Wireless Commun. Networking Conf.*, 2004.

[WHZ06] Y. Wen, W. Huang, and Z. Zhang. CAZAC sequence and its application in LTE random access. In *IEEE Inf. Theory Workshop*, 2006.

[Wie49] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Wiley, 1949.

[Wik] Wikipedia. *Network Processor*. online http://en.wikipedia.org/w/index.php?title=Network_processor&oldid=252358185.

[wir] *wireshark*. online www.wireshark.org (visited on 14. Jan. 2009).

[WKI] D. Wieruch, D. Kühling, and A. Ibing. Realtime list-QRD-m MIMO detector on a Cell processor. unpublished.

[WKI08] D. Wieruch, D. Kühling, and A. Ibing. 250MBit/s 64-state soft-decision Viterbi decoder on a Cell CPU. In *Mikroelektroniktagung*, 2008.

[WKP01] K. Witrisal, Y. Kim, and R. Prasad. A new method to measure parameters of frequency-selective radio channel using power measurements. *IEEE Trans. Commun.*, 49(10):1788–1800, Oct 2001.

[WM04] K. Wong and P. McLane. Bi-directional soft-output m-algorithm for iterative decoding. In *IEEE Int. Conf. Commun.*, 2004.

[WMK08] S. Wu, U. Mitra, and C. Kuo. Iterative joint channel estimation and multiuser detection for DS-CDMA in frequency-selective fading channels. *IEEE Trans. Signal Process.*, 56(7):3261–3277, 2008.

[Woo06] J. Woods. *Multidimensional Signal, Image, and Video Processing and Coding*. Academic Press, 2006.

[WS01] A. Worthen and W. Stark. Unified design of iterative receivers using factor graphs. *IEEE Trans. Inf. Theory*, 47:843–850, 2001.

[WY10] L. Wang and C. Yeh. A three-cell coordinated network MIMO with fractional frequency reuse and directional antennas. In *IEEE Int. Conf. Commun.*, 2010.

[Wym07] H. Wymeersch. *Iterative Receiver Design*. Cambridge University Press, 2007.

[WZS08] F. Wan, W. Zhu, and M. Swamy. A semiblind channel estimation approach for MIMO-OFDM systems. *IEEE Trans. Signal Process.*, 56(7):2821–2834, 2008.

[Xil] Xilinx. *Virtex-5 ML555 Development Kit for PCI and PCI Express Designs, User Guide*. online `http://www.xilinx.com/bvdocs/userguides/ug201.pdf`.

[YA08] T. Yücek and H. Arslan. Time dispersion and delay spread estimation for adaptive OFDM systems. *IEEE Trans. Veh. Techn.*, 57(3), May 2008.

[YFW01] J. Yedidia, W. Freeman, and Y. Weiss. Tr-2001: Bethe free energy, Kikuchi approximations, and belief propagation algorithms. Technical report, Mitsubishi electric research labs, May 2001.

[YFW05] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. Inf. Theory*, 51(7):2282–2312, July 2005.

[YKGI03] J. Yue, K. Kim, J. Gibson, and R. Iltis. Channel estimation and data detection for MIMO-OFDM systems. In *IEEE Global Commun. Conf.*, pages 581–585, 2003.

[YKJ10] J. Ylioinas, J. Karjalainen, and M. Juntti. On the activation ordering of detector, decoder, and channel estimator in iterative receiver for MIMO-OFDM. In *IEEE Int. Conf. Commun.*, 2010.

[YL00] H. Yin and H. Liu. An efficient multiuser loading algorithm for OFDM-based broadband wireless systems. In *IEEE Global Commun. Conf.*, 2000.

[YL08] S. Yoon and S. Lee. A detection algorithm for multi-input multi-output (MIMO) transmission using poly-diagonalization and trellis decoding. *IEEE J. Sel. Areas Commun.*, 26(6):993–1002, Aug. 2008.

[YLCC00] B. Yang, K. Letaief, R. Cheng, and Z. Cao. Timing recovery for OFDM transmission. *IEEE J. Sel. Areas Commun.*, pages 2278–2290, Nov 2000.

[YSC07] D. Yu, K. Seong, and J. Cioffi. Multiuser discrete bit-loading for digital subscriber lines. In *IEEE Int. Conf. Commun.*, 2007.

[YW05] G. Yue and X. Wang. Optimization of irregular repeat-accumulate codes for MIMO systems with iterative receivers. *IEEE Trans. Wireless Commun.*, 4(6):2843–2855, Nov. 2005.

[ZF06a] E. Zimmermann and G. Fettweis. Improved length term calculation and MMSE extension for LISS MIMO detection. In *IEEE Inf. Theory Workshop*, 2006.

[ZF06b] E. Zimmermann and G. Fettweis. Unbiased MMSE tree search detection for multiple antenna systems. In *Int. Symp. Wireless Personal Multimedia Commun.*, 2006.

[ZI09] X. Zhao and A. Ibing. Performance evaluation of a low-complexity LTE base station receiver. In *IEEE International Workshop on Multicarrier Systems and Solutions (MCSS)*, 2009.

[Zim07] E. Zimmermann. *Complexity Aspects in Near-Capacity MIMO Detection-Decoding*. PhD thesis, TU Dresden, 2007.

[ZLNA10] D. Zhang, I. Lai, K. Nikitopoulos, and G. Ascheid. Informed message update for iterative MIMO demapping and turbo decoding. In *Int. Symp. Inf. Theory Appl.*, Oct. 2010.