Universität Passau

Doctoral Thesis

Fault Tolerance Aspects of Virtual Massive MIMO Systems

Author: Victor Tomashevich *Supervisor: Prof. Dr. Ilia Polian*

A thesis submitted in partial fulfillment of the requirements for the degree of Dr. rer. nat.

in the

Fakultät für Informatik und Mathematik

April 2016

UNIVERSITÄT PASSAU

Abstract

Fakultät für Informatik und Mathematik

Dr. rer. nat.

Fault Tolerance Aspects of Virtual Massive MIMO Systems

by Victor TOMASHEVICH

Employment of a very large number of antennas is seen as the key technology to provide future users with very high data rates. At the same time, the implementation complexity will rise due to large memories required and sophisticated signal processing algorithms employed. Continuous technology downscaling allows implementation of such complex digital designs. At the same time, its inherent variability and vulnerability to physical disturbances violate the assumption of perfectly reliable hardware operation.

This work considers Unique Word OFDM which represents the alternative to the standard Cyclic Prefix OFDM providing superior detection quality. The generalization of Unique Word OFDM to a MIMO system is performed which allows interpretation as a virtual massive MIMO system with only few physical antennas. Detection methods for the introduced generalization are discussed and their performance is quantified. Because of the large memory size required, linear detection represents the cost and performance effective solution. The possible memory errors due to radiation effects or voltage scaling are addressed and the nonlinear MMSE detection algorithm is proposed. This algorithm keeps track of the memory errors and is able to significantly mitigate their effect on the quality of the estimated data.

Apart of memory issues, reliability of the actual computational hardware which constitutes the receiver is of concern in this work. An own implementation of the MMSE Sorted Givens Rotations is subjected to transient fault injection. The impact of faults in various parts of the implemented circuit on the detection performance is quantified. Most vulnerable components of the implemented circuit in terms of reliability are identified.

Security is another major address of this work, since most current implementations include cryptographic devices. Fault-based attacks on such systems are known to be able to extract the secret key in feasible time. The remaining part of this work addresses such fault injection-based malicious attacks. Countermeasures based on a combination

of information and hardware redundancy are considered. Recently introduced robust codes target such attacks by providing guaranteed detection capability. The performance of these codes is assessed by application to actual cryptographic and general purpose circuits. The work introduces metrics that help to identify fault locations in the circuit which could escape detection with high probability. These locations are targeted by transistor resizing that renders fault injection unfeasible.

Acknowledgements

First of all, I want to thank my supervisor, Professor Ilia Polian for his valuable suggestions, support and guidance throughout my Ph. D. studies. He encouraged me to work on this interdisciplinary topic and helped me to overcome the challenges in my research. It is also my pleasure to express my gratitude to Professor Tirkkonen for agreeing to act as an external co-examiner of my Ph. D. thesis and to all members of the examination committee.

I am very grateful to my co-authors Sudarshan Srinivasan, Raghavan Kumar, Christina Gimmler-Dumont, Christian Fesl, Fabian Förg, Yaara Neumeier, Professor Osnat Keren and Professor Norbert Wehn for fruitful collaboration and valuable discussions. Without their profound expertise and great commitment this work would not have been completed. Special thanks goes to my colleagues at the chair of computer engineering Jie Jiang and Alexandru Paler for providing excellent work environment and lively discussions sometimes far from the main field of research.

I would like to thank my parents for inculcating in me the interest in science and for teaching me to never rest on the laurels. My heartfelt gratitude goes to my beloved wife, Maria. Her patience, love and encouragement have been my major support. Finally and most importantly, I want to thank my daughters for being an endless source of fun and inspiration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Massive multiple-antenna (MIMO) systems have recently gained attention as potential means to improve spectral efficiency, link quality and coverage compared to contemporary small-scale MIMO [1]. This is achieved by employing hundreds of antennas. The improvements come at the cost of the computational complexity increase at the receiver [2].

Realization of such complex architectures should become possible thanks to the continuous shrinking of complementary metal oxide semiconductor (CMOS) devices during recent years. However, continuous downscaling reached a point, where the expected characteristics of the produced devices cannot be guaranteed [3]. For example, leakage and drive currents vary from one produced die to the other. Sources of such variations include fluctuations of geometric dimensions due to lithographic and etching techniques and variable number of dopant atoms [4]. Furthermore, reducing transistor size makes it susceptible to temperature variations, which accelerate the wear-out. Reducing scale makes the CMOS devices more vulnerable to ionizing radiation, such as alpha particles or high-energy neutrons from cosmic radiation. The latter do not damage the device permanently, but may change the values of the processed or stored logic values, leading to errors that could manifest themselves in wrongly computed variables of the algorithm that is run on this device and compromise the algorithm's correctness and/or performance. It has to be concluded that the assumption of perfectly reliable hardware does not hold anymore.

Whereas the above conclusions are drawn for hardware in general, their impact on implementations of prospect communication systems has not been largely addressed. Assuming systems with hundreds of antennas, application of classical fault-tolerance measures such as modular redundancy [5] is infeasible due to large number of components. Usage of commercial off-the-shelf components renders radiation hardening not suitable either. As massive MIMO systems require memories of large size, applying classical error detecting code protection [5] will prove costly in terms of area and power consumption increase. Fault tolerance requirement and the nature of operation of communications systems are related, since the latter always process noisy data and therefore already

possess some inherent resilience. Therefore, recent research trend is to modify the detection algorithm, taking into account the variability of the underlying hardware, such that the graceful performance degradation is achieved [6]. This controlled degradation is then traded off for reduction of power consumption, allowing energy efficient design [7]. However, these developments consider only generic signal processing subsystems or small-scale MIMO receivers and no results on massive MIMO systems were available so far. One objective of this thesis is to propose detection algorithms for large-scale MIMO systems that take memory errors into account.

The reliability of computational components (apart from memory) that constitute the sub-blocks of the detection algorithm(s) has not been largely addressed either. It is generally not clear, whether one part of the implemented computation is more vulnerable to hardware imperfections than the other. This issue is addressed in this work by analyzing the effect of transient faults in individual sub-components of the QR decomposition (one of the receivers' principal blocks) on the overall detection quality.

As by far most current systems (and communication systems are not an exception) include cryptographic components, security is another focus of this thesis. The reliability and security are related issues in nanoscale electronics, susceptibility of circuits to disturbances allows performing fault-based attacks. These malicious fault injections can be used to perform cryptanalysis and gain access to sensitive information such as secret keys. Among other countermeasures, a class of nonlinear error detecting codes referred to as robust codes has been proposed [8]. Theoretically, robust codes guarantee detection of errors caused by malicious attacks. However, their construction and performance characteristics are based on simplifying assumptions and do not take the structure of actual circuits into account. This work assesses the performance of robust codes applied to actual circuits and refines their theoretical characteristics.

1.2 Literature review

Recently the authors of [9] introduced new variant of orthogonal frequency division multiplexing (OFDM) which is closely related to massive MIMO. It is denoted as unique word (UW) OFDM. The difference lies in the encoding of sub-carriers, such that they can be regarded as a virtual (massive) MIMO system, whereas in standard cyclic prefix OFDM (CP-OFDM), the sub-carriers represent orthogonal channels. The size of the resulting virtual channel matrix is governed by the number of sub-carriers employed.

The unique word signaling has been initially introduced for single-antenna systems (SISO). The inherent coding across sub-carriers allows much better detection quality compared to SISO CP-OFDM [10, 11]. This is achieved by inherent coding gain and ability to apply MIMO detection algorithms such as linear minimum mean square error (LMMSE) or sphere decoding (SD), which are not applicable to single antenna CP-OFDM. Application of sphere decoding to UW-OFDM has been published in [12]. However, the exponential complexity of SD restricted its use only to systems with low number of sub-carriers [12, 13]. The relation to massive MIMO has not been yet elaborated and detection algorithms specifically developed for large scale systems have not been considered.

A fair comparison of SISO UW-OFDM to SISO CP-OFDM is impossible in general. UW-OFDM clearly outperforms CP-OFDM in SISO case [10, 11], due to the fact that it is essentially a (massive) MIMO system. However, current standards employ MIMO with M_t transmit and M_r receive physical antennas. UW-OFDM should be extended to support multiple physical transmit and receive antennas and its performance compared to MIMO CP-OFDM.

Continuous shrinking of CMOS devices allowed implementation of complex and low power designs both in general purpose computing and in application specific computing. They are susceptible to process variations and soft errors that introduce computational errors affecting the performance of higher layers [4]. Addressing this issue by up scaling the power supply voltage contradicts with the low power requirement. Using traditional hardware redundancy methods introduces additional area overhead and in turn increases the power consumption.

The novel approaches to provide hardware variation resilient operation can be classified to circuit-architecture co-design approaches and algorithm-architecture code-design approaches. The examples of circuit-architecture level techniques are RAZOR [14] and CRISTA [15]. RAZOR uses dynamic detection and correction of circuit timing errors by using a shadow latch. The key idea of RAZOR is to tune the supply voltage by monitoring the error rate during circuit operation, thereby eliminating the need for voltage margins and exploiting the data dependence of circuit delay [14]. CRISTA allows voltage overscaling (VOS) by prediction-based elastic clocking [15]. It isolates the critical paths of the design and provides an extra clock cycle for these paths. These methods are applicable to general purpose computing.

Algorithm-architecture co-design methods are advantageous when information from the application can be used. For example, signal processing applications are working on noisy data and do not require correction of all occurring errors. The resilience measures take the application metrics such as signal-to-noise ratio (SNR) or bit error ratio (BER) into account. Based on these performance metrics some amount of computational errors caused by variations of the underlying hardware can be potentially tolerated, leading to energy-efficient design. These methods can be roughly classified into significance driven approach (SDA) [16–18] and stochastic computing [19], however there is no strict distinction between these classes and the concrete algorithms may share common features.

The SDA uses the fact that not all computations of a particular algorithm are equally important for the quality of the algorithm output. Significant parts of the algorithm are constrained to be computed in a fraction of a cycle time. This approach has been applied to video processing designs [16].

The idea of stochastic computing is to treat the computational errors caused by timing errors or transient faults due to particle strikes as additional source of noise. The error correction is then performed in statistical sense, such that some cost function defined by the application is optimized. Stochastic computing features methods like algorithmic noise tolerance (ANT) [20, 21] and soft N-modular redundancy (soft NMR) [22]. ANT targets timing errors introduced by voltage overscaling. The architecture contains the main device, which is allowed to produce wrong outputs due to VOS and an estimator block which computes the same function as the main device, but produces a known systematic error. With the knowledge of the statistics of the correct output of the main

device and the systematic error introduced by the estimator block, the decision device picks either one of them such that the overall error is minimized [23].

Soft NMR differs from the classical NMR in the voter algorithm [22]. The voter in soft NMR treats the outputs of the N identical devices as multiple noisy measurements of some parameter. It then uses estimation theory to design decision rule, which optimizes some given cost function. It does not mask the error by majority voting but corrects the errors in the statistical sense. For example, if the error is known to have Gaussian distribution and the cost function is the minimum mean square error (MMSE), the corrected output of the soft NMR voter is simply the average value of the N outputs.

In terms of MIMO receivers, an algorithm that uses stochastic computing or a sort of soft NMR has been proposed in [24]. The outputs of M_r receive antennas are treated as outputs of N identical modules. Here, the compound noise term is formed that comprises the errors introduced by hardware and the errors due to noise in the communication channel. This compound noise is modeled by a mixture distribution, where the Gaussian distribution of the channel noise is picked with probability $1 - \epsilon$ and an arbitrary contaminant distribution is picked with probability ϵ. Next, robust statistics are used to seek for outliers and robust LMMSE estimate of the transmit symbol vector is obtained. Unfortunately, there are no details given on how to pick a particular contaminant distribution and how it relates to actual hardware induced errors. Also, no insights are provided on how to identify the outliers and most importantly, the performance of the method in terms of detection quality has not been assessed.

A related work has been performed by authors of [25]. Here, a model that combines channel noise and memory errors due to VOS has been proposed. Based on this model, they proposed modified detection algorithm that uses the combined model in deciding on the transmitted symbol [6]. However, their work addresses solely small-scale MIMO systems and therefore uses simplifying assumptions on memory error distribution. The proposed model, with necessary refinements, will be the used in this work to modify detection algorithms suitable for large-scale MIMO systems.

Other memory components of MIMO receiver have been addressed as well. It has been shown in [26] that when the hardware errors occur in the input buffers of the channel decoder it is still possible to recover even from high error rates by simply increasing the number of decoding iterations. The hardware errors in the log-likelihood ratio (LLR) buffer have been analyzed in [27]. These errors are not critical, as the LLR represents the confidence in the decoded bit (positive for 0 and negative for 1). Thus, it is essentially only necessary to protect the sign bit of the stored LLR value [28].

The channel preprocessing block turns out to be the most critical. It has been shown in [29] that the errors in the input buffer to the channel preprocessing drastically spoil the receivers' performance even at low error rates, due to propagation to successor blocks. In general, errors lead to mismatch between decomposed channel matrix and received vector, rendering further detection and decoding useless. The QR decomposition that implements channel preprocessing is a simple and stable algorithm when working with fixed point arithmetic. Apart from that stability it does not posses any inherent resilience. It is suggested in [28] to protect it by low level techniques. The actual computational components of the receiver implementation have not been yet addressed.

Cryptographic circuits and further security-related devices are vulnerable to malicious attacks by fault injection [30, 31]. A circuit which processes protected data is run while being subjected to physical disturbances, and the outcome of calculation, observed at the circuit's outputs, is analyzed. One such technique is differential fault analysis [32]: the difference between fault-affected and fault-free executions is used to derive the secret information stored in the device. A number of fault-injection techniques have been suggested, ranging from rather imprecise voltage or clock-frequency manipulation [30] to highly accurate pinpointed impact by laser pulses [33] or electromagnetic emissions [34]. Successful attacks have been reported for a variety of well-known ciphers [35–37], under different assumption of the attacker's capability to inject faults. While a number of techniques to advertently inject faults into a circuit are known [30], the most promising method with high spatial and temporal resolution uses a laser beam to generate carriers (electrons and holes) that induce parasitic currents and ultimately result in a bit flip. The latest generation of fault-based attacks against state-of-the-art ciphers [35, 38–41] requires only very few successful fault injections, but the exact time and location of each performed fault injection must be controlled with a very high precision.

Countermeasures mainly consist of making the device physically inaccessible or employing fault detection [30]. The device is put into a tamper-proof case with intrusion detection. Fault detection techniques included duplication / modular redundancy and respective variations thereof with voting on the outputs. The drawback of these methods is the large imposed overhead. Another approach is to apply error detecting codes. Usually linear codes are used, however more powerful nonlinear (robust) codes have been recently proposed [8].

1.3 Contributions

This thesis aims at achieving fault-tolerant large-scale MIMO communication systems, taking into account their specific properties. The individual chapters address different aspects of this global goal. The key contributions of this thesis are summarized below:

- Generalization of UW-OFDM to a MIMO system is performed. The resulting system is interpreted as a virtual massive MIMO system allowing application of quasi maximum likelihood detection – likelihood ascent search (LAS). It is found that encoding the individual data streams (relative to transmit antennas) by distinct non-systematic UW generator matrices improves LAS BER performance. The soft output LAS does not perform well in a MIMO UW-OFDM coded system. Therefore, the LAS algorithm is modified to take into account the log likelihood ratios provided by its initial solution. The resulting maximum a posteriori probability LAS and proposed soft output computation deliver better performance in a coded system than the original LAS algorithm. Thorough simulations of MIMO UW-OFDM and standard MIMO CP-OFDM are performed and their performance compared.

- A novel memory error resilient nonlinear MMSE-based detection algorithm for memory dominated massive MIMO system is proposed. The algorithm builds up on the linear MMSE detection with moderate additional computational effort.

The single-element single-bit flip assumption is relaxed, and the proposed solution targets the multiple-element single flip scenario. The simulation results show that even for high memory error rates, the proposed algorithm is able to significantly reduce the impact of the memory errors on the overall receiver performance in terms of BER.

- An architecture for MMSE sorted QR decomposition based on Givens rotations implemented by conventional fixed point arithmetic is introduced. It is derived directly from the classical systolic array implementation of conventional Givens rotations. To the best of the author's knowledge, this is the first non-Gram-Schmidt MMSE sorted QR decomposition implementation using conventional fixed point arithmetic that meets the real-time constraints imposed by the long-term evolution (LTE) standard.

- The first analysis of reliability of the implemented QR decomposition block by actually injecting faults into the hardware, whereas actual works assume solely memory errors. The computational sub-modules yielding the most performance degradation are identified and the degradation is quantified in terms of the frame error ratio (FER) floor.

- A cross-level protection solution, where a light-weight error-detecting code is combined with hardening of insufficiently protected gates using transistor resizing is proposed. Such gates are determined by FPGA-supported fault injection. A thorough electrical analysis is performed in order to modify the electrical parameters of these gates such that faults are highly unlikely. Area and power overhead for a number of error-detecting codes are reported. To the best of the author's knowledge, this is the first work which co-optimizes fault handling by information redundancy and by hardening individual circuit elements. In contrast to high-level error models considered previously, faults within the combinational logic of the circuit are considered and distinguished from errors, which are effects of faults at the circuit's outputs. The novel attack success rate (ASR) metric of a fault is proposed that indicates the difficulty to inject a fault that is not detected.

- Robust error-detecting codes that specifically target malicious attacks and guarantee minimal bounds on detection probability are systematically investigated. The study is based on FPGA-supported fault injection campaigns on the circuit implementation of a recent lightweight block cipher and its sub-modules. The detection capabilities of different robust and non-robust codes with respect to both random faults and malicious attacks, as well as the required overheads are quantified. For the first time, the performance of a novel punctured cubic code on actual cryptographic circuitry is reported. Experimental results show that robust codes with a certain number of redundant bits have better detection properties in security context and higher predictability than their conventional linear counterparts.

1.4 Outline

The rest of this thesis is organized as follows:

Chapter 2 provides necessary background on wireless communication and details cyclic prefix and unique word OFDM waveform design. Next, MIMO detection algorithms used as a basis in this work are introduced. The classical fault tolerance measures along with approaches specific for signal processing systems are shortly outlined. Finally, details on fault based attacks and common countermeasures are provided.

Chapter 3 deals with generalization of UW-OFDM to a MIMO system. Simulation results of detection algorithms specifically tailored for MIMO UW-OFDM are provided. The results of this research are accepted for publication in [42]. Thereafter, nonlinear MMSE-based detection algorithms are proposed and their performance is quantified by simulation for both OFDM waveforms and different memory bit flip probabilities. Main results of this work are submitted to [43].

Chapter 4 introduces own implementation of MMSE sorted Givens rotations and details architecture and synthesis results. The implementation is published in [44]. Next, fault injection experiments are performed that provide insights on most vulnerable computational components of the implementation. This work is published in [45].

Chapter 5 starts with background on robust codes. Then, fault-based metrics are introduced together with the fault-injection framework. Results on cross-level protection technique which combines robust codes with hardening of critical fault locations are provided. This research is published in [46]. Finally, application of robust codes to modern low-power cipher PRINCE is investigated. This work is published in [47].

Chapter 6 concludes the thesis. It puts its outcomes into perspectives, summarizes the achievements and indicates future research directions.

Chapter 2

Background

This chapter starts with theoretical background on the problem of communication through wireless multipath channels. The OFDM signaling which is deployed in current standards is introduced together with its two versions addressed in this thesis, namely cyclic prefix and unique word OFDM. Section 2.1.1 discusses the characteristics of multipath channels and introduces channel classification based on the latter. Here, the particular channel type that is used later in this thesis is isolated. Section 2.1.2 provides details on CP- and UW-OFDM signaling and highlights their differences. This section also introduces main building blocks of the transceiver such as channel coding and modulation. Next, in Sec. 2.2 MIMO CP-OFDM is introduced, followed by relevant detection algorithms. Finally, Sec. 2.3 provides background on fault tolerance mechanisms for contemporary signal processing systems, fault-based attacks and respective countermeasures.

2.1 Wireless communication

2.1.1 Multipath channels

Data transmission over wireless channels is obscured by physical phenomena of scattering, reflection and diffraction of radio waves. This situation is illustrated in Fig. 2.1. At the receiver, the receive signal is the superposition of the transmit signal copies associated with different propagation paths. The path depicted in solid is the line of sight path (LOS). Remaining paths are due to distinct scatterers or the clusters of scatterers. These paths are depicted by dashed lines, where the different dash patterns indicate different amount of delay with respect to the LOS path. In case the delays of the respective paths are close, they can be combined to a single propagation path with the same delay.

The modulated or bandpass transmit signal can be expressed as [48]

$$\acute{x}(t) = \Re\left\{x(t)e^{i(2\pi f_c t+\phi_0)}\right\} \tag{2.1}$$

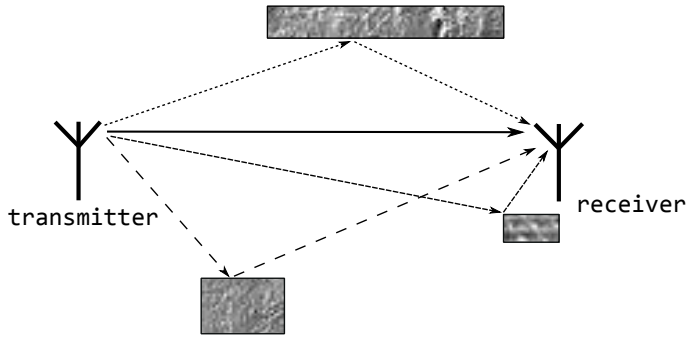


FIGURE 2.1: Multipath propagation

where $x(t)$ is the complex baseband equivalent [49], f_c is the carrier frequency and ϕ_0 is the initial phase offset.

Assume $\phi_0 = 0$. The bandpass receive signal is obtained as the superposition of delayed versions of $\acute{x}(t)$, coming from $L(t)$ propagation paths, which are characterized by their complex amplitudes $a_l(t)$ and delays $\tau_l(t)$, $l = 0, \ldots, L$:

$$\begin{aligned}
\acute{y}(t) &= \Re\left\{\sum_{l=0}^{L(t)} a_l(t)x(t-\tau_l(t))e^{i(2\pi(f_c+f_{d_l})(t-\tau_l(t)))}\right\} \\
&= \Re\left\{\sum_{l=0}^{L(t)} a_l(t)e^{i(-2\pi(f_c+f_{d_l})\tau_l(t)+2\pi f_{d_l}t)}x(t-\tau_l(t))e^{i2\pi f_c t}\right\} \\
&= \Re\left\{\left[\sum_{l=0}^{L(t)} a_l(t)e^{-i\phi_l(t)}x(t-\tau_l(t))\right]e^{i2\pi f_c t}\right\}
\end{aligned} \tag{2.2}$$

Looking at the bracketed term reveals that it can be regarded as the complex baseband equivalent $y(t)$ of the bandpass receive signal $\acute{y}(t)$. It can be concluded that the baseband representation of the receive signal is given as the superposition of scaled and phase shifted copies of the baseband transmit signal. Using the equivalent baseband representation allows removing the carrier frequency related term from the subsequent analysis. The phase shift $\phi_l(t)$ may cause the versions of the transmit signal $x(t)$ to cancel each other out at the receiver, causing severe receive power drops which is known as fading.

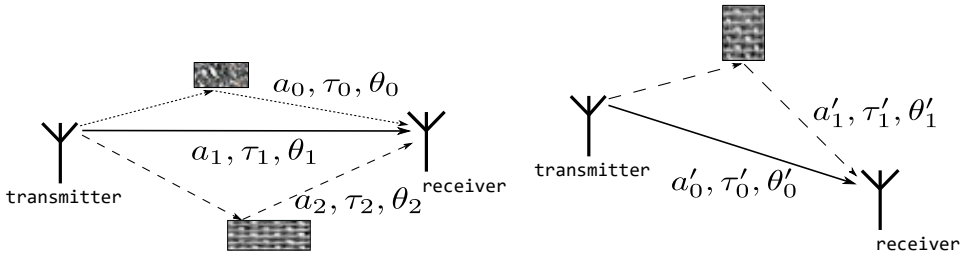
Finally, considering the above conclusions, the multipath channel is characterized by its impulse response

$$h(t,\tau) = \sum_{l=0}^{L(t)} a_l(t)e^{-i\phi_l(t)}\delta(\tau-\tau_l(t)) \tag{2.3}$$

where $\delta(\tau-\tau_l(t))$ is the Dirac delta function [50]. The phase shift $\phi_l(t)$ is formulated as

$$\phi_l(t) = 2\pi(f_c+f_{d_l})\tau_l(t) - 2\pi f_{d_l}t. \tag{2.4}$$

The channel impulse response (CIR) in Eq. 2.3 is time-varying. The complex amplitude $a_l(t)$ depends on the characteristics of reflecting/scattering materials and changes slowly



(A) Channel at time t_1 (B) Channel at time t_2

FIGURE 2.2: Time-varying multipath channel

over time relative to changes of the phase shift $\phi_l(t)$. This phase shift depends on the time-varying delays $\tau_l(t)$, where each delay is associated with a distinct propagation path l. The overall number L of propagation paths and the values of the respective delays change over time as the receiver moves around in the multipath environment. Consider an example in Fig. 2.2. At time t_1, there are propagation paths defined by their amplitudes a_l, delays τ_l and angles of arrival (AoA) $\theta_l(t)$. The $l = 0$ path corresponds to the LOS. The AoA is measured with respect to the antenna broadside. That is, if the transmitter and receiver are right in front (LOS path in Fig. 2.2a), AoA $= 0^o$. In the next time instance t_2, the receiver has moved to another location and now there are only two propagation paths. The path parameters α'_i, τ'_i, θ'_i are marked with prime to emphasize their time-varying nature.

The overall phase shift $\phi_l(t)$ also depends on the Doppler frequency f_{d_l}. The Doppler effect manifests itself as the change of the perceived carrier frequency at the receiver due to the relative movement of the transmitter, receiver and/or the multipath environment [48, 50]. The Doppler frequency quantifies the amount of this change. Consider the moving receiver depicted in Fig. 2.3 The receiver moves with speed v, and it requires

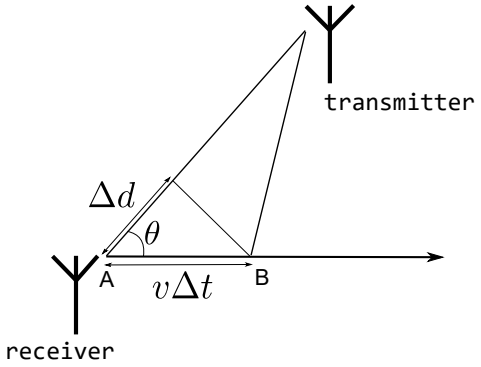


FIGURE 2.3: Doppler effect

the time Δt to move from point A to point B. The difference in distance which the wave has to travel is given by Δd, measured in wavelengths. The introduced phase shift due to Δd is obtained as

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda} = \frac{2\pi}{\lambda} v\Delta t \cos\theta \tag{2.5}$$

where λ is the wavelength and θ is the AoA. The Doppler frequency is then obtained from the given phase shift as

$$f_d = \frac{\Delta\phi}{2\pi\Delta t} = \frac{v}{\lambda}\cos\theta = f_c\frac{v}{c}\cos\theta \tag{2.6}$$

where f_c is the carrier frequency and c is the speed of light. Therefore, the Doppler frequency is the function of the carrier frequency, speed and AoA. It increases with increasing the carrier frequency. Doppler shift is also directly proportional to the receiver speed v. Given the carrier frequency and the receiver speed, the Doppler frequency is maximal if $\theta = 0^o$:

$$f_{d,max} = f_c\frac{v}{c} \tag{2.7}$$

The AoAs also change over time as the receiver moves around in the scattering environment as indicated in Fig. 2.2, thereby contributing to changing Doppler frequencies and associated phase shifts.

Therefore, the multipath channel can be parametrized with respect to delay τ and time t. The values of the parameters allow channel classification in terms of their frequency and temporal variation. These parameters and the respective classification are introduced next.

2.1.1.1 Delay spread

The time domain parameter related to τ is the delay spread T_d, which is defined as the difference between the arrival time of the transmit signal over the LOS propagation path and the arrival time of the last delayed transmit signal copy [48]. The value of the delay spread with respect to the symbol duration T_s indicates whether the channel is able to introduce inter-symbol interference (ISI). In case $T_s \ll T_d$, the delayed symbol versions arrive far outside the duration of the symbol and would affect the subsequent symbols, resulting in ISI. Conversely, if symbol duration is much longer than the delays spread, $T_s \gg T_d$, all echoes of the symbol still arrive within its duration and do not affect the subsequent symbols. In this case the ISI is negligible.

As the CIR of the channel is a random process, the delay spread is a random variable. It is obtained in terms of the auto-correlation function of the CIR. The CIR auto-correlation function is defined as

$$R(\tau_1, \tau_2, t, \Delta t) = E\left\{h^*(\tau_1, t)h(\tau_2, t + \Delta t)\right\} \tag{2.8}$$

In the general case, the auto-correlation function is computed for all possible combinations of t and τ. The CIR random process is considered wide sense stationary (WSS), meaning that its auto-correlation function does not depend on the time instant t at which it is measured, but depends solely on the time shift Δt [48]. Also the delays τ_1 and τ_2 of distinct multipath components are considered uncorrelated [48]. Therefore the auto-correlation function is rewritten as

$$R(\tau, \Delta t) = E\left\{h^*(\tau)h(\tau, \Delta t)\right\} \tag{2.9}$$

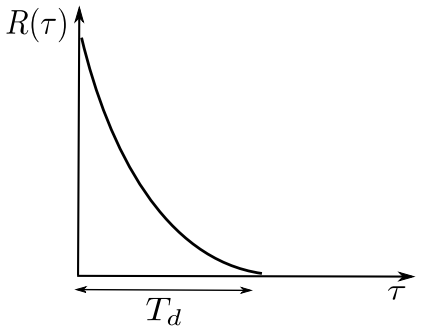


FIGURE 2.4: Power delay profile

The auto-correlation function defines the channel power with respect to the propagation delay τ and the time shift Δt. The power with respect to the propagation delay is maximal when $\Delta t = 0$. The CIR auto-correlation function in this case is referred to as the power delay profile (PDP)

$$R(\tau) = R(\tau, \Delta t = 0) \tag{2.10}$$

The PDP shows the distribution of power as a function of delay τ. The PDP provides information on how much power is concentrated in each of the propagation paths that are associated to the distinct values of τ. In case some paths are conveying very low power their contribution to the ISI will be negligible. Therefore they can be excluded from further processing. This way the complexity of reducing ISI is eased. The mean delay $\bar{\tau}$ and the root mean square delay spread σ_τ are obtained form PDP as

$$\mu_{T_d} = \frac{\int_0^\infty \tau R(\tau) d\tau}{\int_0^\infty R(\tau) d\tau} \tag{2.11}$$

and

$$\sigma_{T_d} = \sqrt{\frac{\int_0^\infty (\tau - \mu_{T_d})^2 R(\tau) d\tau}{\int_0^\infty R(\tau) d\tau}} \tag{2.12}$$

Consider an exponentially distributed PDP depicted in Fig. 2.4. The value of T_d for which $R(\tau) \approx 0$ for all $\tau > T_d$ is referred to as the delay spread. In this case usually the standard deviation of the PDP is chosen as the value of the delay spread T_d. In case of exponentially distributed PDP, the mean and standard deviation are the same, therefore any one of the moments can be chosen to represent the delay spread.

2.1.1.2 Coherence bandwidth

In the frequency domain the parameter related to τ is the coherence bandwidth [48]. In order to capture the time-varying channel in frequency domain, the Fourier transform

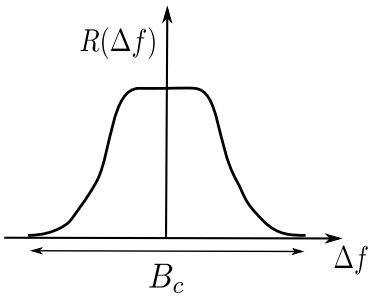


FIGURE 2.5: Spaced-frequency correlation function

of its CIR is performed with respect to τ.

$$H(f,t) = \int_{-\infty}^{\infty} h(\tau,t)e^{-i2\pi f\tau}d\tau \tag{2.13}$$

Its auto-correlation function is formulated as

$$R(f_1,f_2,t,\Delta t) = E\left\{H^*(f_1,t)H(f_2,t+\Delta t)\right\} \tag{2.14}$$

Assuming wide-sense stationary and uncorrelated scattering, the auto-correlation function is simplified as

$$\begin{aligned} R(f_1,f_2,\Delta t) &= E\left\{\int_{-\infty}^{\infty} h^*(\tau_1,t)e^{i2\pi f_1\tau_1}d\tau_1 \int_{-\infty}^{\infty} h(\tau_2,t+\Delta t)e^{-i2\pi f_2\tau_2}d\tau_2\right\} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E\left\{h^*(\tau_1,t)h(\tau_2,t+\Delta t)\right\} e^{i2\pi f_1\tau_1}e^{-i2\pi f_2\tau_2}d\tau_1 d\tau_2 \\ &= \int_{-\infty}^{\infty} R(\tau,\Delta t)e^{-i2pi(f_2-f_1)\tau}d\tau \\ &= R(\Delta f,\Delta t) \end{aligned} \tag{2.15}$$

where $\Delta f = f_2 - f_1$. Therefore, the auto-correlation of the channel transfer function $H(f,t)$ in frequency depends only on the frequency difference Δf. At $\Delta t = 0$ this function is referred to as the spaced-frequency correlation function and is defined as the Fourier transform of the PDP.

$$R(\Delta f) = \int_{-\infty}^{\infty} R(\tau)e^{-i2\pi\Delta f\tau}d\tau \tag{2.16}$$

Consider the spaced-frequency correlation function depicted in Fig. 2.5. It is obtained from the exponential PDP in Fig. 2.4. The frequency range B_c, where $R(\Delta f) = 0$ for any $\Delta f > B_c$ is referred to as the coherence bandwidth. In other words, coherence bandwidth B_c is the range of frequencies for which the amount of fading can be considered identical. As the coherence bandwidth is obtained from the PDP, it is proportional to the inverse of the delay spread: $B_c \approx \frac{1}{T_d}$.

The multipath channel can be classified with respect to the discussed parameters as follows [48–50]:

- *Frequency flat channel.* With respect to the coherence bandwidth, the channel is considered frequency flat if $B_s \ll B_c$. This means that the whole bandwidth occupied by the signal undergoes the same amount of fading. Conversely, in time domain this implies $T_s \gg T_d$. This condition ensures that the amount of ISI introduced by the channel is negligible. In particular, the symbol duration should be at least an order of magnitude larger that the delay spread [48].
- *Frequency selective channel.* The channel is considered frequency selective if $B_s \gg B_c$. Thereby, the different portions of the signal bandwidth undergo different distortion introduced by the channel. Conversely, in time domain the frequency selectivity implies $T_s \ll T_d$. In particular, if the symbol duration is an order of magnitude less than the delay spread, a frequency selective channel will introduce significant ISI.

A communication system can be classified as narrow-band or wide-band only with respect to coherence bandwidth or the delay spread of the channel. Given T_d, increasing the T_s with respect to it would result in decreasing the signal bandwidth B_s. Decreasing T_s relative to T_d increases the signal bandwidth B_s and introduces ISI. Therefore, systems referred to as wide-band are considered to be affected by ISI. Consider a realization of the channel transfer function illustrated in Fig. 2.6. The signal bandwidth B_{s_1} spreads across two significant power drops (fades). This indicates that the signal bandwidth most likely exceeds the channels coherence bandwidth. The signal bandwidth B_{s_2} is on the contrary so narrow, that the channel transfer function is almost constant across it. This indicates that B_{s_2} is very likely much narrower than B_c. In this case, for the communication system with B_{s_2} the channel can be considered a flat fading channel, whereas the same channel is considered frequency selective for the system with B_{s_1}.

OFDM communication systems considered in this thesis are classified as wide-band and are therefore passing the data through a frequency selective channel that introduces ISI. The ISI is handled by introduction of the cyclic prefix. This process will be explained in subsequent sections. Along with the parameters related to τ, the parameters related to the time evolution of the multipath channel are of interest.

2.1.1.3 Doppler power spectrum

The first parameter related to t is the Doppler power spectrum. Consider the scattering function which is defined as the Fourier transform of the CIR auto-correlation function with respect to Δt

$$S(\tau, f_d) = \int_{-\infty}^{\infty} R(\tau, \Delta t) e^{-i2\pi f_d \Delta t} d\Delta t \tag{2.17}$$

The scattering function identifies the average power of the multipath channel with respect to the delay τ and Doppler shift f_d [48]. The Doppler power spectrum is obtained

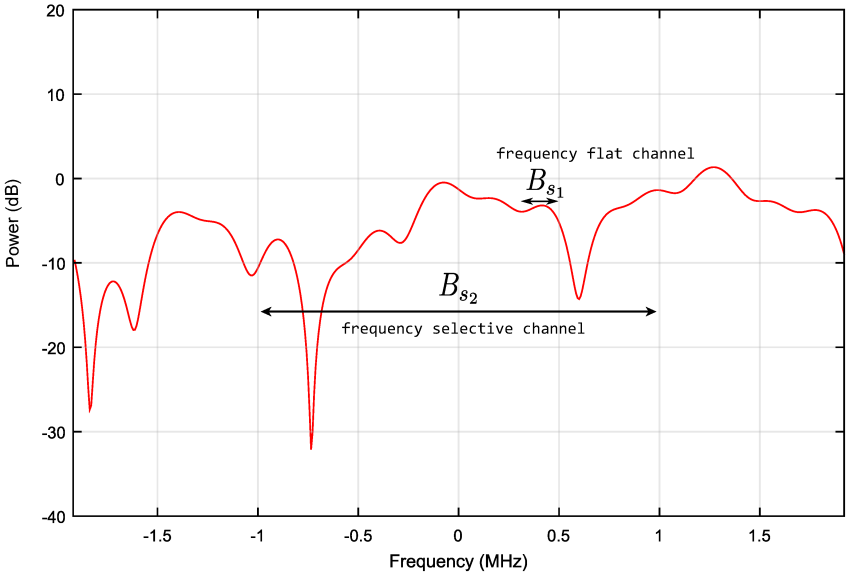


FIGURE 2.6: Fading channel frequency response

by integrating the scattering function over the whole range of delays

$$S(f_d) = \int_{-\infty}^{\infty} S(\tau, f_d) d\tau \tag{2.18}$$

It should be noted that the PDP can be also obtained by integrating the scattering function over the range of Doppler shifts [48, 50].

$$R(\tau) = \int_{-\infty}^{\infty} S(\tau, f_d) df_d \tag{2.19}$$

The frequency range B_d for which $|S(f_d)|$ is greater than zero is the Doppler spread of the channel. As the maximum Doppler frequency can either increase or decrease the f_c, the Doppler spread is given as $B_d = 2f_d$.

2.1.1.4 Coherence time

The Doppler power spectrum can also be obtained from the channel transfer function auto-correlation function (Eq. 2.15) [48, 50]

$$S(\Delta f, f_d) = \int_{-\infty}^{\infty} R(\Delta f, \Delta t) e^{-i2\pi f_d \Delta t} d\Delta t \tag{2.20}$$

In order to characterize the Doppler shift at a single frequency, frequency shift $\Delta f = 0$ is set to zero

$$S(f_d) = \int_{-\infty}^{\infty} R(\Delta t) e^{-i2\pi\ f_d \Delta t} d\Delta t \tag{2.21}$$

The auto-correlation function $R(\Delta t)$ shows the variation of the channel with respect to time t. The range of values of t for which $R(\Delta t)$ is nonzero is denoted the coherence time T_c. The value of coherence time gives the time frame in which the CIR is considered constant. Obviously, the coherence time is approximately inversely proportional to the Doppler spread of the channel: $T_c \approx \frac{1}{B_d}$.

With respect to the introduced time-related channel parameters, the multipath channels are classified as follows [48, 50]:

- *Slow-fading channel.* If the channel varies slower than the symbol duration, $T_s \ll T_c$ it is considered slow fading. As coherence time is inversely proportional to the Doppler spread, slow fading implies that $B_s \gg B_d$. For example, in the 802.11n channel model [51], the receiver and the transmitter are considered stationary and people and/or objects are moving between them. The maximal speed of the environment was picked $v = 1.2$ km/h in order to fit the Doppler power spectrum measurement results [51]. The maximum Doppler frequency is therefore $f_d = 3$ Hz at $f_c = 2.4$ GHz. The channel coherence time is then equal to $T_c = \frac{1}{2f_d} = 0.17$ s. The symbol duration in 802.11n is $T_s = 0.061$ μs which is much shorter than the T_c. It can be concluded that the 802.11n channel is a slow fading channel.
- *Fast-fading channel.* In case the coherence time is less than the symbol duration, $T_s > T_c$, or respectively $B_s < B_d$, the channel is considered fast varying or fast fading channel.

2.1.1.5 Frequency selective channel model

It is convenient to model the frequency selective channel as the discrete-time tapped delay line. It is derived as outlined next [49].

Consider sampling of the transmit signal according to the sampling theorem [52]:

$$x(t) = \sum_{l=-\infty}^{\infty} x\left(lT\right) \frac{\sin\left(\frac{\pi(t-lT)}{T}\right)}{\frac{\pi(t-lT)}{T}} \tag{2.22}$$

where T is the sampling period. According to the sampling theorem, the sampling frequency must be at least twice higher than the highest frequency present in the $x(t)$. The Fourier transform of the transmit signal is given as

$$X(f) = \begin{cases} T \sum\limits_{l=-\infty}^{\infty} x(lT) e^{-i2\pi flT} & , |f| \leq \frac{1}{2T} \\ 0 & , \text{otherwise} \end{cases} \tag{2.23}$$

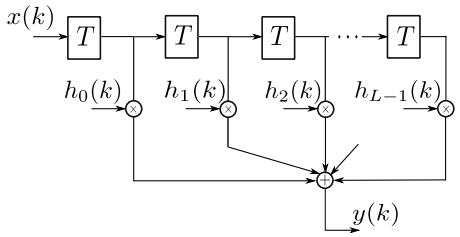


FIGURE 2.7: Tapped delay line

Using Eq. 2.2 and the fact that the convolution in time domain results in multiplication in frequency domain, the baseband equivalent of the receive signal is given as

$$
\begin{aligned}
y(t) &= \int_{-\infty}^{\infty} H(f,t)X(f)e^{i2\pi ft}df \qquad (2.24)\\
&= \int_{-\infty}^{\infty} H(f,t)T\sum_{l=-\infty}^{\infty} x(lT)e^{i2\pi f(t-lT)}df\\
&= T\sum_{l=-\infty}^{\infty} x(lT)\left(\int_{-\infty}^{\infty} H(f,t)e^{i2\pi f(t-lT)}df\right)\\
&= T\sum_{l=-\infty}^{\infty} x(lT)h((t-lT),t)\\
&= T\sum_{l=-\infty}^{\infty} h(lT,t)x(t-lT)\\
&= \sum_{l=-\infty}^{\infty} (Th(lT,t))x(t-lT)
\end{aligned}
$$

Now, the delay τ is given in terms of the discrete intervals lT. Assuming the delay spread of T_d, the number of taps L in the delay line is given as $L = T_d/T$. Assuming discrete time $t = kT$, $k = 0, 1\ldots$ and setting $T = 1$, the receive signal is finally given as

$$y(k) = \sum_{l=0}^{L-1} h_l(k)x(k-l) \qquad (2.25)$$

The resulting tapped delay line model is depicted in Fig. 2.7. The channel coefficients h_l are modeled as complex numbers, where both the real and imaginary parts are zero mean Gaussian distributed random variables [49, 53]. This is reasonable, because each channel coefficient at particular delay l is produced by a superposition of a large number of random events.

The frequency selective channel is transformed to more attractive frequency flat channel in current standards by OFDM. This technique will be dealt with in subsequent sections.

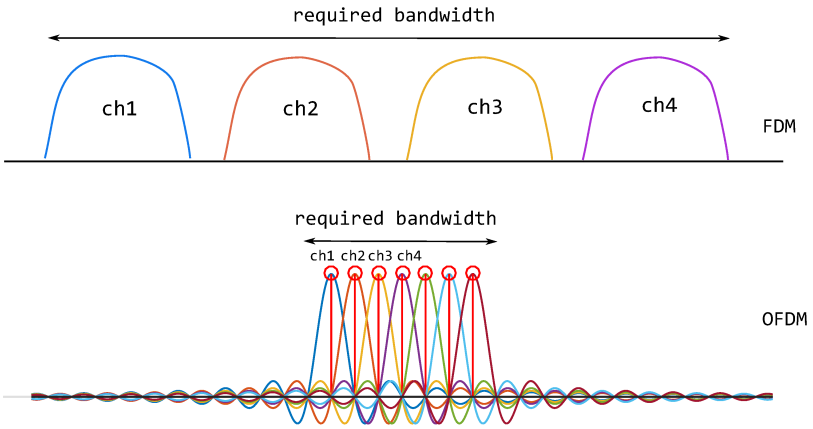


FIGURE 2.8: OFDM vs FDM

2.1.2 Orthogonal frequency division multiplexing

Multiplexing defines how the available resources are allocated to the users [48]. In particular, each user can be provided a time slot as in Time Division Multiplexing. Alternatively a user can be identified by its specific orthogonal code as in Code Division Multiplexing. Lastly, in Frequency Division Multiplexing (FDM), data corresponding to distinct sub-channels is mapped to a portion of the available bandwidth. In OFDM, in contrast to the single-carrier transmission, the data is mapped to orthogonal sub-carriers. Due to overlapping spectrum of orthogonal frequencies, the required bandwidth is reduced. Because of orthogonality, there is no interference between sub-carriers, and no guard bands are required as opposed to FDM. The difference between two multiplexing schemes is highlighted in Fig. 2.8.

If a signal experiences a deep fade, only a fraction of sub-carriers is affected, as illustrated in Fig. 2.9. In frequency selective channels OFDM offers much better BER than FDM [49].

The transmit data is physically a time domain signal and is organized in N parallel streams. Each of these streams occupies one of the N orthogonal sub-carriers. Next, a combined time domain signal is obtained for transmission through the channel. This signal is obtained by the inverse fast Fourier transform (IFFT) of size N. Here, IFFT is just a mathematical tool to map the N "spectral" components to a combined time domain signal. At the receiver, the N data streams are extracted by fast Fourier transform (FFT) from the combined time domain signal. The individual sub-carrier symbols are obtained.

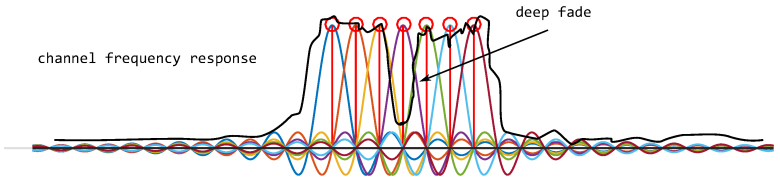


FIGURE 2.9: OFDM in a deep fade

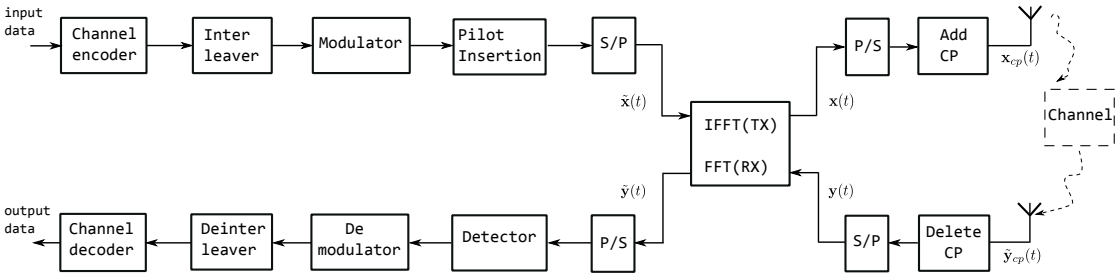


FIGURE 2.10: CP-OFDM transceiver

2.1.2.1 Cyclic prefix OFDM

An OFDM single antenna communication system model featuring the standard components is depicted in Fig. 2.10. The rest of this section will guide the reader through the system and explain the relevant transmit/receive building blocks in detail.

2.1.2.1.1 Encoder / decoder.

The user generated binary data is first encoded by a linear error correcting code. The linear codes can be classified as block or convolutional. The standard rate $1/2$ $(171, 133)$ convolutional code [54] is used in simulations performed in this thesis.

The encoding introduces redundancy which is exploited at the receiver in order to correct bit errors. Linear codes are powerful against single errors. In case of error bursts, their performance is significantly degraded. Interleaving is employed to cope with burst errors [54–56]. An interleaver scrambles the bit positions within the codeword according to some rule. The interleaved codeword is transmitted through a channel, which can introduce a burst of errors (Fig. 2.11). Such situation is common for a wireless channel, as a deep fade over a codeword duration renders the latter completely incorrect. At the receiver, the codeword is deinterleaved, effectively spreading the multiple errors within a burst over a number of codewords. This results in a number of codewords each having just a single error, which can be effectively corrected by the linear code (Fig. 2.11).

The soft-decision Viterbi decoding is used in all simulations in this thesis if not stated otherwise. The details on the construction and decoding of convolutional codes are outlined in Appendix A.

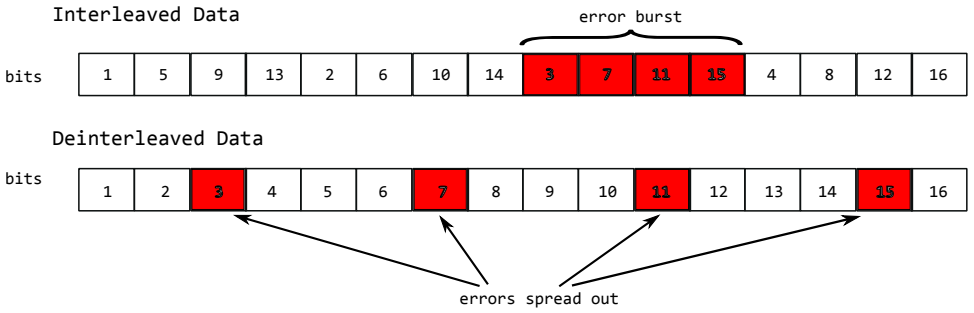


FIGURE 2.11: Interleaving

The performance of a communication system is measured at the output of the channel decoder in terms of BER for a range of SNR values. The SNR is denoted as the ratio of bit energy E_b to the additive white Gaussian noise (AWGN) power spectral density N_0.

2.1.2.1.2 Modulation / demodulation

Next, the encoded binary data has to be mapped to physical signals (sine and cosine) for transmission through the air. This task is performed by modulation. Modulation defines the mapping of bits to a symbol. A symbol is a combination of sine and cosine waves with a given amplitude and phase shift [49]. A modulation scheme is defined by the modulation alphabet $\mathbb{A}$. For example, 4-quadrature amplitude modulation (4-QAM) [49] alphabet contains following symbols: $\mathbb{A} = \{-1 - 1i, -1 + 1i, +1 + 1i, +1 - 1i\}$. The size of modulation alphabet $|\mathbb{A}|$ defines the number of bits n_b that are mapped to a modulation symbol through a following relation:

$$n_b = \log_2(|\mathbb{A}|) \tag{2.26}$$

That is, in case of 4-QAM, two bits are mapped to each of the four allowed symbols. The mapping of four bit tuples $\{00, 01, 10, 11\}$ to a particular symbol can be done in a number of ways. One of the possible mappings is illustrated in Fig. 2.12

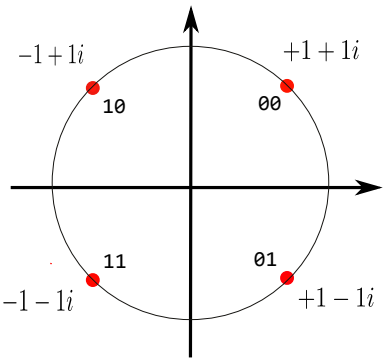


FIGURE 2.12: 4-QAM Gray mapping

The depicted constellation diagram represents the Gray-mapped 4-QAM modulation [49]. It can be observed, that bit 0 is mapped to +1 and bit 1 is mapped to −1. Therefore, the modulation operation can be expressed in vector form as

$$x = \mathbf{v}^T \mathbf{c} \tag{2.27}$$

where x is the resulting symbol, $\mathbf{v} = \begin{bmatrix} 1 & i \end{bmatrix}^T$ in case of 4-QAM and $\mathbf{c}$ is the (code) bit tuple where bit values are mapped to $\{-1, +1\}$ as discussed above.

The main advantage of Gray mapping is that the nearest symbols differ in only one bit. At the receiver, the symbols affected by AWGN have to be mapped back to the symbols from the modulation alphabet. This operation is referred to as demodulation. The scatter plot of 4-QAM Gray-mapped constellation is depicted in Fig. 2.13. It shows the noisy symbols as blue crosses. These noisy symbols are quantized to the nearest symbols from $\mathbb{A}$ in the Euclidean distance sense.

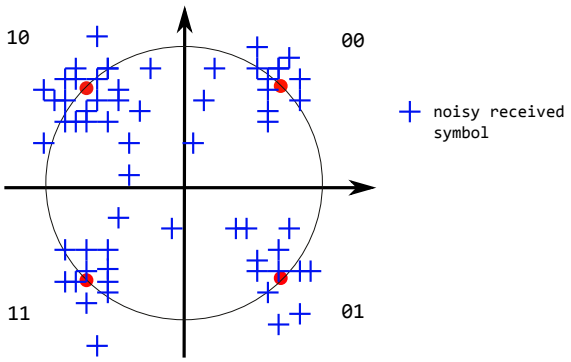


FIGURE 2.13: 4-QAM scatter plot

Assume that the symbol $11 \rightarrow -1-1i$ was transmitted. The wrong misinterpretation as $-1+1i$ or $+1-1i$ due to noise is more likely than the misinterpretation to $+1+1i$ because of the largest Euclidean distance between $-1-1i$ and $+1+1i$. Therefore, with Gray mapping, a symbol error results in a single bit error. It is easy to verify that it is not the case for a different mapping.

Next, the vector of modulated data symbols, forming the t-th OFDM symbol vector, is defined in frequency domain as $\tilde{\mathbf{x}}(t) \in \mathbb{C}^{N\times 1}$, where N is the number of orthogonal sub-carriers. The vector of time domain symbols is formed by applying IFFT of size N.

$$\mathbf{x}(t) = \begin{bmatrix} x(tN) \\ \vdots \\ x(tN+N-1) \end{bmatrix}_{N\times 1} = \frac{1}{N}\mathbf{F}_N^H \tilde{\mathbf{x}}(t) \tag{2.28}$$

where $\frac{1}{N}\mathbf{F}_N^H$ is the IFFT matrix of size $N \times N$. The IFFT matrix is given as

$$\frac{1}{N}\mathbf{F}_N^H = \frac{1}{N}\begin{bmatrix} f_{00} & f_{10} & \cdots & f_{(N-1)0} \\ f_{01} & f_{11} & \cdots & f_{(N-1)1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{0(N-1)} & f_{1(N-1)} & \cdots & f_{(N-1)(N-1)} \end{bmatrix} \tag{2.29}$$

with $f_{nk} = e^{j\frac{2\pi}{N}nk}$, $n,k = 0,\ldots,N-1$. The FFT matrix is denoted $\mathbf{F}_N$.

2.1.2.1.3 Cyclic prefix

Although OFDM sub-carriers are orthogonal to each other, the channel may introduce inter-symbol interference in time domain as explained in Sec. 2.1.1. The reflected versions of the $(t-1)$-th time domain symbol vector may arrive at the receiver exactly at same time the t-th time domain symbol vector arrives at the receiver when channel delay spread is larger than the symbol duration. In order to mitigate the ISI, OFDM time domain symbol vector $\mathbf{x}(t)$ is cyclically extended, by copying last N_g data symbols to the front. This is illustrated in Fig. 2.14. With N_g picked larger or equal to the maximum delay spread of the channel, the ISI lands exactly within the duration of the cyclic prefix and does not affect the user data.

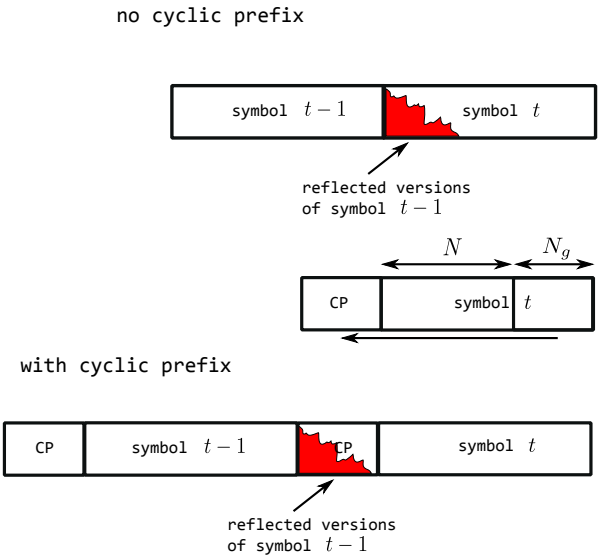


FIGURE 2.14: Cyclic prefix

The addition of the cyclic prefix can be expressed as a matrix operation

$$\mathbf{x}_{cp}(t) = \mathbf{A}_{cp}\mathbf{x}(t) \tag{2.30}$$

where

$$\mathbf{A}_{cp} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{N_g} \\ \mathbf{I}_{N-N_g} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_g} \end{bmatrix}_{(N+N_g)\times N}$$

is the cyclic prefix insertion matrix. The length of the cyclically extended OFDM symbol vector is now given in data symbols by $N_{tot} = N + N_g$.

The cyclically extended OFDM symbol vector is then transmitted through the quasi-static frequency selective fading channel [49]. "Quasi-static" means that the CIR is constant during the transmission of the whole t-th OFDM symbol vector. The multipath spread of the channel is characterized by maximum delay spread L, measured in data symbols. As described in 2.1.1, this channel is modeled as a tap delay line with L taps. Hence, in order to combat ISI, N_g has to be picked larger or equal to L. The CIR vector $\mathbf{h}(t)$ is

$$\mathbf{h}(t) = \begin{bmatrix} h_0(t) \\ \vdots \\ h_{L-1}(t) \end{bmatrix}_{L\times 1}$$

where the index t is kept to highlight that the channel may vary from one OFDM symbol vector to the next. The k-th ($0 \leq k \leq N_{tot} - 1$) received data symbol from the t-th OFDM symbol vector is given by the discrete time convolution with the CIR vector:

$$y_{cp}(tN_{tot} + k) = \sum_{l=0}^{L-1} h_l(t)x_{cp}(tN_{tot} + k - l) + n(tN_{tot} + k) \tag{2.31}$$

Here, $n(tN_{tot}+k)$ is the element of the AWGN noise vector, and is zero mean with variance $\frac{1}{2}{\sigma_n}^2$ per complex dimension.

Consider an OFDM symbol vector at time $t=1$. The 0-th received data symbol from that OFDM symbol vector is obtained from Eq. 2.31 as

$$\begin{aligned} y_{cp}(N_{tot}+0) = h_0(1)x_{cp}(N_{tot}-0) + h_1(1)x_{cp}(N_{tot}-1) + \cdots \\ \cdots + h_{L-1}(1)x_{cp}(N_{tot}-(L-1)) + n(N_{tot}+0) \end{aligned} \tag{2.32}$$

Obviously, the terms with index less than N_{tot} are coming from the previous $(t-1)$-th OFDM symbol vector and are representing the ISI. For the t-th receive OFDM symbol vector convolution with CIR is expressed in matrix form as

$$\mathbf{y}_{cp}(t) = \mathbf{H}_{toep}(t)\mathbf{x}_{cp}(t) + \mathbf{H}^{ISI}_{toep}(t)\mathbf{x}_{cp}(t-1) + \mathbf{n}(t) \tag{2.33}$$

where $\mathbf{H}_{toep}(t)$ and $\mathbf{H}^{ISI}_{toep}(t)$ of size $N_{tot} \times N_{tot}$ are Toeplitz matrices [57], defined as follows

$$\mathbf{H}_{toep}(t) = \begin{bmatrix} h_0(t) & & & & & \\ \vdots & \ddots & & & & \\ h_{L-1}(t) & \cdots & h_0(t) & & & \\ & \ddots & \vdots & \ddots & & \\ & & h_{L-1}(t) & \cdots & h_0(t) \end{bmatrix} \tag{2.34}$$

$$\mathbf{H}^{ISI}_{toep}(t) = \begin{bmatrix} h_{L-1}(t) & \cdots & h_1(t) \\ & \ddots & \vdots \\ & & h_{L-1}(t) \\ & & \\ & & \end{bmatrix} \tag{2.35}$$

The removal of ISI terms $\mathbf{H}^{ISI}_{toep}(t)$ is performed by cutting off the first N_g data symbols of the OFDM symbol vector $\mathbf{y}_{cp}(t)$. In matrix form this operation is expressed as

$$\mathbf{y}(t) = \begin{bmatrix} y(tN) \\ y(tN+N-1) \end{bmatrix}_{N\times 1} = \mathbf{A}_{DEcp}\mathbf{y}_{cp}(t) \tag{2.36}$$

where

$$\mathbf{A}_{DEcp} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_N \end{bmatrix}_{N\times N_{tot}} \tag{2.37}$$

is the cyclic prefix removal matrix.

Hence, the frequency domain receive symbol vector is given by

$$\begin{aligned}
\tilde{\mathbf{y}}(t) &= \mathbf{F}_N \mathbf{y}(t) \\
&= \mathbf{F}_N(\mathbf{A}_{DEcp}\mathbf{y}_{cp}(t)) \\
&= \mathbf{F}_N(\mathbf{A}_{DEcp}(\mathbf{H}_{toep}(t)\mathbf{x}_{cp}(t) + \mathbf{H}_{toep}^{ISI}(t)\mathbf{x}_{cp}(t-1) + \mathbf{n}(t))) \\
&= \mathbf{F}_N(\mathbf{A}_{DEcp}\mathbf{H}_{toep}(t)\mathbf{x}_{cp}(t) + \mathbf{A}_{DEcp}\mathbf{n}(t)) \\
&= \mathbf{F}_N(\hbar\mathbf{A}_{DEcp}\mathbf{H}_{toep}(t)\mathbf{A}_{cp}\frac{1}{N}\mathbf{F}_N^H\tilde{\mathbf{x}}(t) + \mathbf{A}_{DEcp}\mathbf{n}(t)) \\
&= \frac{1}{N}\mathbf{F}_N(\mathbf{A}_{DEcp}\mathbf{H}_{toep}(t)\mathbf{A}_{cp})\mathbf{F}_N^H\tilde{\mathbf{x}}(t) + \mathbf{F}_N\mathbf{A}_{DEcp}\mathbf{n}(t) \\
&= \frac{1}{N}(\mathbf{F}_N\mathbf{H}_{cir}(t)\mathbf{F}_N^H)\tilde{\mathbf{x}}(t) + \tilde{\mathbf{n}}(t)
\end{aligned} \tag{2.38}$$

The matrix $\mathbf{H}_{cir}(t) = \mathbf{A}_{DEcp}\mathbf{H}_{toep}(t)\mathbf{A}_{cp}$ is a circulant matrix [57] and has the following form

$$\mathbf{H}_{cir}(t) = \begin{bmatrix}
h_0(t) & 0 & \cdots & 0 & \cdots & h_{L-1}(t) & h_{L-2}(t) & \cdots & h_1(t) \\
h_1(t) & h_0(t) & 0 & \cdots & 0 & 0 & h_{L-1}(t) & \cdots & h_2(t) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
h_{L-2}(t) & \cdots & \cdots & h_0(t) & 0 & \cdots & \cdots & 0 & h_{L-1}(t) \\
h_{L-1}(t) & \cdots & \cdots & \cdots & h_0(t) & 0 & \cdots & \cdots & 0 \\
0 & h_{L-1}(t) & \cdots & \cdots & \cdots & h_0(t) & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & 0 & h_{L-1}(t) & h_{L-2}(t) & \cdots & \cdots & h_0(t)
\end{bmatrix}_{N\times N} \tag{2.39}$$

As can be seen from Eq. 2.39, the circulant matrix is a Toeplitz matrix where each row is rotated one element to the right, relative to the preceding row. Circulant matrices possess following properties [57]:

1. The eigenvectors of any circulant matrix are the columns of $\mathbf{F}_N^H$
2. The eigenvalues of any circulant matrix equal the FFT of the first column of the matrix itself. That is

$$\begin{bmatrix} \tilde{h}_0 \\ \tilde{h}_1 \\ \vdots \\ \tilde{h}_{N-1} \end{bmatrix} = \mathbf{F}_N \begin{bmatrix} h_0 \\ \vdots \\ h_{L-1} \\ \mathbf{0}_{(N-L)\times 1} \end{bmatrix}$$

Regarding these properties the eigenvalue decomposition of $\mathbf{H}_{cir}(t)$ is given as

$$\mathbf{H}_{cir}(t) = \frac{1}{N}\mathbf{F}_N^H \begin{bmatrix} \tilde{h}_0 & & \\ & \ddots & \\ & & \tilde{h}_{N-1} \end{bmatrix} \mathbf{F}_N = \frac{1}{N}\mathbf{F}_N^H \tilde{\mathbf{H}}_d(t)\mathbf{F}_N \tag{2.40}$$

where $\tilde{\mathbf{H}}_d$ is the diagonal matrix with elements $\tilde{h}_k = \sum_{l=0}^{L-1} h_l e^{-j\frac{2\pi}{N}kl}$, with $0 \leq k \leq N$.

Plugging Eq. 2.40 into Eq. 2.38 and dropping the OFDM symbol vector index t, the system equation for single antenna or single-input single-output (SISO) CP-OFDM is obtained as follows:

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{F}_N \frac{1}{N}\mathbf{F}_N^H \tilde{\mathbf{H}}_d \mathbf{F}_N \frac{1}{N}\mathbf{F}_N^H \tilde{\mathbf{x}} + \tilde{\mathbf{n}} \\ &= \tilde{\mathbf{H}}_d \tilde{\mathbf{x}} + \tilde{\mathbf{n}} \end{aligned} \tag{2.41}$$

Hence, for each of the N orthogonal sub-carriers the frequency selective channel is transformed to a frequency flat channel with single tap weight $\tilde{h}_k$. The data is easily recovered by multiplying the received frequency domain symbol vector by the inverse of the diagonal channel matrix $\tilde{\mathbf{H}}_d$

$$\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{H}}_d^{-1} \tilde{\mathbf{y}} \tag{2.42}$$

The estimated symbol data is then passed to channel decoding.

2.1.2.2 Unique word OFDM

OFDM systems require insertion of a guard interval (GI) of duration T_{GI} into the time domain data frame in order to combat inter-symbol interference caused by multipath propagation. Conventional OFDM systems use the cyclic prefix, which is simply a copy of a data fraction, and is therefore random. On the contrary, UW-OFDM signaling uses a deterministic sequence in the GI, denoted as the *unique word* [9]. Frame structures of CP-OFDM compared with UW-OFDM are outlined in Fig. 2.15.

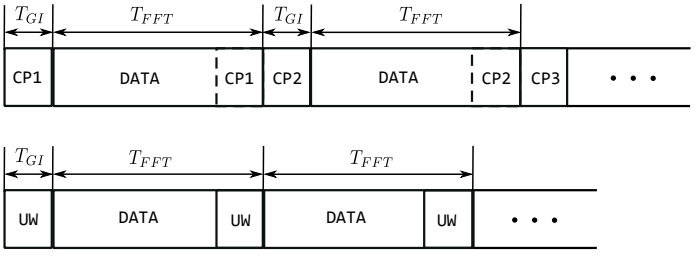


FIGURE 2.15: UW vs CP frame structure

Since UW is a known sequence, it can be used for synchronization or channel estimation purposes. The spectral efficiency of UW system remains unaffected. It can be observed from Fig. 2.15 that UW is the part of the FFT interval of duration T_{FFT}, whereas CP is not. Therefore, the length of the UW-OFDM frame is reduced from $T_{FFT} + T_{GI}$ to T_{FFT}, thus retaining the overall spectral efficiency [10].

More importantly, UW generation introduces a form of coding across sub-carriers [58]. UW can be generated by both systematic and non-systematic encoding [11].

2.1.2.2.1 Systematic generation of UW

In UW-OFDM the available $N = N_d + N_r$ sub-carriers are shared by data and redundant symbols [9]. The N_d sub-carriers are occupied by data symbols and the remaining N_r sub-carriers are dedicated to redundant symbols. Therefore the frequency domain symbol vector $\tilde{\mathbf{x}} \in \mathbb{C}^{N\times 1}$ can be denoted as consisting of data and redundant parts $\tilde{\mathbf{x}} = \begin{bmatrix}\tilde{\mathbf{d}}^H & \tilde{\mathbf{x}}_r^H\end{bmatrix}^H$, where $\tilde{\mathbf{d}}^H \in \mathbb{C}^{N_d\times 1}$ is the part reserved for data, and $\tilde{\mathbf{x}}_r^H \in \mathbb{C}^{N_r\times 1}$ contains redundancy. UW is generated in two steps [9]:

1. A zero UW is generated, such that the time domain OFDM symbol vector is given as $\mathbf{x} = \begin{bmatrix}\mathbf{x}_d^H & \mathbf{0}\end{bmatrix}^H$ and $\mathbf{x} = \frac{1}{N}\mathbf{F}_N^H\tilde{\mathbf{x}}$.
2. The nonzero UW $\mathbf{u}^H \in \mathbb{C}^{N_r\times 1}$ is added in the time domain resulting in $\mathbf{x}' = \mathbf{x} + \begin{bmatrix}\mathbf{0} & \mathbf{u}^H\end{bmatrix}^H$.

The redundancy is added during the first step, and it is considered in detail here. The IFFT on the frequency domain symbol vector must yield

$$\frac{1}{N}\mathbf{F}_N^H\mathbf{P}\begin{bmatrix}\tilde{\mathbf{d}}\\ \tilde{\mathbf{x}}_r\end{bmatrix} = \begin{bmatrix}\mathbf{x}_d\\ \mathbf{0}\end{bmatrix} \tag{2.43}$$

Here, $\mathbf{P}$ is the introduced permutation matrix that allocates data and redundant sub-carriers in such a way that energy contribution of the redundant sub-carrier symbols is minimal. The matrix in Eq. 2.43 can be renamed as

$$\frac{1}{N}\mathbf{F}_N^H\mathbf{P} = \mathbf{M} = \begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{21} & \mathbf{M}_{22}\end{bmatrix}$$

where $\mathbf{M}_{ij}$ are sub-matrices of an appropriate size. From matrix multiplication in Eq. 2.43

$$\mathbf{M}_{21}\tilde{\mathbf{d}} + \mathbf{M}_{22}\tilde{\mathbf{x}}_r = \mathbf{0}$$

The generation of redundant sub-carrier symbols $\tilde{\mathbf{x}}_r$ from the data symbols follows directly as

$$\tilde{\mathbf{x}}_r = \mathbf{T}\tilde{\mathbf{d}} \tag{2.44}$$

where $\mathbf{T} = -\mathbf{M}_{22}^{-1}\mathbf{M}_{21}$, $\mathbf{T} \in \mathbb{C}^{N_r\times N_d}$.

In style of block coding theory, the frequency domain symbol vector $\tilde{\mathbf{x}}$ can be interpreted as a code word of a complex RS (Reed-Solomon)-code [58]:

$$\tilde{\mathbf{x}} = \mathbf{P}\begin{bmatrix}\tilde{\mathbf{d}}\\ \tilde{\mathbf{x}}_r\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{I}\\ \mathbf{T}\end{bmatrix}\tilde{\mathbf{d}} = \mathbf{G}\tilde{\mathbf{d}} \tag{2.45}$$

where $\mathbf{G} \in \mathbb{C}^{(N_d+N_r)\times N_d}$ is the code generator matrix that introduces correlations within the symbol vector $\tilde{\mathbf{x}}$. The interpretation as the RS code is valid, because it is defined as

a set of codewords whose Fourier transform produces a consecutive set of zeroes. This is exactly the case in Eq. 2.43 with the only difference that the encoding is performed on the set of complex numbers [11].

The second step adds the UW in the time domain. The transmit UW-OFDM symbol vector is now denoted $\mathbf{x}' = \mathbf{x} + \begin{bmatrix}\mathbf{0}^H & \mathbf{u}^H\end{bmatrix}^H$. The frequency domain version $\tilde{\mathbf{u}} \in \mathbb{C}^{N_r \times 1}$ of the unique word is obtained as

$$\tilde{\mathbf{u}} = \mathbf{F}_N \begin{bmatrix}\mathbf{0} \\ \mathbf{u}\end{bmatrix}$$

This allows the transmit time domain symbol vector to be rewritten as

$$\mathbf{x}' = \frac{1}{N}\mathbf{F}_N^H(\tilde{\mathbf{x}} + \tilde{\mathbf{u}}) = \frac{1}{N}\mathbf{F}_N^H(\mathbf{G}\tilde{\mathbf{d}} + \tilde{\mathbf{u}}) \tag{2.46}$$

Next, $\mathbf{x}'$ is transmitted over the same multipath channel, characterized by the CIR vector $\mathbf{h}(t) = \begin{bmatrix} h_0(t) \\ \vdots \\ h_{L-1}(t)\end{bmatrix}_{L\times 1}$, as indicated in Sec. 2.1.2.1.

Consider two time domain transmit symbol vectors

$$\begin{bmatrix}\mathbf{x}'(t-1) \\ \mathbf{x}'(t)\end{bmatrix} = \begin{bmatrix}\mathbf{d}(t-1) \\ \mathbf{u}(t-1) \\ \mathbf{d}(t) \\ \mathbf{u}(t)\end{bmatrix} \tag{2.47}$$

It can be recognized that the ISI terms caused by of $\mathbf{x}'(t-1)$ fall within the duration of UW vector $\mathbf{u}(t-1)$.

The receive symbol vector is the result of the discrete convolution of the CIR and the transmit symbol vector, which is expressed as multiplication with $\mathbf{H}_{toep}(t)$ as given in Eq. 2.35:

$$\mathbf{y}(t) = \mathbf{H}_{toep}(t)\begin{bmatrix}\mathbf{u}(t-1) \\ \mathbf{d}(t) \\ \mathbf{u}(t)\end{bmatrix} + \mathbf{n}(t) \tag{2.48}$$

The structure of the transmitted UW-OFDM frame is now equivalent to the frame with added cyclic prefix in case of CP-OFDM. The inclusion of $\mathbf{u}(t-1)$ is modeled by multiplication with a copy matrix $\mathbf{\Theta}_c$, similar to cyclic prefix insertion matrix given in Eq. 2.1.2.1.3:

$$\mathbf{\Theta}_c = \begin{bmatrix}\mathbf{0} & \mathbf{I}_{N_r} \\ \mathbf{I}_{N-N_r} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_r}\end{bmatrix} \tag{2.49}$$

The t-th time domain receive symbol vector is given as

$$\mathbf{y}(t) = \mathbf{\Theta}_x \mathbf{H}_{toep}(t) \mathbf{\Theta}_c \mathbf{x}'(t) \tag{2.50}$$

where $\mathbf{\Theta}_x$ is the UW extraction matrix similar to cyclic prefix removal matrix given in Eq. 2.37:

$$\mathbf{\Theta}_x = \begin{bmatrix} \mathbf{0}_{N \times N_r} & \mathbf{I}_N \end{bmatrix} \tag{2.51}$$

Clearly, the matrix given below

$$\mathbf{H}_{cir}(t) = \check{\mathbf{\Theta}}_x \mathbf{H}_{toep}(t) \check{\mathbf{\Theta}}_c \tag{2.52}$$

is the same block circulant matrix as in Eq. 2.39, as the UW insertion and extraction matrices transform the block convolution matrix in the same way as the CP insertion and removal matrices. Therefore, the block eigenvalue decomposition of $\mathbf{H}_{cir}$ is the same as in the CP-OFDM case.

Dropping the time index t for simplicity, the receive symbol vector is therefore reformulated as

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{F}_N \mathbf{H}_{cir} \mathbf{x}' + \mathbf{F}_N \mathbf{n} \\ &= \frac{1}{N} \mathbf{F}_N \mathbf{H}_{cir} \mathbf{F}_N^H (\mathbf{G}\tilde{\mathbf{d}} + \tilde{\mathbf{u}}) + \mathbf{F}_N \mathbf{n} \end{aligned} \tag{2.53}$$

Using the eigenvalue decomposition of the circulant matrix in Eq. 2.40, the system equation for UW-OFDM is given as

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}_d \mathbf{G}\tilde{\mathbf{d}} + \tilde{\mathbf{H}}_d \tilde{\mathbf{u}} + \tilde{\mathbf{n}} \tag{2.54}$$

where $\tilde{\mathbf{H}}_d$ is the same diagonal matrix as in Eq. 2.41, containing flat fading tap coefficients on the main diagonal. Subtracting the known component $\tilde{\mathbf{H}}_d \tilde{\mathbf{u}}$ from the frequency domain receive symbol vector $\tilde{\mathbf{y}}$ yields the final UW-OFDM system equation

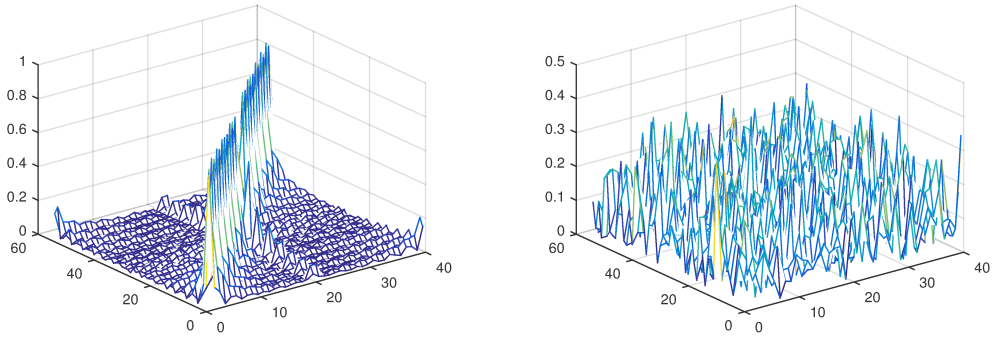
$$\begin{aligned} \tilde{\mathbf{y}} - \tilde{\mathbf{H}}_d \tilde{\mathbf{u}} &= \tilde{\mathbf{H}}_d \mathbf{G}\tilde{\mathbf{d}} + \tilde{\mathbf{H}}_d \tilde{\mathbf{u}} + \tilde{\mathbf{n}} - \tilde{\mathbf{H}}_d \tilde{\mathbf{u}} \\ \tilde{\mathbf{y}}' &= \tilde{\mathbf{H}}_d \mathbf{G}\tilde{\mathbf{d}} + \tilde{\mathbf{n}} \end{aligned} \tag{2.55}$$

2.1.2.2.2 Non-systematic generation of UW

Opposed to systematic UW generation, where the redundancy is placed at dedicated sub-carriers, non-systematic UW generation spreads the redundancy among all sub-carriers [11]. The generator matrix is now denoted $\bar{\mathbf{G}}$, and is given as

$$\bar{\mathbf{G}} = \mathbf{A}\mathbf{P} \begin{bmatrix} \mathbf{I} \\ \bar{\mathbf{T}} \end{bmatrix} \tag{2.56}$$

where $\mathbf{A} \in \mathbb{R}^{(N_d+N_r)\times(N_d+N_r)}$ is non-singular, and permutation matrix $\mathbf{P}$ is same as with the systematic UW generation. The generator matrix still has to produce zeroes at the



(A) $\mathbf{G}'$ (B) $\mathbf{G}''$

FIGURE 2.16: Non-systematic generator matrices

positions of N_r symbols in the time domain analogous to Eq. 2.43.

$$\frac{1}{N}\mathbf{F}_N^H\mathbf{A}\mathbf{P}\begin{bmatrix}\mathbf{I}\\ \bar{\mathbf{T}}\end{bmatrix}\tilde{\mathbf{d}} = \begin{bmatrix}\mathbf{x}_d\\ \mathbf{0}\end{bmatrix} \tag{2.57}$$

Introducing $\bar{\mathbf{M}} = \frac{1}{N}\mathbf{F}_N^H\mathbf{A}\mathbf{P} = \begin{bmatrix}\bar{\mathbf{M}}_{11} & \bar{\mathbf{M}}_{12}\\ \bar{\mathbf{M}}_{21} & \bar{\mathbf{M}}_{22}\end{bmatrix}$, Eq. 2.57 can be fulfilled by choosing $\bar{\mathbf{T}} = -\bar{\mathbf{M}}_{22}^{-1}\bar{\mathbf{M}}_{21}$. The optimum value of matrix $\mathbf{A}$, such that Eq. 2.57 is fulfilled, is obtained in [11] by the steepest descent algorithm. Two initializations are proposed:

1. Initialization with identity matrix

$$\mathbf{A}^{(0)} = \mathbf{I} \tag{2.58}$$

   implying $\bar{\mathbf{T}}^{(0)} = \mathbf{T}$ and

$$\bar{\mathbf{G}}^{(0)} = \mathbf{P}\begin{bmatrix}\mathbf{I} & \mathbf{T}^H\end{bmatrix}^H = \mathbf{G} \tag{2.59}$$

   Therefore, iterative search starts with the systematic generator matrix, which is considered a reasonable choice. After optimization, the resulting generator matrix is obtained and denoted $\mathbf{G}'$.

2. Random initialization. Each element of $\mathbf{A}^{(0)}$ is a Gaussian random variable with zero mean and variance one.

$$a_{ij}^{(0)} \sim N(0,1) \tag{2.60}$$

   The resulting generator matrix is in this case denoted $\mathbf{G}''$.

The matrices $\mathbf{G}'$ and $\mathbf{G}''$ are shown in Fig. 2.16. It can be observed that since $\mathbf{G}'$ is a perfection of the systematic generator matrix $\mathbf{G}$ it exhibits a band structure. The energy of data symbols is spread locally within the codeword. In contrast, $\mathbf{G}''$ spreads the energy of data symbol almost uniformly over the codeword [11]. This is implied by the full matrix structure of $\mathbf{G}''$. The system with $\mathbf{G}''$ can be regarded as single carrier system, where the symbol energy is distributed uniformly across the available bandwidth. In contrast, a system with $\mathbf{G}'$ still behaves as an OFDM system. The authors of [11]

have shown that due to these properties, the system with $\mathbf{G}''$ outperforms the system with $\mathbf{G}'$ in a scenario where no outer channel code is used. In a coded system, however, the UW-OFDM system with $\mathbf{G}'$ outperforms the one with $\mathbf{G}''$ in terms of BER.

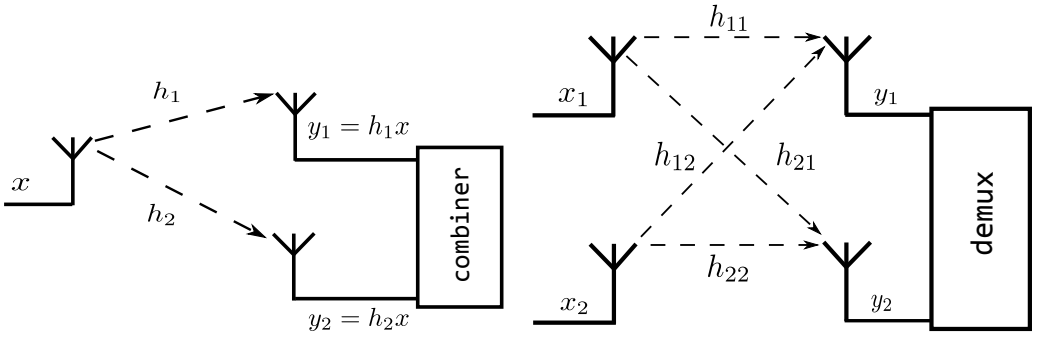
As the obtained generator matrices distribute portions of a single data symbol over a number of codeword symbols they can be interpreted as a combination of channel independent precoder and a channel encoder [11]. Since $\mathbf{G}''$ is obtained with random initialization, it is possible to obtain multiple random generator matrices. These random $\mathbf{G}''$ generator matrices introduce another degree of freedom to the system in Eq. 2.55.

2.2 MIMO

The multiple-input multiple-output or MIMO is referred to multiple inputs to the channel from a number of transmit antennas and multiple outputs from the channel to a number of receive antennas respectively. MIMO systems have been researched thoroughly over the past decade [59–62] and are now part of most wireless standards [51, 63]. Systems with multiple antennas at either or both ends of the communication link can be subdivided in two classes [62]:

- The ones that try to overcome multipath fading effects by providing diversity.
- The others that exploit multipath fading by performing spatial multiplexing.

Systems employing multiple antennas at either the transmitter (MISO) or the receiver (SIMO) provide diversity and have been used already in the last century in order to improve the communication reliability [64, 65]. The purpose of these systems is to overcome the effects of fading by combining multiple independent versions of the same transmit symbol. The receive diversity is illustrated in Fig. 2.17a. Note that in case of



(A) Receive Diversity (B) Spatial Multiplexing

FIGURE 2.17: Diversity vs spatial multiplexing

just one antenna at both sides of the link the transmit signal can be severely degraded. The M_r versions of the same transmitted signal will very likely undergo different amount of fading. Then a gain in BER is achieved by combining these received copies. The well-known types of receive diversity combining are selective combining, equal gain combining (EGC) and maximum ratio combining (MRC) [66–68]. With selective combining the

copy with the largest channel coefficient is picked for the output. With EGC, the copies of the transmitted signal are added together. With MRC, the copies are scaled according to the values of channel coefficients and added to produce the output.

Transmit diversity is referred to combining symbols from multiple transmit antennas at a single receive antenna. The transmit symbols can not be identical copies, as the receive antenna would get simply a scaled superposition of the same transmit symbol. A form of space-time coding has to be employed at the transmitter. This is realized by space-time codes [69, 70].

Systems with multiple antennas on both the receiver and transmitter side (MIMO) can provide either diversity or spatial multiplexing [71]. This thesis addresses the latter and leaves the diversity topic out of scope.

Spatial multiplexing is illustrated in Fig. 2.17b. It offers throughput increase, as ideally M_t data streams can be transmitted simultaneously [72] over same time slot and frequency band. The operation of spatial multiplexing is explained next.

The generic spatial multiplexing MIMO system equation is given below

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{2.61}$$

The transmit data is composed of M_t streams forming transmit symbol vector $\mathbf{x}$. Each transmission path between any transmit/receive antenna pair (i, j) is characterized by a channel coefficient h_{ij}. These coefficients form the channel matrix $\mathbf{H}$ which is assumed to be known at the receiver side. In reality, this matrix is estimated from pilot signals by channel estimation, but it is out of scope of this work. The vector of receive symbols is denoted $\mathbf{y}$ and $\mathbf{n}$ is the vector of AWGN. It should be noted that, as system equation is generic, superscripts denoting OFDM frequency domain are omitted. In later sections, detection algorithms used in this work will be first explained for this generic system equation. Only then they will be cast for CP- or UW-OFDM.

The goal of the receiver in a spatial multiplexing MIMO system is to retrieve the transmit symbol vector $\mathbf{x}$, knowing $\mathbf{y}$ and $\mathbf{H}$. Assuming no noise at the receiver for simplicity and square $\mathbf{H}$, the transmit vector is obtained as

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{y} \tag{2.62}$$

The unambiguous solution exists only if the channel matrix contains linearly independent rows and is invertible. This is the case with very high probability, when elements of the channel matrix are independently identically distributed (i.i.d) [73]. Physically, this situation corresponds to existence of a large number of scatters and a lot of independent transmission paths. In that sense the multipath environment is exploited at the receiver and the transmitted data can be recovered. However, the i.i.d channel model represents an ideal case and is used for computation of upper bounds on achievable throughput.

In reality, there is some amount of spatial correlation between the channel responses observed by adjacent antennas. This correlation reduces the achievable throughput of a spatial multiplexing system [62]. It is therefore necessary to use the accurate channel

model to obtain realistic simulation results. The MIMO channel model is elaborated in detail in the next section.

2.2.1 Channel model

A number of models for MIMO channels have been proposed recently [74]. Geometric models are based on the layout of physical surroundings (scatterers). Parametric models abstract physical layout and propagation in order to describe the channel by a set of paths with statistically known AoA and angle of departure (AoD) characteristics. These angles are defined as the azimuth angle of incoming or departing plane wave with respect to the broadside of the receive or transmit antenna. Correlation based models specify the spatial correlation between the channel matrix entries. A number of standardized models has been developed that share the aspects of the aforementioned model classes [73].

The UW-OFDM single-antenna system has been assessed against the 802.11a standard [10]. Consequently, the related MIMO supporting CP-OFDM standard 802.11n is addressed in this work. The standardized 802.11n channel model [75] will be therefore used in simulations throughout this chapter.

The purpose of a MIMO channel model is to specify the correlation between adjacent antennas on both the transmitter and receiver side. Each channel impulse response between any antenna pair is a superposition of numerous propagation paths. Therefore channel matrix elements are complex Gaussian distributed random variables, where $\mathbf{h}$ is vectorized version of $\mathbf{H}$:

$$p(\mathbf{h}) = \frac{1}{\pi^{M_t N_r} det\mathbf{C}_H} \exp(-\mathbf{h}^H \mathbf{C}_H^{-1} \mathbf{h}) \tag{2.63}$$

The matrix $\mathbf{C}_H$ specifies the correlation between the elements of $\mathbf{h}$. Zero correlation reduces the model to independent identically distributed variables. In that case $\mathbf{C}_H = \sigma^2\mathbf{I}$. The i.i.d model has been used initially in MIMO channel modeling and is still useful for identifying upper bounds on system performance.

The complex correlation coefficient between two receive antennas i and j is defined in the following. The channel coefficients at two receive antennas are denoted as h_{mi} and h_{mj}, assuming transmit path m is fixed. Assuming a frequency selective channel, the correlation coefficient for each of the L tap delays is

$$c_{Tx_m}^{Rx_{ij}} = \frac{E\{h_{mi}h_{mj}^*\} - E\{h_{mi}\}E\{h_{mj}^*\}}{\sqrt{\sigma_{h_{mi}}^2 \sigma_{h_{mj}}^2}} \tag{2.64}$$

As the channel matrix entries are random variables with zero mean and unit variance [76], the correlation coefficient reduces to

$$c_{Tx_m}^{Rx_{ij}} = E\{h_{mi}h_{mj}^*\} \tag{2.65}$$

The spatial correlation between adjacent antennas depends on AoA, AoD statistics and power azimuth spectrum (PAS) [77]. PAS identifies the distribution of the received

power as a function of AoA. The 802.11n model defines several clusters of scatterers characterized by same mean AoA and mean AoD. For each cluster, PAS is found to follow the Laplacian distribution [78]. Another parameter impacting the spatial correlation is the angular spread, which is obtained from PAS. First the PAS is normalized in order to fulfill the properties of a probability density function.

$$P_{norm}(\varphi) = \frac{P(\varphi)}{\int P(\varphi)d\varphi} \tag{2.66}$$

The angular spread is then defined as the square root of the second central moment of PAS, where φ_o is the mean AoA.

$$\sigma_a = \sqrt{\int (\varphi - \varphi_0)^2 P(\varphi)d\varphi} \tag{2.67}$$

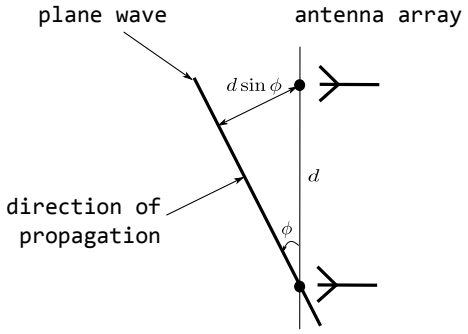


FIGURE 2.18: Reception by two neighboring antennas

The PAS and spatial correlation coefficient form a Fourier transform pair. Therefore, the correlation coefficient can be obtained as IFFT of PAS. The other method of computation of spatial correlation between two adjacent antennas is to directly obtain the correlation between their channel impulse responses, given the PAS.

Consider the receive antenna array, consisting of antennas a and b depicted in Fig. 2.18. The antennas are separated by a distance d. Assuming plane wave propagation, the wave requires to travel additional $d\sin\varphi$ distance to reach antenna b, with respect to antenna a. The channel impulse response at both antennas is obtained from PAS as given in Eq. 2.68 and 2.69, where ψ is some initial phase.

$$h_a(\varphi) = \int_{-\pi}^{\pi} e^{j\psi}\sqrt{P(\varphi)}d\varphi \tag{2.68}$$

$$h_b(\varphi) = \int_{-\pi}^{\pi} e^{j(\psi+\frac{2\pi d\sin\varphi}{\lambda})}\sqrt{P(\varphi)}d\varphi \tag{2.69}$$

The correlation between the two responses is then obtained as

$$c(d,\varphi_0) = E\{h_a(\varphi)h_b^*(\varphi)\} = \int_{-\pi}^{\pi} e^{-\frac{j2\pi d\sin\varphi}{\lambda}}P(\varphi)d\varphi \tag{2.70}$$

$$= c_{rr}(d,\varphi_0) + jc_{ri}(d,\varphi_0) \tag{2.71}$$

where c_{rr} is the correlation between the real parts of channel impulse responses and c_{ri} is the correlation between the real and the imaginary parts. In the extreme case of $\varphi_0 = 0^o$ and $\sigma_a = 0^o$, it is evident that the correlation is maximal. The wave arrives perpendicular to the array and does not incur any time difference between the antennas, while zero angular spread means that there is no angular variation in the received power. Defining the normalized distance between antennas as $D = \frac{2\pi d}{\lambda}$ the spatial correlation coefficients are finally obtained as

$$c_{rr}(D, \varphi_0) = \int_{-\pi}^{\pi} \cos(D \sin \varphi) P(\varphi) d\varphi \tag{2.72}$$

$$c_{ri}(D, \varphi_0) = \int_{-\pi}^{\pi} \sin(D \sin \varphi) P(\varphi) d\varphi \tag{2.73}$$

It can be observed from Eq. 2.72 and 2.73 that the spatial correlation depends on the distance between the antennas: the larger the distance, the less correlation. It also depends on the angular spread. When angular spread is small, most power would arrive to both antennas from the same angle. This would imply that the magnitudes of the responses at both antennas are almost identical, only with the difference in phase, due to antenna separation D, leading to high correlation.

After correlation coefficients are obtained for both transmitter and receiver, the transmit and receive correlation matrices $\mathbf{C}_{Tx}$ and $\mathbf{C}_{Rx}$ are formed. Thereafter the 802.11n model applies to the Kronecker model [79] to form the final channel matrix, with spatial correlation taken into account.

The Kronecker model assumes that the correlation matrices at the transmitter and receiver are separable. The overall correlation matrix equals the Kronecker product of transmit and receive correlation matrices.

$$\mathbf{C}_H = \mathbf{C}_{Tx} \otimes \mathbf{C}_{Rx} \tag{2.74}$$

Finally the channel matrix is computed as follows, where $\mathbf{H}_{i.i.d}$ is the matrix with uncorrelated coefficients.

$$\mathbf{H} = \mathbf{C}_{Rx}^{1/2} \mathbf{H}_{i.i.d} \mathbf{C}_{Tx}^{1/2} \tag{2.75}$$

The computation in Eq. 2.75 is repeated for each tap delay.

2.2.2 MIMO cyclic prefix OFDM

Consider the MIMO CP-OFDM transceiver model depicted in Fig. 2.19. The generalization of CP-OFDM signal model to the MIMO case is straight forward. There are M_t parallel data symbol vectors $\tilde{\mathbf{x}}_m$, $1 \leq m \leq M_t$ that are transmitted simultaneously over M_t transmit antennas. Hence, on each k-th sub-carrier, $k = 0, \ldots, N-1$, the M_t data symbols from respective data symbol vectors are transmitted simultaneously over M_t

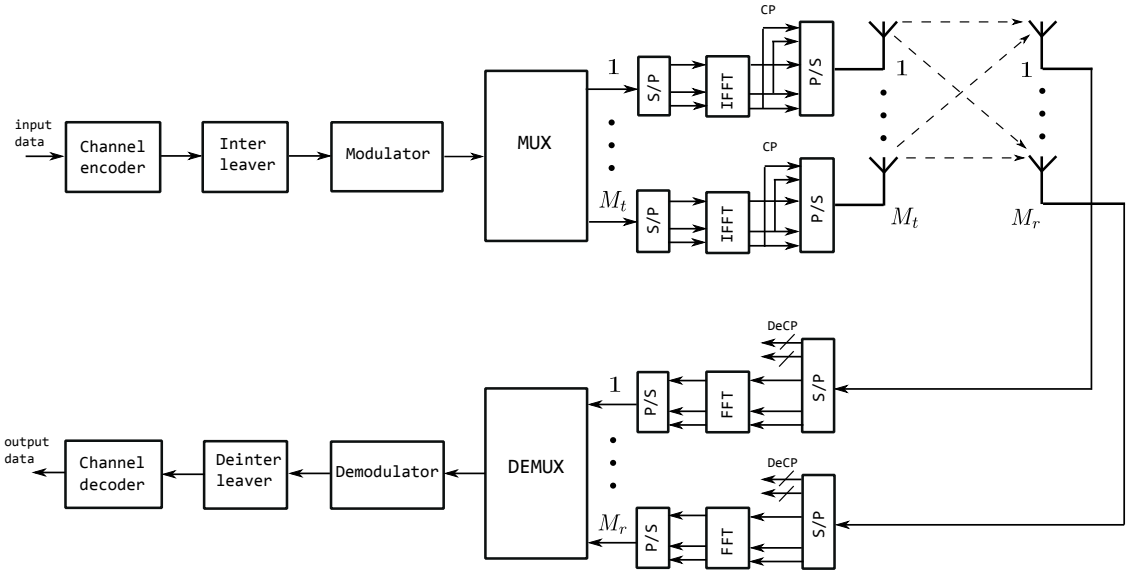


FIGURE 2.19: MIMO CP-OFDM transceiver

antennas. Therefore, $\tilde{\mathbf{x}}(k)$ is now a vector, given as

$$\tilde{\mathbf{x}}(k) = \begin{bmatrix} \tilde{x}_1(k) \\ \vdots \\ \tilde{x}_{M_t}(k) \end{bmatrix}_{M_t \times 1} \tag{2.76}$$

Here, the compound vector of transmitted symbols with respect to sub-carriers and transmit antennas is denoted as

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{x}}(0) \\ \vdots \\ \tilde{\mathbf{x}}(N-1) \end{bmatrix}_{M_t N \times 1} \tag{2.77}$$

where $\tilde{\mathbf{x}}(k)$ is from Eq. 2.76.

Each data stream fed to the respective transmit antenna is transformed to the time domain by a IFFT block. The M_t IFFT blocks work in parallel as outlined in Fig. 2.19. Mathematically, this parallel operation is expressed as the Kronecker product of the IFFT matrix and the identity matrix of size M_t - $\frac{1}{N}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})$. As the data on sub-carriers and transmit antennas is regarded as a single compound vector $\tilde{\mathbf{x}}$, the compound time domain vector $\mathbf{x}$ is formulated as

$$\mathbf{x} = \frac{1}{N}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\tilde{\mathbf{x}} \tag{2.78}$$

In more detail:

$$\mathbf{x} = \frac{1}{N} \underbrace{\begin{bmatrix} f_{11}^H \mathbf{I}_{M_t} & f_{12}^H \mathbf{I}_{M_t} & \cdots & f_{1N}^H \mathbf{I}_{M_t} \\ f_{21}^H \mathbf{I}_{M_t} & f_{22}^H \mathbf{I}_{M_t} & \cdots & f_{2N}^H \mathbf{I}_{M_t} \\ \vdots & \vdots & \vdots & \vdots \\ f_{N1}^H \mathbf{I}_{M_t} & f_{N2}^H \mathbf{I}_{M_t} & \cdots & f_{NN}^H \mathbf{I}_{M_t} \end{bmatrix}}_{M_t N \times M_t N} \underbrace{\begin{bmatrix} \tilde{x}_1(0) \\ \vdots \\ \tilde{x}_{M_t}(0) \\ \\ \tilde{x}_1(1) \\ \vdots \\ \tilde{x}_{M_t}(1) \\ \\ \vdots \\ \\ \tilde{x}_1(N-1) \\ \vdots \\ \tilde{x}_{M_t}(N-1) \end{bmatrix}}_{M_t N \times 1} \tag{2.79}$$

The insertion of the cyclic prefix is expressed as multiplication by the $\check{\mathbf{A}}_{cp}$ matrix analogous to Sec. 2.1.2.1

$$\mathbf{x}_{cp} = \check{\mathbf{A}}_{cp} \mathbf{x} \tag{2.80}$$

where $\check{\mathbf{A}}_{cp}$ is the SISO cyclic prefix insertion matrix augmented to the appropriate size by the Kronecker product with the identity matrix of size M_t

$$\check{\mathbf{A}}_{cp} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{N_g} \\ \mathbf{I}_{N-N_g} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_g} \end{bmatrix} \otimes \mathbf{I}_{M_t} \tag{2.81}$$

The multipath channel between the i-th transmit and j-th receive antennas is modeled as a $L-1$ tapped delay line with coefficients $h_{ij}(l)$, $l = 0, \ldots, L-1$ generated from the 802.11n channel model. Therefore, in MIMO case, the CIR is a matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 \\ \vdots \\ \mathbf{H}_{L-1} \end{bmatrix} \tag{2.82}$$

where the sub-matrices $\mathbf{H}_l$ contain the channel coefficients of all paths between the M_t transmit and M_r receive antennas relative to the tap delay l:

$$\mathbf{H}_l = \begin{bmatrix} h_{11}(l) & \cdots & h_{1M_t}(l) \\ \vdots & \ddots & \vdots \\ h_{M_r 1}(l) & \cdots & h_{M_r M_t}(l) \end{bmatrix}$$

Again, to remove ISI, first $M_r N_g$ elements of receive OFDM time domain symbol vector $\mathbf{y}_{cp}$ are removed analogous to the single-antenna CP-OFDM by multiplying with CP removal matrix $\check{\mathbf{A}}_{DEcp}$

$$\mathbf{y} = \check{\mathbf{A}}_{DEcp}\mathbf{y}_{cp} \tag{2.83}$$

where $\check{\mathbf{A}}_{DEcp}$ is the SISO CP removal matrix augmented to appropriate size by the Kronecker product with the identity matrix of size M_r

$$\check{\mathbf{A}}_{DEcp} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_N \end{bmatrix} \otimes \mathbf{I}_{M_r} \tag{2.84}$$

At the receiver side, the compound frequency domain received symbol vector $\tilde{\mathbf{y}} \in C^{NM_r \times 1}$ is given analogous to Eq. 2.38:

$$\begin{aligned} \tilde{\mathbf{y}} &= (\mathbf{F}_N \otimes \mathbf{I}_{M_r})\mathbf{y} \\ &= (\mathbf{F}_N \otimes \mathbf{I}_{M_r})(\check{\mathbf{A}}_{DEcp}\mathbf{H}_{toep}\mathbf{x}_{cp} + \check{\mathbf{A}}_{DEcp}\mathbf{n}) \\ &= \frac{1}{N}(\mathbf{F}_N \otimes \mathbf{I}_{M_r})(\check{\mathbf{A}}_{DEcp}\mathbf{H}_{toep}\check{\mathbf{A}}_{cp})(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\tilde{\mathbf{x}} + (\mathbf{F}_N \otimes \mathbf{I}_{M_r})\check{\mathbf{A}}_{DEcp}\mathbf{n} \\ &= \frac{1}{N}((\mathbf{F}_N \otimes \mathbf{I}_{M_r})\mathbf{H}_{cir}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t}))\tilde{\mathbf{x}} + \tilde{\mathbf{n}} \end{aligned} \tag{2.85}$$

where $\mathbf{H}_{cir} = \check{\mathbf{A}}_{DEcp}\mathbf{H}_{toep}\check{\mathbf{A}}_{cp}$ is a block circulant matrix. It has the following form:

$$\mathbf{H}_{cir} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{H}_{L-1} & \mathbf{H}_{L-2} & \cdots & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{H}_{L-1} & \cdots & \mathbf{H}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{L-2} & \cdots & \cdots & \mathbf{H}_0 & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{H}_{L-1} \\ \mathbf{H}_{L-1} & \cdots & \cdots & \cdots & \mathbf{H}_0 & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{L-1} & \cdots & \cdots & \cdots & \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{H}_{L-1} & \mathbf{H}_{L-2} & \cdots & \cdots & \mathbf{H}_0 \end{bmatrix}_{NM_r \times M_t} \tag{2.86}$$

Assuming the eigenvalue decomposition of the block circulant matrix $\mathbf{H}_{cir}$ as given in Eq. 2.40, and performing FFT at the receiver side, the system equation for MIMO CP-OFDM is obtained as follows:

$$\begin{aligned} \tilde{\mathbf{y}} &= \frac{1}{N}(\mathbf{F}_N \otimes \mathbf{I}_{M_r})(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\tilde{\mathbf{H}}_b\frac{1}{N}(\mathbf{F}_N \otimes \mathbf{I}_{M_r})(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\tilde{\mathbf{x}} + (\mathbf{F}_N \otimes \mathbf{I}_{M_r})\mathbf{n} \\ &= \tilde{\mathbf{H}}_b\tilde{\mathbf{x}} + \tilde{\mathbf{n}} \end{aligned} \tag{2.87}$$

Here, $\tilde{\mathbf{H}}_b$ is a block diagonal matrix given as

$$\tilde{\mathbf{H}}_b = \begin{bmatrix} \tilde{\mathbf{H}}(0) & & & & \\ & \ddots & & & \\ & & \tilde{\mathbf{H}}(k) & & \\ & & & \ddots & \\ & & & & \tilde{\mathbf{H}}(N-1) \end{bmatrix} \tag{2.88}$$

where the diagonal block $\tilde{\mathbf{H}}(k)$ contains the flat fading frequency domain coefficients of all existent paths between M_t transmit and M_r receive antennas relative to the subcarrier k, $k = 0, \ldots, N-1$.

$$\tilde{\mathbf{H}}(k) = \begin{bmatrix} \tilde{h}_{11}(k) & \tilde{h}_{12}(k) & \cdots & \tilde{h}_{1M_t}(k) \\ \tilde{h}_{21}(k) & \tilde{h}_{22}(k) & \cdots & \tilde{h}_{2M_t}(k) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{h}_{M_r1}(k) & \tilde{h}_{M_r1}(k) & \cdots & \tilde{h}_{M_rM_t}(k) \end{bmatrix}_{M_r \times M_t}$$

2.2.3 Detection

The transmitted data in a spatial multiplexing MIMO system can be recovered by linear and nonlinear detection. In case of linear detection the estimate of the transmit data vector is obtained by linearly combining the elements of the receive data vector. Among linear schemes, linear minimum mean square error (LMMSE) detection performs the best [80]. Nonlinear detection is represented by successive interference cancellation [72] and maximum likelihood (ML) detection. In terms of BER, ML detection outperforms other linear and nonlinear methods [81], however its complexity increases exponentially with the number of antennas and constellation size [82]. Even though LMMSE is inferior to ML in terms of BER it is still attractive because of its very low complexity. Successive interference cancellation is in between in terms of performance and complexity.

Based on the above, only LMMSE and ML detection will be considered in this work, as they represent the lower/upper bounds in terms of complexity/performance respectively. In the following sections, both methods are outlined for the general linear model given in Eq. 2.61.

2.2.3.1 LMMSE

The objective of LMMSE estimator is to minimize the mean square error between the transmit symbol vector $\mathbf{x}$ and the estimate of the transmit symbol vector $\hat{\mathbf{x}}$. The minimum mean square error is given as

$$mmse = \min E\{(\mathbf{x} - \hat{\mathbf{x}})^2\} \tag{2.89}$$

where $E\{\cdot\}$ denotes expectation. By constraining the estimator to be linear, the estimate of the transmit symbol vector is written as [83]

$$\hat{\mathbf{x}} = \mathbf{W}^H \mathbf{y} \tag{2.90}$$

where $\mathbf{W}$ is the linear estimator, also called the linear combiner.

Therefore, the goal is to find the optimal $\mathbf{W}^H$, that minimizes Eq. 2.89. Expanding Eq. 2.89 and plugging Eq. 2.90 into Eq. 2.89 the cost function is obtained as

$$\begin{aligned} E\{(\mathbf{x}-\hat{\mathbf{x}})^2\} &= E\{(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^H\} \\ &= -E\{\mathbf{x}\mathbf{y}^H\}\mathbf{W} + E\{\mathbf{x}\mathbf{x}^H\} + \mathbf{W}^H E\{\mathbf{y}\mathbf{y}^H\}\mathbf{W} - \mathbf{W}^H E\{\mathbf{y}\mathbf{x}^H\} \\ &= -\mathbf{C}_{\mathbf{xy}} + \mathbf{C}_{\mathbf{xx}} + \mathbf{W}^H \mathbf{C}_{\mathbf{yy}} \mathbf{W} - \mathbf{W}^H \mathbf{C}_{\mathbf{xy}} \end{aligned} \tag{2.91}$$

where $\mathbf{C}_{\mathbf{xx}}$ is the transmit covariance matrix, $\mathbf{C}_{\mathbf{yy}}$ is the receive covariance matrix and $\mathbf{C}_{\mathbf{xy}}$ is the transmit/receive covariance matrix.

The minimum of the mean square error cost function in Eq. 2.91 is found by taking the derivative with respect to $\mathbf{W}$ and setting it to zero. Thereby, $\mathbf{W}^H$ is obtained as

$$\mathbf{W}^H = \mathbf{C}_{\mathbf{xy}} \mathbf{C}_{\mathbf{yy}}^{-1} \tag{2.92}$$

Now, the covariance matrix $\mathbf{C}_{\mathbf{xy}}$ in Eq. 2.92 is expanded as

$$\begin{aligned} \mathbf{C}_{\mathbf{xy}} &= E\{\mathbf{x}(\mathbf{H}\mathbf{x}+\mathbf{n})^H\} \\ &= E\{\mathbf{x}\mathbf{x}^H\}\mathbf{H}^H + \underbrace{E\{\mathbf{x}\mathbf{n}^H\}}_{0} \\ &= \mathbf{C}_{\mathbf{xx}}\mathbf{H}^H \end{aligned} \tag{2.93}$$

The receive covariance matrix $\mathbf{C}_{\mathbf{yy}}$ is expanded as follows

$$\begin{aligned} \mathbf{C}_{\mathbf{yy}} &= E\{(\mathbf{H}\mathbf{x}+\mathbf{n})(\mathbf{H}\mathbf{x}+\mathbf{n})^H\} \\ &= \mathbf{H}E\{\mathbf{x}\mathbf{x}^H\}\mathbf{H}^H + \mathbf{H}\underbrace{E\{\mathbf{x}\mathbf{n}^H\}}_{0} + \underbrace{E\{\mathbf{n}\mathbf{x}^H\}}_{0}\mathbf{H}^H + E\{\mathbf{n}\mathbf{n}^H\} \\ &= \mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^H + \mathbf{C}_{\mathbf{nn}} \end{aligned} \tag{2.94}$$

In the latter equations, the respective expectations are zero due to the fact that the transmit symbol vector $\mathbf{x}$ and AWGN noise vector $\mathbf{n}$ are uncorrelated. The matrix $\mathbf{C}_{\mathbf{nn}}$ is the noise covariance matrix.

Hence, the LMMSE estimate in Eq. 2.90 is given as

$$\hat{\mathbf{x}} = \mathbf{C}_{\mathbf{xx}}\mathbf{H}^H(\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^H + \mathbf{C}_{\mathbf{nn}})^{-1}\mathbf{y} \tag{2.95}$$

It is easy to rewrite the estimate in a more convenient form. This is is achieved by allowing the following identity [84]

$$\mathbf{C}_{\mathbf{xx}}\mathbf{H}^H(\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^H + \mathbf{C}_{\mathbf{nn}})^{-1} = (\mathbf{H}^H\mathbf{C}_{\mathbf{nn}}^{-1}\mathbf{H}^H + \mathbf{C}_{\mathbf{xx}}^{-1})^{-1}\mathbf{H}^H\mathbf{C}_{\mathbf{nn}}^{-1} \tag{2.96}$$

It can be easily shown that the identity is correct by left multiplying with $(\mathbf{H}^H\mathbf{C}_{\mathbf{nn}}^{-1}\mathbf{H}^H + \mathbf{C}_{\mathbf{xx}}^{-1})$ and right multiplying with $(\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^H + \mathbf{C}_{\mathbf{nn}})$ both sides of Eq. 2.96.

Thus, using the identity in Eq. 2.96, the LMMSE estimate is equivalently reformulated as

$$\hat{\mathbf{x}} = (\mathbf{H}^H\mathbf{C}_{\mathbf{nn}}^{-1}\mathbf{H} + \mathbf{C}_{\mathbf{xx}}^{-1})^{-1}\mathbf{H}^H\mathbf{C}_{\mathbf{nn}}^{-1}\mathbf{y} \tag{2.97}$$

This form of the estimate is favorable, because the size of the matrix to be inverted is smaller than that of the matrix to be inverted in Eq. 2.95.

Assuming i.i.d. Gaussian distributed transmit data and additive noise, the data and noise covariance matrices are

$$\mathbf{C}_{\mathbf{xx}} = \sigma_x^2\mathbf{I}$$
$$\mathbf{C}_{\mathbf{nn}} = \sigma_n^2\mathbf{I}$$

where σ_n^2 is the AWGN noise variance and σ_x^2 is the transmit data variance. The LMMSE estimate is in this case given as

$$\hat{\mathbf{x}} = \left(\mathbf{H}^H\mathbf{H} + \frac{\sigma_n^2}{\sigma_x^2}\mathbf{I}\right)^{-1}\mathbf{H}^H\mathbf{y} \tag{2.98}$$

2.2.3.1.1 Soft output generation

In order to improve BER performance, the decoder should take the a priori probabilities of the input bits into account. This information is defined in terms of log-likelihood ratios and has to be passed from the detector to the channel decoder. The LLRs are computed for each bit of the estimated transmit symbol vector $\mathbf{x}$. In the binary domain, the symbol vector $\mathbf{x}$ is regarded as a two-dimensional array, where index j refers to the symbol position within the symbol vector and index b refers to the b-th bit of the j-th symbol.

The LLR for each bit $x_{j,b}$ equals

$$L_{j,b} = \log\frac{P(x_{j,b} = 1|\mathbf{y}, \mathbf{H})}{P(x_{j,b} = 0|\mathbf{y}, \mathbf{H})} \tag{2.99}$$

where $P(\cdot)$ denotes probability of an event. Using Bayes theorem (Eq. A.21) and assuming equal probability of transmit bit values, the LLR is expressed as

$$L_{j,b} = \log\left(\frac{\sum\limits_{\mathbf{x}\in\chi_{j,b}^1} p(\mathbf{y}|\mathbf{x}, \mathbf{H})}{\sum\limits_{\mathbf{x}\in\chi_{j,b}^0} p(\mathbf{y}|\mathbf{x}, \mathbf{H})}\right) \tag{2.100}$$

where $\chi_{j,b}^1$ and $\chi_{j,b}^0$ are sets of all transmission vectors with bit position b of symbol j equal to one or zero respectively and $p(\cdot)$ denotes probability density function (pdf).

The multivariate complex Gaussian probability densities in Eq. 2.100 are given as

$$p(\mathbf{y}|\mathbf{x},\mathbf{H}) = \frac{1}{(\pi\sigma_n^2)^{M_r}} \exp\left(-\frac{\|\mathbf{Hx}-\mathbf{y}\|_2^2}{\sigma_n^2}\right) \tag{2.101}$$

Therefore, the LLR is reformulated as

$$\begin{aligned} L_{j,b} &= \log\left(\frac{\sum\limits_{\mathbf{x}\in\chi_{j,b}^1} \exp\left(-\frac{\|\mathbf{Hx}-\mathbf{y}\|_2^2}{\sigma_n^2}\right)}{\sum\limits_{\mathbf{x}\in\chi_{j,b}^0} \exp\left(-\frac{\|\mathbf{Hx}-\mathbf{y}\|_2^2}{\sigma_n^2}\right)}\right) \\ &= \log \sum_{\mathbf{x}\in\chi_{j,b}^1} \exp\left(-\frac{\|\mathbf{Hx}-\mathbf{y}\|_2^2}{\sigma_n^2}\right) - \log \sum_{\mathbf{x}\in\chi_{j,b}^0} \exp\left(-\frac{\|\mathbf{Hx}-\mathbf{y}\|_2^2}{\sigma_n^2}\right) \end{aligned} \tag{2.102}$$

The number of terms that have to be evaluated in Eq. 2.102 equals $2^{n_b M_t}$, where n_b is the number of bits per modulation symbol.

Using max-log approximation ($\log \sum_s a_s = \max a_s$) [85], Eq. 2.102 is simplified as given below

$$L_{j,b} \approx \frac{1}{\sigma_n^2}\left(\min_{\mathbf{x}\in\chi_{j,b}^0} \|\mathbf{y}-\mathbf{Hx}\|_2^2 - \min_{\mathbf{x}\in\chi_{i,b}^1} \|\mathbf{y}-\mathbf{Hx}\|_2^2\right) \tag{2.103}$$

Now, direct computation of probabilities and logarithm is eliminated, but still $2^{n_b M_t}$ Euclidean distances need to be computed. The complexity of this computation grows exponentially with n_b and M_t and is therefore not well tailored for hardware implementation.

Further simplification was proposed in [86]. Instead of computing LLR on the receive symbol vector $\mathbf{y}$, the output $\hat{\mathbf{x}}$ of the LMMSE is considered. It can be reformulated as

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{W}^H\mathbf{y} \\ &= \mathbf{x} + (\mathbf{W}^H\mathbf{H} - \mathbf{I}_{M_t})\mathbf{x} + \mathbf{W}^H\mathbf{n} \end{aligned} \tag{2.104}$$

where the second term is the residual interference. Assuming the residual interference $(\mathbf{W}^H\mathbf{H}-\mathbf{I}_{M_t})\mathbf{x}$ to be approximately Gaussian, the model in Eq. 2.104 can be considered as a set of M_t independent SISO Gaussian channels.

Therefore, the approximate LMMSE-based LLR is computed for each of the M_t independent streams or elements of the transmit symbol vector $\mathbf{x}$

$$\bar{L}_{j,b} \approx \frac{1}{\sigma_e^2}\left(\min_{x_j\in\chi_b^0} |\hat{x}_j - x_j|^2 - \min_{x_j\in\chi_b^1} |\hat{x}_j - x_j|^2\right) \tag{2.105}$$

where σ_e^2 is the diagonal element of the MMSE matrix $\mathbf{C}_{ee}$, which is given as

$$\mathbf{C}_{ee} = \sigma_n^2\left(\mathbf{H}^H\mathbf{H} + \frac{\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1} \tag{2.106}$$

The sets χ_b^0 and χ_b^1 contain all binary representations of symbol j, for which the bit at position b equals zero or one respectively. Therefore, the number of Euclidean distances to be evaluated has been reduced to $2^{n_b} M_t$. Furthermore, it is to note that in case of QAM modulation and Gray mapping $\bar{L}_{j,b}$ is a piecewise linear function that depends only on the real or the imaginary part of $\hat{x}_j$ [86].

For example, in case of 4-QAM and Gray mapping used in this work, the LLRs are computed separately for in-phase and quadrature components

$$\bar{L}_{j,1} = \frac{4}{\sigma_e^2} \Re\{\hat{x}_j\} \tag{2.107}$$

$$\bar{L}_{j,2} = \frac{4}{\sigma_e^2} \Im\{\hat{x}_j\} \tag{2.108}$$

2.2.3.1.2 Cyclic prefix

In case of MIMO CP-OFDM the frequency domain system equation is given in Eq. 2.87. As $\tilde{\mathbf{H}}_b$ is a block diagonal matrix, the detection can be regarded as solving N independent linear systems. Each system is equivalent to generic MIMO system equation (Eq. 2.61), where vectors and matrices are defined in frequency domain

$$\tilde{\mathbf{y}}(k) = \begin{bmatrix} \tilde{y}_1(k) \\ \tilde{y}_2(k) \\ \vdots \\ \tilde{y}_{M_r}(k) \end{bmatrix} = \tilde{\mathbf{H}}(k) \begin{bmatrix} \tilde{x}_1(k) \\ \tilde{x}_2(k) \\ \vdots \\ \tilde{x}_{M_t}(k) \end{bmatrix} + \tilde{\mathbf{n}}(k) \tag{2.109}$$

Thus, for each sub-carrier k, and with $\mathbf{C_{nn}} = N\sigma_n^2\mathbf{I}$, $\mathbf{C_{xx}} = \sigma_x^2\mathbf{I}$, the LMMSE estimate is given as

$$\hat{\tilde{\mathbf{x}}}(k) = \left(\tilde{\mathbf{H}}^H(k)\tilde{\mathbf{H}}(k) + \frac{N\sigma_n^2}{\sigma_x^2}\mathbf{I} \right)^{-1} \tilde{\mathbf{H}}^H(k)\tilde{\mathbf{y}}(k) \tag{2.110}$$

2.2.3.2 Maximum likelihood

As the name suggests ML estimation seeks to maximize the likelihood of received symbol vector $\mathbf{y}$, given the transmit symbol vector $\mathbf{x}$.

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{A}^{M_t}}{\operatorname{argmax}}\, p_{y|x}(\mathbf{y}|\mathbf{x}) \tag{2.111}$$

The pdf in Eq. 2.111 is exactly the multivariate Gaussian pdf, given in Eq. 2.101. The maximizazion of pdf in Eq. 2.101 can be performed on its logarithm, since logarithm is a strictly monotonic function. The constant term in Eq. 2.101 is irrelevant for the maximization.

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{A}^{M_t}}{\operatorname{argmax}} \left(-\frac{\|\mathbf{Hx} - \mathbf{y}\|_2^2}{\sigma_n^2} \right) \tag{2.112}$$

To maximize the term in Eq. 2.112 it is sufficient to minimize the Euclidean distance between the receive signal vector $\mathbf{y}$ and the transmit symbol vector $\mathbf{x}$. Hence, the

maximization is replaced by minimization as expressed in Eq. 2.113

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{A}^{M_t}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \tag{2.113}$$

The brute force approach to solving this minimization problem is to search through all possible transmit symbol vectors and pick the one with the minimum Euclidean distance. This approach is not practical for MIMO OFDM systems, as its complexity grows exponentially with increasing the constellation size and the number of antennas [82].

In order to solve the ML problem with practical complexity the sphere decoding algorithm has been proposed.

2.2.3.2.1 Sphere decoding

Sphere decoding (SD) relates to the problem of finding a closest vector in a lattice which was initially introduced in [87]. The algorithm was soon adapted to the problem of maximum likelihood detection of received data in a MIMO communication system [88]. The algorithm has rapidly gained close attention from the research community and a number of versions have been developed [89, 90]. Subsequent improvements aimed at reducing the algorithm complexity [91, 92]. Versions of the algorithm particularly suitable for hardware implementation were proposed by the authors of [93].

The idea underlying the SD is to map the minimization problem in Eq. 2.113 to a tree search. First, the QR decomposition of the channel matrix $\mathbf{H}$ is performed. The result of this decomposition is the unitary matrix $\mathbf{Q}$ and the upper triangular matrix $\mathbf{R}$. Next, the system equation (Eq. 2.61) is transformed by multiplication with the complex conjugate of $\mathbf{Q}$

$$\bar{\mathbf{y}} = \mathbf{R}\mathbf{x} + \mathbf{Q}^H\mathbf{n} \tag{2.114}$$

where $\bar{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$. The quality of the received signal is not affected by the latter transformation. The reason is that the altered noise vector $\bar{\mathbf{n}} = \mathbf{Q}^H\mathbf{n}$ has the same statistics as the initial noise vector $\mathbf{n}$ due to the properties of $\mathbf{Q}$ [57]. Therefore, the minimization problem can be reformulated as

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{A}^{M_t}}{\operatorname{argmin}} \|\bar{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 \tag{2.115}$$

The Euclidean distance to be minimized in Eq. 2.115 is denoted as $\Delta(\mathbf{x})$

$$\Delta(\mathbf{x}) = \|\bar{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 \tag{2.116}$$

This Euclidean distance is to be computed for a particular transmit symbol vector $\mathbf{x}$. As $\mathbf{R}$ is upper-triangular, $\bar{\mathbf{y}} - \mathbf{R}\mathbf{x}$ is an upper triangular system of equations. This implies that the Euclidean distance $\Delta(\mathbf{x})$ can be obtained by parts corresponding to the partial symbol vectors, $\mathbf{x}^{(i)}$ of size $M_t - i + 1$, $i = M_t, \ldots, 1$.

The upper triangular system of equations can be mapped to a tree. The number of levels in the tree, excluding the root, is given by M_t. Each node on the level i below the

root represents a possible value of the partial transmit symbol vector $\mathbf{x}^{(i)}$. Each node is labeled with a partial Euclidean distance (PED), denoted as $\Delta_i(\mathbf{x}^{(i)})$. The PED of each child node on level i can be easily computed from the PED of the respective parent node on level $i+1$ [93].

$$\Delta_i(\mathbf{x}^{(i)}) = \Delta_{i+1}(\mathbf{x}^{(i+1)}) + |\delta_i(\mathbf{x}^{(i)})|^2 \tag{2.117}$$

Here, $|\delta_i(\mathbf{x}^{(i)})|$ is the positive Euclidean distance increment mapped to the branch connecting the parent node on level $i+1$ to the child node on level i. The distance increments are easily obtained from Eq. 2.116 as follows

$$|\delta_i(\mathbf{x}^{(i)})|^2 = |\bar{y}_i - \sum_{j=i+1}^{M_t} r_{ij}s_j - r_{ii}x_i|^2 \tag{2.118}$$

The mapping of a MIMO system with $M_t = M_r = 3$ and real valued modulation alphabet $\{+1, -1\}$ to a tree is shown in Fig. 2.20.

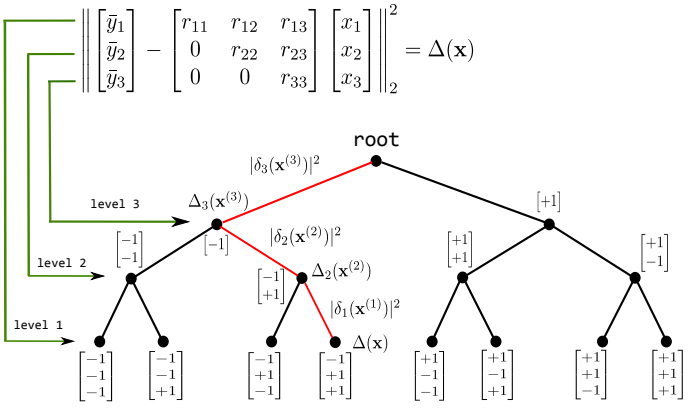


FIGURE 2.20: Tree search

The leaf nodes of the tree represent all possible transmit symbol vectors. The search through the tree is initiated to obtain the leaf node with the smallest Euclidean distance to the receive symbol vector.

The search through the tree can be performed either by breadth-first or depth-first tree traversal [94]. The breadth-first tree traversal is a non-recursive scheme which starts at the root and traverses the tree in downward direction only, whereas depth-first traversal is a recursive scheme.

Irrespective of the tree traversal strategy, visiting all available nodes in the tree corresponds to exhaustive ML search. Therefore, in order to decrease the number of visited nodes, the parts of the tree have to be pruned. The search is restricted to nodes, whose corresponding PEDs lie inside a hypersphere of radius ρ

$$\|\bar{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 < \rho^2 \tag{2.119}$$

Equation 2.119 is referred to as the sphere constraint (SC) [93]. If PED of a node violates the SC:

$$\Delta_i(x^{(i)}) > \rho^2 \tag{2.120}$$

that node and all its children will be pruned from the tree.

The sphere decoding algorithm is summarized in Alg. 1

- The algorithm starts at the root and initializes the PED of the root and the SC (line 2).
- The depth first tree traversal is initiated. Starting from the level below the root, a level specific list $\mathbf{S}_i$ containing the transmit symbols from the alphabet $\mathbb{A}$ is established (line 4, function LISTINIT). For each symbol in the list its Euclidean distance increment $\delta(\mathbf{x}^{(i)})$ is computed by Eq. 2.118. The distance increments are then sorted in ascending order [95]. The value of the receive vector $\bar{\mathbf{y}}^{(i)}$ is stored as well. Its value is specific for the current level, therefore it is labeled with index i.

  As the Euclidean distance increments are sorted in ascending order, the one which adds the least amount to the overall Euclidean distance is used to obtain the PED of level i by Eq. 2.117. The used symbol is removed from the list (line 6).
- If the PED of the node on current level does not violate the SC (line 11), the symbol value is fixed. It is necessary to subtract the portion of x_i from the current receive symbol vector $\bar{\mathbf{y}}^{(i)}$. This update is performed as follows:

  $$\bar{\mathbf{y}}^{(i-1)} = \bar{\mathbf{y}}^{(i)} - \mathbf{r}_i x_i \tag{2.121}$$

  where $\mathbf{r}_i$ is the i-th column of $\mathbf{R}$.
- If a leaf node is not yet reached, the algorithm moves one level down (line 14). The list initialization is called with the updated version of the receive symbol vector (line 15).
- If a leaf node is reached (line 16), the current best solution vector is found. The SC is tightened by setting it equal to the overall Euclidean distance corresponding to the reached leaf node (line 18).
- SC violation by PED of a node at level i implies that all children of that nodes will also violate the SC as the Euclidean distance increments of branches leading to them would only increase the PED. Hence, it is moved up one level (line 22). The best ML solution is sought with the next symbol from $\mathbf{S}_i$ (line 6).
- If the list on the current level is empty, it is moved up one level (line 8).
- The search terminates when the root is reached (line 5).

Algorithm 1 Sphere decoding algorithm

Input: $\bar{\mathbf{y}}$, $\mathbf{R}, \mathbb{A}$. Output: $\hat{\mathbf{x}}_{ML}$.
$\Delta_{M_t+1}(\mathbf{x}^{(M_t+1)}) = 0$, $\rho^2 = \infty$
$i = M_t$
$\mathbf{S}_i = \text{LISTINIT}(i, \bar{\mathbf{y}}^{(i)}, \mathbf{R}, \mathbb{A})$
while $i < M_t + 1$ **do**
    get first x_i and its $|\delta_i(\mathbf{x}^{(i)})|^2$ from $\mathbf{S}_i$. Remove these entries from $\mathbf{S}_i$.
    if isempty($\mathbf{S}_i$) **then**
        $i = i + 1$
    else
        $\Delta_i(\mathbf{x}^{(i)}) = \Delta_{i+1}(\mathbf{x}^{(i+1)}) + |\delta_i(\mathbf{x}^{(i)})|^2$ ▷ update PED
        if $\Delta_i(\mathbf{x}^{(i)}) < \rho^2$ **then**
            if $i > 1$ **then** ▷ not at a leaf node yet
                update $\bar{\mathbf{y}}^{(i)}$ (Eq. 2.121)
                $i = i - 1$
                $\mathbf{S}_i = \text{LISTINIT}(i, \bar{\mathbf{y}}^{(i)}, \mathbf{R}, \mathbb{A})$
            else ▷ new best ML solution found
                $\hat{\mathbf{x}} = \mathbf{x}^{(1)}$
                $\rho^2 = \Delta_i(\mathbf{x}^{(i)})$
                $i = i + 1$;
            end if
        else
            $i = i + 1$
        end if
    end if
end while
function $\mathbf{S}_i$ =LISTINIT$(i, \bar{\mathbf{y}}^{(i)}, \mathbf{R}, \mathbb{A})$
    for all $x_i \in \mathbb{A}$ **do**
        compute Euclidean distance increments (Eq. 2.118)
        $\mathbf{s}_i = \begin{bmatrix} x_i & \delta_i(\mathbf{x}^{(i)}) \end{bmatrix}$
    end for
    Sort the rows $\mathbf{s}_i$ of $\mathbf{S}_i$ in ascending order
end function

2.2.3.2.1.1 Soft output generation

Consider the LLR computation in Eq. 2.103. Here, for every bit, one of the minima corresponds to the ML metric, obtained by the SD algorithm.

$$\lambda^{ML} = \frac{1}{\sigma_n^2} \|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|_2^2 \tag{2.122}$$

where $\hat{\mathbf{x}} = \mathbf{x}^{ML}$ is the ML solution of Eq. 2.113, obtained by SD. The other minimum corresponds to the ML solution vector with the flipped b-th bit of the j-th symbol.

$$\lambda_{j,b}^{\overline{ML}} = \frac{1}{\sigma_n^2} \min_{\mathbf{x} \in \chi_{j,b}^{(\overline{x_{j,b}^{ML}})}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \tag{2.123}$$

Here, $\overline{x_{j,b}^{ML}}$ is the bit-wise complement of the ML solution, or the ML counter-hypothesis.

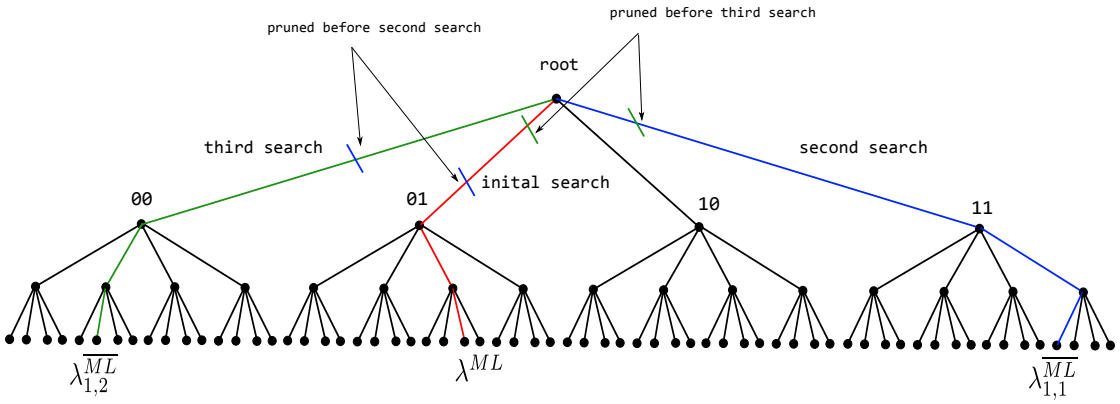


FIGURE 2.21: Repeated tree search

Using the above definitions, the LLR is reformulated as

$$L_{j,b} = \begin{cases} \lambda^{ML} - \lambda_{j,b}^{\overline{ML}} & \text{if } x_{j,b}^{ML} = 0 \\ \lambda_{j,b}^{\overline{ML}} - \lambda^{ML} & \text{if } x_{j,b}^{ML} = 1 \end{cases} \tag{2.124}$$

For each bit, the ML metric λ^{ML} is obtained by the SD algorithm. Thereafter, for each bit it is necessary to obtain the ML counter-hypothesis $\lambda_{j,b}^{\overline{ML}}$.

This solution to this problem was initially provided by the repeated tree search (RTS) sphere decoding [96]. Consider the example tree given in Fig. 2.21. It corresponds to a MIMO system with $M_t = 3$ transmit and $M_r = 3$ receive antennas and 4-QAM Gray mapped modulation alphabet. Therefore, the transmitted symbol is mapped to $\mathbb{B}^2 \in \{0,1\}^2$.

The binary version of the ML solution vector obtained by first run of the SD is highlighted in red. During this first run, the ML metric λ^{ML} is computed. Next, the ML counter-hypothesis metric of the first bit of the first symbol $\lambda_{1,1}^{\overline{ML}}$ has to be computed. The first bit of the first symbol of the ML solution (bit 0) is complemented and fixed. This fixing corresponds to pre-pruning the root left-hand side of the tree. Thereafter the SD is rerun on that reduced tree and $\lambda_{1,1}^{\overline{ML}}$ is obtained. The third run of the SD computes the ML counter-hypothesis $\lambda_{1,2}^{\overline{ML}}$ for the second bit of the first symbol. The search starts from the initial tree. The second bit of the ML solution (bit 1) is complemented and fixed. This fixing corresponds to pruning the second and fourth branches from the root. The SD then searches through this reduced tree and obtains $\lambda_{1,2}^{\overline{ML}}$. The tree pre-pruning and SD search is repeated for all remaining $L_{j,b}$. Each new run, the SD operates on a newly reduced tree. However, the complexity increase is given by $n_b M_t$ additional runs.

This increase in complexity has been addressed in the single tree search (STS) SD [97]. The complexity reduction is achieved by searching for the ML solution and all counter-hypotheses concurrently. The main idea is in formulating update rules and the pruning criterion on a list containing the ML λ^{ML} and counter ML $\lambda_{j,b}^{\overline{ML}}$ metrics.

2.2.3.2.2 Likelihood ascent search

Likelihood ascent search is the quasi-ML detection algorithm developed for massive MIMO systems [98–100]. Therefore, it is a natural candidate for MIMO UW-OFDM.

The algorithm performs a sequence of local searches based on an initial solution vector in order to arrive as close as possible to the ML solution. The local search operates on the neighborhood of the initial solution. The neighborhood is defined as a set of vectors that differ from initial vector in up to η positions. For example, the 1-neighborhood of a vector $\mathbf{x} = [+1 - 1 - 1 + 1]$ contains the following vectors

$$S_1(\mathbf{x}) = \{[-\mathbf{1} - 1 - 1 + 1], [+1 + \mathbf{1} - 1 + 1], [+1 - 1 + \mathbf{1} + 1], [+1 - 1 - 1 - \mathbf{1}]\} \tag{2.125}$$

In general, the size of η-neighborhood is given as

$$|S_\eta(\mathbf{x})| = \binom{M_t}{\eta} \tag{2.126}$$

In case $\eta = M_t$, the neighborhood contains all possible solution vectors, and the search within this neighborhood corresponds to the global search. It has been shown that the local search provides good results for $\eta \leq 3$ for massive MIMO systems with hundreds of antennas [98].

The stateflow of LAS algorithm with search within up to $S_3(\mathbf{x})$ is depicted in Fig. 2.22. The LAS algorithm starts with some initial solution vector $\mathbf{x}^{(0)}$, for example the LMMSE estimate.

Next, the search within $S_1(\mathbf{x}^{(0)})$ is initiated. If a vector with shorter Euclidean distance is found, it is declared the current solution. The search is continued within the 1-neighborhood of the current solution. This sub-procedure is denoted 1-LAS. If no vector with shorter Euclidean distance can be found, current solution vector represents a local minimum.

An attempt to escape from this local minimum is performed by initiating a search within 2-neighborhood of the current solution vector. This sub-procedure is denoted 2-LAS. If a better vector is found, it is declared the current solution and the search restarts with 1-LAS.

If a better vector is not found in 2-neighborhood, a search within $S_3(\mathbf{x})$ of the current solution vector. This sub-procedure is denoted 3-LAS. If this search is successful, the search goes on with 1-LAS. Otherwise, the current solution vector is declared the final solution.

The number of performed search stages can be varied and represents the tradeoff between performance and complexity. Running solely 1-LAS features the least complexity, but the risk to get stuck in the local minimum is significant. Running all three search stages is expected to provide the closest to ML solution at the cost of much longer run-time due to large size of the 3-neighborhood.

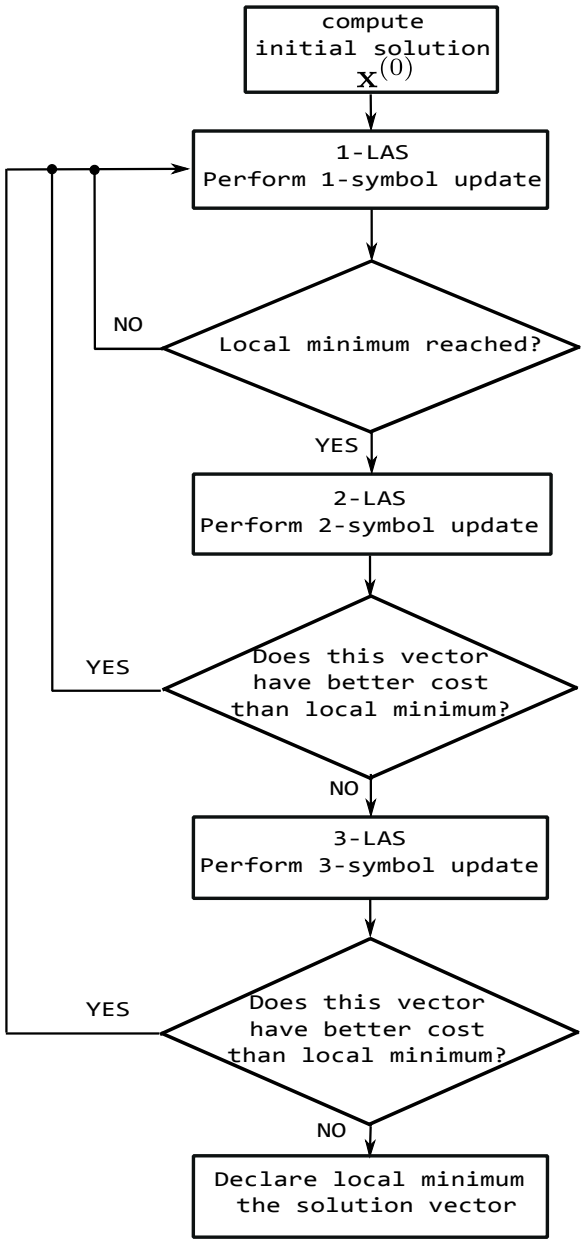


FIGURE 2.22: Stateflow of 3-LAS algorithm

2.3 Fault tolerance

Fault tolerance is the property of the system to operate properly in the event of a failure (or multiple faults) within its components [5]. Traditionally, fault tolerance was of concern in safety critical applications such as aviation or spacecraft [5]. The reliability $R(t)$ is a traditional measure of fault tolerance of a software/ hardware system. It is defined as the probability that the system will be functional up to the time t, given that it was operational at time 0:

$$R(t) = P(T > t) \tag{2.127}$$

where T is the lifetime of the system. This classical fault tolerance measure is suitable for systems that can fail permanently. This is the case for the general purpose computing

systems. For example, a fault in the control path of a microprocessor will most likely put it out of order.

Opposed to general purpose computing, digital signal processing (DSP) systems generally do not fail completely (if only data processing is concerned), but rather their performance is gracefully degraded. DSP applications are characterized by their performance metrics such as BER or SNR. Therefore, faults in these systems will degrade the quality of the output in terms of the performance metrics [16]. Recent trends in the fault tolerant design of DSP systems trade off slight degradation in quality for significant energy savings. Opposed to general purpose computing, where all faults could be critical, DSP systems allow certain amount of faults. Consequently the new fault tolerance measures designed specifically for DSP applications differ from the classical fault tolerance measures. For example, new DSP fault tolerance measures address correcting errors in the statistical sense [19].

Fault tolerant design addresses the means to mitigate the effect of faults on the system's behavior/performance. For any system, this is done by exploiting some sort of redundancy. Usually, hardware, information or time redundancy, or a combination of these is employed [5]. In that sense, fault tolerance has parallels to communications, where information redundancy or time redundancy is used to mitigate the influence of the communication channel [19]. For example, the memory can be regarded as a channel too. Hence, application of error detecting/correcting codes is similar to communications, whereas different error profiles necessitate the use of different codes.

2.3.1 Faults and errors

In terms of fault tolerance, the notions of "fault" and "error" are distinguished. By definition, the *fault* is a model of a defect or other deviation in the hardware circuit [5]. A fault does not necessarily result in an error. Consider an AND logic gate with stuck-at-0 fault at its output. This fault permanently sets the output of the gate to logical-0. This fault will result in an error only if both inputs of the AND gate would be set to logical-1. Therefore, an *error* is defined as the manifestation of a fault.

FIGURE 2.23: Stuck-at-0 fault

In terms of duration, the faults are classified as *permanent*, *transient* and *intermittent* [5]. A permanent fault sets a hardware component permanently out of use. These faults are addressed by testing [101]. A transient fault causes a temporary malfunction, but is gone after some time and the normal operation is restored. An intermittent fault causes the component to oscillate between normal and faulty states.

The transient faults are in the scope of this work. One source of transient faults is the voltage overscaling [102, 103]. VOS refers to setting the supply voltage of the hardware

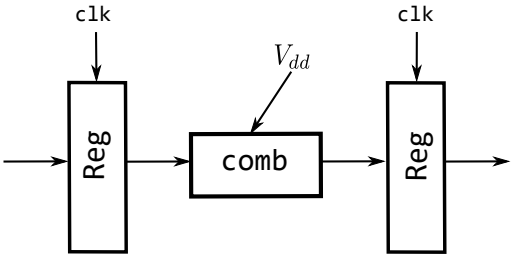


FIGURE 2.24: Voltage overscaling

circuit below the value, for which the correct operation is guaranteed. Consider a circuit depicted in Fig. 2.24. The input to the combinational part is stored in the left register. The combinational part computes a function of the input and stores the result in the right register. The registers are operated with the clock frequency f_{clk}. The critical path delay T_{cp} of the combinational part is defined as the wost case delay over all possible input transitions [102]. The correct operation is ensured when the clock period $T_{clk} = \frac{1}{f_{clk}}$ is greater or equal than the critical path delay

$$T_{clk} \geq T_{cp} \tag{2.128}$$

The gates forming the combinational part are driven by the supply voltage V_{dd}. The supply voltage has to be set such that the condition in Eq. 2.128 is satisfied. The relation between supply voltage V_{dd} and circuit delay τ_c is given by [104]

$$\tau_c = \frac{C_L V_{dd}}{\beta(V_{dd} - V_t)^\alpha} \tag{2.129}$$

where C_L is the load capacitance, α is the velocity saturation index, β is the gate transconductance and V_t is the device threshold voltage. The voltage for which $T_{clk} = T_{cp}$ is referred to $V_{dd-crit}$. Obviously, reducing the supply voltage below $V_{dd-crit}$ leads to an error at the output in case the critical path through the circuit is excited. Hence, $V_{dd-crit}$ is a lower bound on the supply voltage.

VOS introduces input-dependent errors whenever a critical delay path through the circuit is excited. Consider that the combinational part in Fig. 2.24 is a five-bit carry ripple adder (CRA) [105] depicted in Fig. 2.25. It is made of five full adders (FA)s. The inputs to the CRA are $a = 00111$ and $b = 01011$. Assume that the delay of the full adder is $T_{FA} = 3$ ns. The critical path delay is $T_{cp} = 5T_{FA}$ as the carry bit in the least significant bit (LSB) propagates all the way to the most significant bit (MSB). Assuming $T_{clk} = 15$

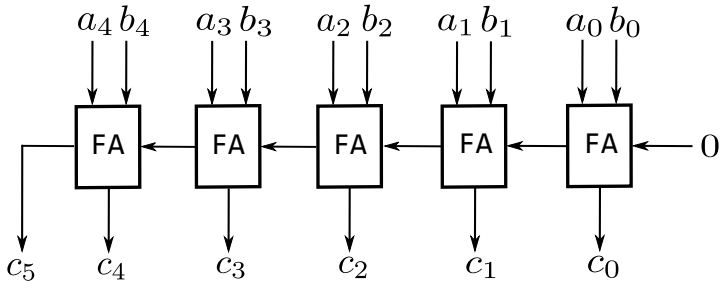


FIGURE 2.25: Carry ripple adder

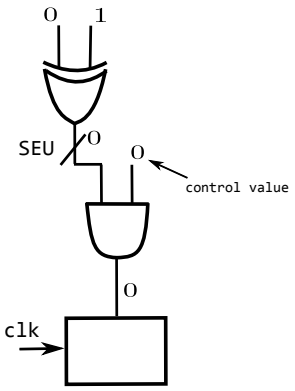


FIGURE 2.26: Logical masking of SEU

ns, the operation condition (Eq. 2.128) is satisfied, and the correct result $c = 100010$ is written to its output register.

Suppose now the supply voltage is reduced such that $T_{FA} = 5$ ns. This implies that the accurate result will be available up to its MSB in $T_{cp} = 25$ ns. However, the clock period is still $T_{clk} = 15$ ns. Therefore, by the end of the clock period, only three bits of the output will be written to the output register and the stored output will be $c = 00010$ instead of $c = 10010$. The erroneous output of the adder will propagate further (Fig. 2.24 can be a sub-block of a larger system). If the inputs do not excite the critical path, the output of the adder will be stored correctly. For example, this is the case with $a = 000110$ and $b = 000001$.

Another major source of transient faults are the hits by highly energetic particle (protons, neutrons, alpha particles), also referred to as the single event upset (SEU) [106, 107]. As it passes through an electronic device, the particle creates additional charge along its path. If this charge is collected at a sensitive node of a circuit, the logical value at this node may get flipped, potentially causing a *soft error* [108]. Soft error in a memory element (flip-flop) persists until the memory element is rewritten [109]. SEU in combinational logic will result in a soft error only if the wrong value would propagate to the next memory element [110]. This is the case if three possible masking effects do not occur. Logical masking prevents the propagation of the SEU, if any of the subsequent gates on the path is driven by the control value, as illustrated in Fig. 2.26. Electrical masking occurs when the wrong logical value is attenuated by the electrical properties of gates on the path, such that the resulting pulse is of insufficient magnitude to be reliably stored in a memory element. Timing masking occurs, when the wrong logical value reaches the memory element outside its clocking interval.

An expression for estimating the soft error rate is given by [111]

$$P_s \propto AF \exp\left(-\frac{Q_{crit}}{Q_s}\right) \tag{2.130}$$

where A is the area, F is the particle flux, Q_{crit} is the critical charge required to flip the logic value at a node, and Q_s is the collection efficiency. As the technology scales, Q_{crit} reduces proportionally as well as Q_s. Therefore, the probability of an effective particle hit at a node remains approximately the same. However, the number of nodes per chip

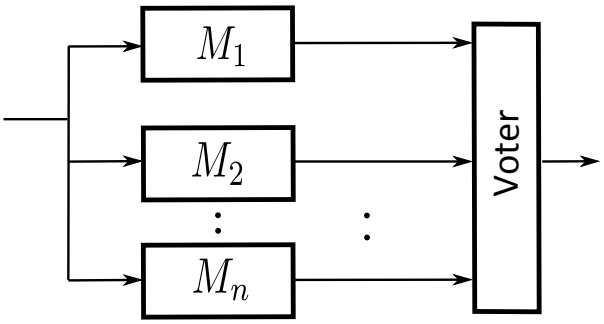


FIGURE 2.27: N-modular redundancy

increases while the clock period and pipeline depth reduces [110]. Thus, the chip-level soft error rate will increase.

Additionally, the hardware faults can be classified as *benign* and *malicious* [5]. A benign fault simply puts the component out of order. A malicious fault alters the output of the component or the result of the computation in such a way that it looks reasonable. However, such altered result can be used in gaining information about secret keys stored. Therefore, malicious attacks on cryptographic circuits are based on injecting such faults.

2.3.2 Redundancy

The redundancy used for fault tolerance can be classified as hardware, information and time redundancy. In hardware redundancy, the classical approach is the N-modular redundancy (NMR) or M-out-N redundancy[5]. Here, the component is duplicated N times as depicted in Fig. 2.27. The voter performs majority voting on N outputs. The system is functional as long as M components are functional and fails (majority voting chooses wrong value), when more than M components fail. Given the reliability of the component $R(t)$, the reliability of an NMR system (under assumption of uncorrelated failures in the components and reliable infrastructure) is the probability that at least M component are functional at time t

$$R_{NMR}(t) = \sum_{i=M}^{N} \binom{N}{i} R^i(t)(1 - R(t))^{N-1} \tag{2.131}$$

The most common example of NMR is the triple modular redundancy (TMR). Taking the reliability of the voter $R_{voter}(t)$ into account, the reliability of a TMR system is given by [5]

$$\begin{aligned} R_{TMR}(t) &= R_{voter}(t) \sum_{i=2}^{3} \binom{3}{i} R^i(t)(1 - R(t))^{3-i} \\ &= R_{voter}(t)(3R^2(t) - 2R^3(t)) \end{aligned} \tag{2.132}$$

NMR is able to mask permanent and transient faults. Extensions of NMR include dynamic and hybrid redundancy [5]. In dynamic redundancy, one component is active at

a time, and $N-1$ component form a reserve. In case the fault of the active component is detected, it is replaced by one from the reserve. Hybrid redundancy employs N modules organized in NMR and K spare modules. The system constantly monitors the components forming the NMR in order to avoid the situation when more than M components are out of order. In case one out of N components is out of order, it is replaced by the component from the K spare ones.

Information redundancy uses error detecting /correcting codes. Codes are commonly applied to protect memories against transient faults. Since soft errors due to particle strikes are relatively rare, single error detecting/correcting block codes are employed (parity, Hamming code) [5]. The usage of more powerful codes such as Berger or cyclic codes is less widespread, since they introduce a large amount of overhead in terms of area and power consumption. The NMR system considered previously can be interpreted as a repetition code, where outputs of the N components represent the codeword. The voter then performs majority decoding.

In order to detect faults in arithmetic circuits, arithmetic codes are used. These codes are preserved under arithmetic operations [5].

Algorithm based fault tolerance (ABFT) is another information redundancy based approach. The redundancy is implemented at the application level by introducing redundant computations. As the exact computations depend on the particular algorithm, the implementation will differ for from one class of applications to the other. ABFT has been implemented for matrix computations (matrix multiplication, matrix inversion, etc) [112]. The redundancy is implemented using a checksum code [113]. Given a matrix $\mathbf{A}$, the column checksum matrix $\mathbf{A}_c$ is defined as

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{A} \\ \mathbf{e}^T\mathbf{A} \end{bmatrix} \tag{2.133}$$

where $\mathbf{e}^T = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$ Similarly, the row checksum matrix is defined as

$$\mathbf{A}_r = \begin{bmatrix} \mathbf{A} & \mathbf{A}\mathbf{e} \end{bmatrix} \tag{2.134}$$

Finally, the full checksum matrix is given as

$$\mathbf{A}_f = \begin{bmatrix} \mathbf{A} & \mathbf{A}\mathbf{e} \\ \mathbf{e}^T\mathbf{A} & \mathbf{e}^T\mathbf{A}\mathbf{e} \end{bmatrix} \tag{2.135}$$

Full checksum matrix allows error detection and correction. Consider an example matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$$

The respective full checksum matrix is given as

$$\mathbf{A}_f = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \\ 2 & 5 & 7 \end{bmatrix}$$

Now assume that an error has happened at the element a_{12} of matrix $\mathbf{A}$:

$$\mathbf{A}_f = \begin{bmatrix} 1 & \mathbf{4} & 3 \\ 1 & 3 & 4 \\ 2 & 5 & 7 \end{bmatrix}$$

Recomputing the row and column checksum matrices indicates that the parities do not match for row 1 and column 2. Hence, the position of the erroneous element is identified. The error is corrected by finding the difference between the recomputed row checksum and the current checksum and adding the difference to the affected element [112]. The same can be performed for column checksums. That is, for the considered example, the difference d between row checksums is given as

$$d = a_{13} - (a_{11} + a_{12}) = -2$$

and the corrected element is obtained as

$$\hat{a}_{12} = a_{12} + d = 2$$

In matrix multiplication, if the terms are protected by the column and row checksum matrices respectively, the resulting matrix is protected by the column and row checksums of the matrix product [112]

$$\mathbf{A}_c\mathbf{B}_r = \begin{bmatrix} \mathbf{A} \\ \mathbf{e}^T\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{Be} \end{bmatrix} = \begin{bmatrix} \mathbf{AB} & \mathbf{ABe} \\ \mathbf{e}^T\mathbf{AB} & \mathbf{e}^T\mathbf{ABe} \end{bmatrix} \tag{2.136}$$

Therefore, the error protection is maintained for matrix multiplication. ABFT has been extended to weighted checksum codes [114] and generalized for real-number codes [115]. The main advantage of ABFT is small hardware overhead. The drawbacks are the time penalty, lack of general applicability and the need to recompute in case of high error rates [116].

Stochastic logic [117] presents a different computing paradigm compared to the conventional logic. Stochastic circuits operate on probabilities instead of logical or arithmetic values used in conventional logic. This allows using simple logic gates to perform arithmetic operations. For instance, the multiplication is performed by just a single AND gate [118]. The input to the gate are two independent random bit streams X_1 and X_2 as shown in Fig. 2.28. The number of ones in the input stream X identifies the input probability value x. The output is the random bit stream Y, with output probability y given as

$$y = P(Y = 1) = P(X_1 = 1 \text{ and } X_2 = 1) = P(X_1 = 1)P(X_2 = 1) = x_1x_2 \tag{2.137}$$

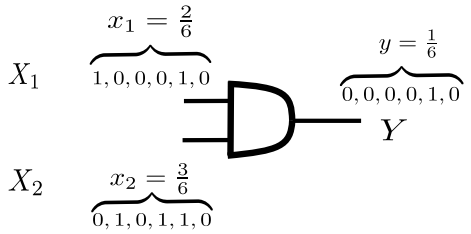


FIGURE 2.28: Multiplication by an AND gate in a stochastic circuit

Therefore, an AND gate implements multiplication. Scaled addition is implemented with a multiplexer [118]. The precision depends on the number of bits in the input/output stream. The stochastic logic exhibits inherent redundancy. A single error at the gate output due to transient fault will not affect the result, given the number of bits in the stream is sufficiently large. This requirement is at the same time the main drawback of the stochastic logic. Large bit streams require random number generators of large size and introduce a time penalty. Another issue is that the correct operation requires random streams to be uncorrelated. Despite these drawbacks, stochastic logic has been used to implement several DSP systems, such as low density parity check (LDPC) decoder [119] and image processing [120].

2.3.3 Stochastic computation

Conventional hardware redundancy systems try to correct or mask all occurring errors. Stochastic computation [19] (not to be confused with stochastic circuits from the last section), on the contrary, treats hardware errors as additional source of noise and tries to provide good enough results from the application perspective. From the application perspective, exact correction of all hardware errors may not be necessary. Such situation arises in DSP applications that process inherently noisy data. Stochastic computation is the extension of conventional hardware redundancy that tries to correct the errors in statistical sense [121]. It does not strive to ensure completely error free operation, but seeks to optimize the application specific metrics such as BER or SNR.

Most of the proposed stochastic computation systems target VOS-induced errors [19]. These systems are able to mitigate the impact of VOS, trading off slight performance degradation for significant energy savings due to reduction of the power supply voltage. Stochastic computation design has been applied to a variety of DSP applications, such as digital filtering [122], FFT [123] and Viterbi decoder [124].

The key representative of stochastic computation approach is the algorithmic noise tolerance (ANT) [21]. The ANT architecture is depicted in Fig. 2.29. The main block computes its output y_a, which can be affected by the VOS-induced error η due to critical path delay violation. Parallel to the main block there is a low complexity replica which computes the estimate of the main block output y_e. This low complexity replica is termed the estimator. Due to its lower complexity, for example by use of reduced precision [23], the reduction in V_{dd} does not incur any error. However, reducing precision truncates the LSB and the output of the estimator contains a precision error e compared to the error free output y_0 of the main block. Further, the difference in statistics of η

and e is exploited. The final output is picked according to the following decision rule

$$\hat{y} = \begin{cases} y_a, & \text{if } |y_a - y_e| < T_h \\ y_e, & \text{otherwise} \end{cases} \tag{2.138}$$

The output of the main block is used if the difference between the output of the main block and the estimator is below a certain threshold T_h. Otherwise, the error is detected and the low precision output of the estimator is used. The value of the threshold T_h is chosen as the maximum difference between the correct output of the main block and the estimator

$$T_h = \underset{\forall input}{\text{argmax}} |y_0 - y_e| \tag{2.139}$$

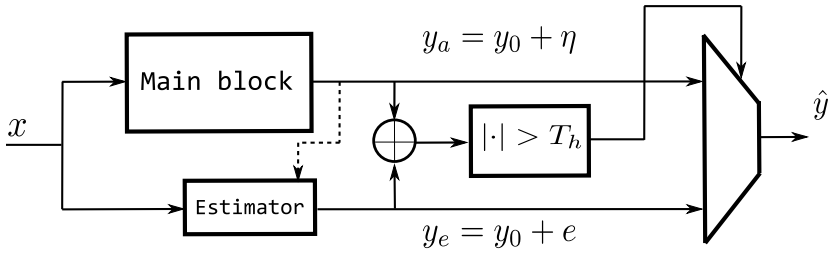


FIGURE 2.29: Algorithmic noise tolerance

Algorithmic soft error tolerance (ASET) is the extension of ANT which addresses soft errors in addition to VOS-induced errors [125]. In ANT, the estimator block is assumed error free. Since a particle can also hit the estimator circuit, this assumption is relaxed in ASET. Now, the assumption is that during a clock cycle at most one internal node in the main block or the estimator can have a logic value flipped. The ASET system is depicted in Fig. 2.30. In addition to the main block, the estimator is now built of two identical blocks, $E1$ and $E2$. The decision rule is outlined in Eq. 2.140. The decision block compares the main block output y_a and the output of the first estimator y_{e1}. If the difference $d^E(y_a, y_{e1})$ (Euclidean distance) between them is less than the threshold T_h, output of the main block is used as the final output. Otherwise, an additional comparator checks whether the outputs of the estimators y_{e1}, y_{e2} are equal. If it is the

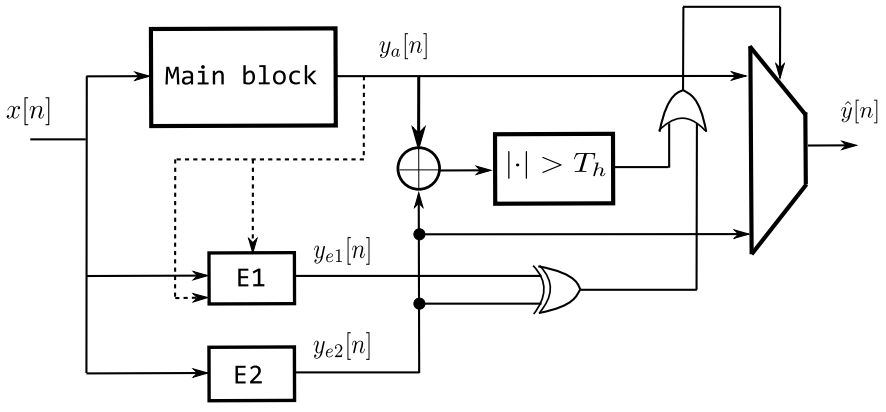


FIGURE 2.30: Algorithmic soft error tolerance

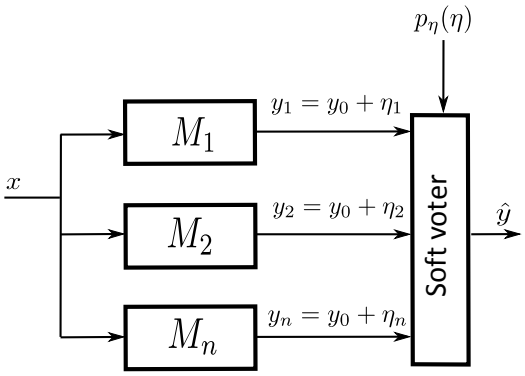


FIGURE 2.31: Soft NMR

case, the single SEU assumption implies that the main block is erroneous and the output of the second estimator is used as the final output. In case the outputs of the estimators are different, the same assumption implies that the error has occurred in either one of them. Then, the output of the main block is passed to the overall output.

$$\hat{y}[n] = \begin{cases} y_a, & \text{if } d^E(y_a, y_{e1}) \leq T_h \\ y_a, & \text{if } d^E(y_a, y_{e1}) > T_h \\ & \text{and } d^H(y_{e1}, y_{e2}) = 1 \\ y_{e2}, & \text{if } d^E(y_a y_{e1}) > T_h \\ & \text{and } d^H(y_{e1}, y_{e2}) = 0 \end{cases} \tag{2.140}$$

Another similar approach is termed soft NMR by the authors of [22]. The soft NMR system is depicted in Fig. 2.31. In contrast to traditional NMR, the voter in soft NMR is a Bayesian detector that treats the N available outputs as noisy observations, contaminated by the additive error η_i, whose statistics $p_\eta(\eta)$ are assumed to be known or can be obtained experimentally. It then combines in order to optimize some predefined cost function. The advantage over the conventional NMR is that the soft NMR is able to produce good estimate even if all N outputs are erroneous. This is achieved by using the error statistics. The soft NMR treats the computations in the N modules as a noisy communications channel.

The soft voter is considered a realization of a Bayesian detector. The latter tries to minimize the associated cost in making a decision. Define $C(\hat{y}, y_0)$ the cost in choosing $\hat{y}$ as the correct value, given that y_0 is indeed the correct value. Denoting the $\delta(y_1, \ldots, y_N) = \hat{y}$ the decision rule based on the N observations, the average cost is given as

$$E\{C(\delta(y_1, \ldots, y_N), Y_0)\} \tag{2.141}$$

where Y_0 is the random variable representing the correct output. The goal is to find the decision rule that minimizes the average cost. The detector must be constrained to a predefined hypothesis set $\mathcal{H}$ such that $\hat{y} \in \mathcal{H}$.

The average cost is application dependent. For example, in DSP applications, the appropriate metric is the MMSE. The MMSE cost function is $C = (\hat{y} - y_0)^2$. In case of

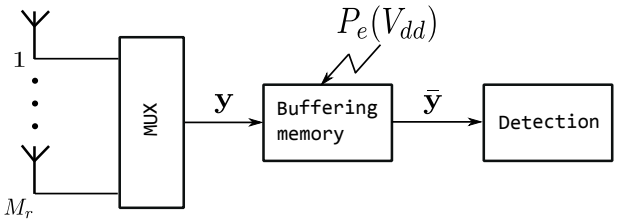


FIGURE 2.32: MIMO receiver with faulty buffer memory

Gaussian error statistics, it is known that the cost function is minimized when $\hat{y}$ is the mean of the observations. Thus, the soft voter chooses the output out of the set of allowed outputs $\mathcal{H}$ that is closest to the mean of the observations.

$$\hat{y} = \underset{\forall y_0 \in \mathcal{H}}{\operatorname{argmin}} \left| y_0 - \frac{\sum_{i=1}^{N} y_i}{N} \right| \tag{2.142}$$

2.3.4 Unified channel and memory error model

Stochastic computation approaches represent a form of hardware redundancy and are therefore applicable to computational sub-components of a larger DSP implementation. Concerning MIMO receivers, the authors of [126] argue that due to the memory dominance of the respective implementations it is worth to concentrate on memory errors, and develope a unified statistical model that combines memory errors due to VOS and channel noise.

Based on this model, the Viterbi algorithm [127] and the breadth-first search detector [6] have been modified to take the VOS-induced errors within the receiver's buffering memory into account. These memory error resilient algorithms trade off slight BER performance degradation for significant reduction in power consumption due to VOS. The statistical memory and channel noise model is outlined next.

Consider a MIMO receiver depicted in Fig. 2.32. As the buffering memory stores the real and imaginary parts separately, the system model in Eq. 2.61 is considered real valued in this section.

It should be noted that the buffering memory contains m receive symbol vectors $\mathbf{y}$ corresponding to the period of time where the channel matrix $\mathbf{H}$ is constant (channel coherence time). The elements of the receive vectors may be either contaminated by the memory errors (highlighted in red) or be memory error free as shown in Fig. 2.33.

The memory errors in model developed in [126] are due to VOS. The effect of voltage scaling on embedded memories has been researched extensively [128–130]. The induced error can be modeled as spatially uniformly distributed random variable where the probability of failure for each cell in the memory is the same $P_e(V_{dd})$ for a fixed supply voltage, V_{dd}, and linearly increases in the logarithmic domain with reduction of the supply voltage [129, 130].

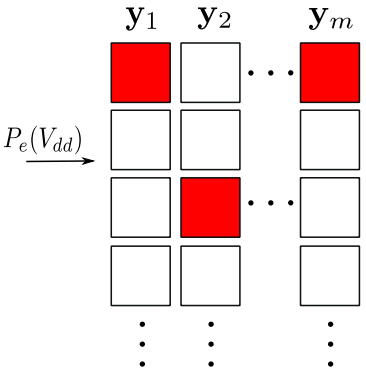


FIGURE 2.33: Faulty buffer memory

The receive symbols are stored in the buffering memory in fixed point representation. That is, for each complex y_i, two fixed point numbers for real and imaginary parts are stored, both possibly containing memory errors. The fixed point representation is defined by two parameters:

- d - length (number of bits) of integer part.
- f - length (number of bits) of fractional part.

The length l of the fixed point number is $l = d + f$. Therefore any bit stored in the buffering memory can be flipped with the same probability P_e. For a fixed point number of length l, each bit position from 0 to $l-1$ can be flipped with same probability P_e.

Hence, the probability to have k bits flipped is given by

$$P(k) = \binom{l}{k} P_e^k (1 - P_e)^{l-k} \tag{2.143}$$

The value of the error at a particular bit position j depends on the fixed point format $[d].[f]$ and the error free bit value y_i^j of i-th element of the receive symbol vector $\mathbf{y}$

$$e_i^j = \begin{cases} +2^{(j-f)} & \text{when } y_i^j = 0 \\ -2^{(j-f)} & \text{when } y_i^j = 1 \end{cases} \tag{2.144}$$

The pdf of the i-th element of the receive vector after the memory is in general given as

$$p(\bar{y}_i) = \sum_{k=0}^{l} P(k) p(y_i, k) \tag{2.145}$$

The pdf $p(y_i, 0)$ defines the distribution of $\bar{y}_i$ when there is no bit flip and in this case $\bar{y}_i = y_i$. The pdf $p(y_i, 1)$ defines the distribution of $\bar{y}_i$ when there is just a single bit flip. This pdf is given as

$$p(y_i, 1) = \sum_{j=0}^{l-1} p^j(y_i, 1) \tag{2.146}$$

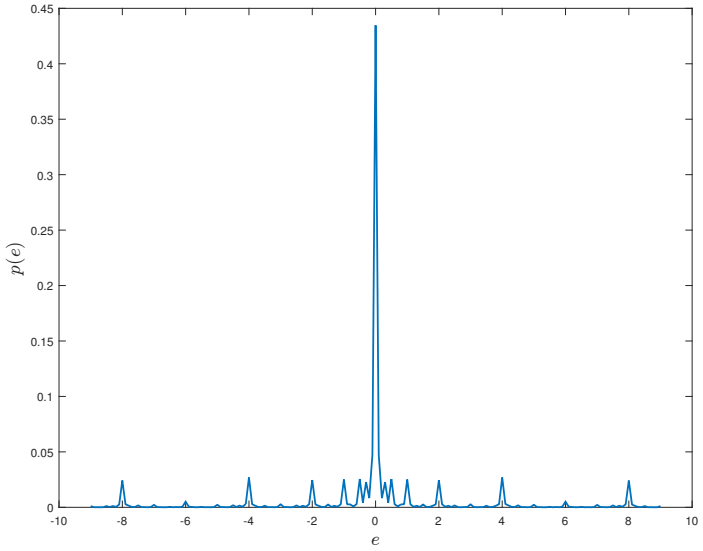


FIGURE 2.34: Pdf of memory error

where $p^j(y_i, 1)$ is the pdf of $\bar{y}_i$ given the error at bit position j, where the error value is given in Eq. 2.144 Defining $p^{j,g}(y_i, 2)$ as the pdf of $\bar{y}_i$ given two bit flips at bit positions j and g, the pdf of $\bar{y}_i$ given two bit flips is

$$p(y_i, 2) = \frac{1}{2!} \sum_{j=0}^{l-1} \sum_{\substack{g=0, \\ g \neq j}}^{l-1} p^{j,g}(y_i, 2) \tag{2.147}$$

In general, in case of k bit flips

$$p(y_i, k) = \frac{1}{k!} \sum_{j_1=0}^{l-1} \sum_{\substack{j_2=0, \\ j_2 \neq j_1}}^{l-1} \cdots \sum_{\substack{j_k=0, \\ j_k \neq j_1, \\ \cdots, j_k \neq j_{k-1}}}^{l-1} p^{j_1, j_2, \cdots, j_k}(y_i, k) \tag{2.148}$$

As P_e is typically a low value, the probability to have multiple bit flips within same y_i is negligible. Figure 2.34 illustrates the probability density function of the memory errors for the [4].[8] fixed point format and $P_e = 10^{-1}$. It can be observed that even for such high bit flip probability, the pdf contains prominent peaks at zero and $\pm 2^{(j-f)}$. Indeed, this pdf can be approximated by a discrete probability mass function with nonzero values at 0 and $\pm 2^{(j-f)}$, $j = 0, \ldots, l-1$, that correspond to single bit flips at bit positions $j = 0, \ldots, l-1$.

Hence, pdf of i-th element of receive vector after the memory $\bar{y}_i = y_i + e_i^j$, j $= 0, \ldots, l-1$ is

$$\begin{aligned} p(\bar{y}_i) &= P(0)p(y_i, 0) + P(1)p(y_i, 1) \\ &= P(0)p(y_i, 0) + P(1) \sum_{j=0}^{l-1} p^j(y_i, 1) \end{aligned} \tag{2.149}$$

where $P(0) = (1 - P_e)^l$ and $P(1) = P_e(1 - P_e)^{l-1}$.

2.3.5 Fault-based attacks and countermeasures

Cryptographic algorithms are employed in most contemporary systems. An example of such algorithms are ciphers, constructed such that the secret key can not be obtained by any other means except the brute force guessing, and the latter is not realizable in a practical amount of time [131].

Unfortunately, it has been recently shown that even though the algorithm itself is mathematically invincible, information that can be used to guess the secret key in a practical amount of time, can be gained from its actual implementation [30]. For example, power consumption of the circuit can be related to the individual steps of the encryption algorithm which in turn can be correlated to the secret key. Such attacks are referred to as side channel attacks [132].

Fault-based attacks are a sub-class of side-channel attacks [31]. By maliciously injecting faults to a specific location of a circuit that implements the cipher the attacker obtains a distorted cipher-text. By comparing the distorted cipher-text and the correct one the number of key candidates may be drastically reduced, such that the brute force search becomes feasible. This technique is denoted differential fault analysis (DFA) [32, 133]. A number of fault-based attacks on classical ciphers has been reported [134]. A number of fault-based attacks on the advanced encryption standard (AES) implementation has been reported in [34, 38, 135]. The attacks on state-of-the-art lightweight ciphers, such as LED [136] and PRESENT [137] has been reported in [35, 39–41]. The attacks can be mounted with very few fault injections, given the ability of the attacker to precisely control the time and location of the injection.

The methods to inject fault can be classified based on their cost and precision. The low cost methods comprise power supply variation, tampering the clock signal, overheating the circuit, illumination with high energy light source such as ultraviolet lamp [31]. Reducing the supply voltage introduces transient faults. Their impact grows with lower supply voltage. This method is not precise, but has been reported effective on large circuits [138]. Altering the clock cycle length results in multiple errors in stored byte or multiple bytes [139]. Temperature rise has been reported to cause multiple bit flips in a dynamic random access memories [140]. The main advantage of this method is its simplicity. The main drawback is the risk of destroying the circuit.

The high cost and precision methods include using laser or x-ray beams. Commercially available fault injection workstations contain a laser emitter, focusing lens and stepper motors that allow very precise targeting of circuits [31].

Countermeasures against fault-based attacks include protective techniques that complicate fault-injection, such as shielding or placing the critical structures into physical locations that are hard to access, and methods to detect the faults once they happen [30, 31]. The detection approaches can be further subdivided into methods that identify physical fault-injection attempts and error-detection techniques working on the logical level. Instances of physical detectors are light sensors which are activated when the circuit packaging is being opened in order to gain access to its internal structures, or

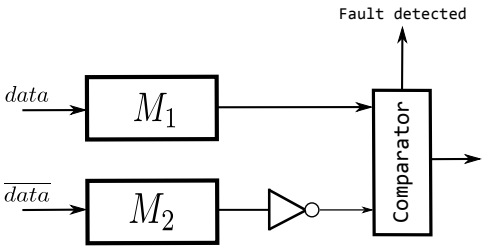


FIGURE 2.35: Simple duplication with complementary redundancy

voltage-level detectors that report attempts to lower the power-supply voltage in order to introduce delay faults. These countermeasures are complicating the analysis, but can be overcome by a sophisticated attacker with access to state-of-the-art equipment.

Logic-level detection employs hardware redundancy, time redundancy and information redundancy. Hardware redundancy is based on NMR and its variations [30]. Examples of these methods are simple duplication with comparison where results of two identical cryptographic blocks are compared and multiple duplication with comparison which is a standard NMR with fault detection. Simple duplication with complementary redundancy (SDCR) is depicted in Fig. 2.35. The duplicated circuit operates on the complemented input. In case the outputs of the two circuits match, the attack is detected. SDCR protects against multiple errors, since it is difficult to inject different errors with complementary effects.

Time redundancy is implemented by running the circuit several times with the same input and comparing the results. One of such method is referred to as simple or multiple time redundancy with comparison depending on the number of redundant computations [141]. Recomputing with swapped operands is performed by first computing the result with big endian and little endian bits of the input swapped. Finally, with recomputing with shifted operands [142], the cryptographic function is recomputed with the operands shifted by a given number of bits. The result is shifted backward and compared to the original result.

The main drawback of hardware redundancy methods is the introduced significant area and power overhead. The drawback of time based redundancy is in the time penalty, introduced by the need of running the same encryption or decryption algorithm several times.

Information redundancy is based on error-detecting or error-correcting codes [5] and its application to circuit protection is treated in detail in Chapter 5.

Chapter 3

Detection of virtual massive MIMO systems

Unique word signaling introduces correlations between individual sub-carriers. Therefore, the UW system model (Eq. 2.55) can be interpreted as a MIMO system with the number of virtual receive antennas given by the number of sub-carriers N. As N is a large number ($N = 64$ in 802.11n) [51], UW system can be treated as a virtual massive MIMO system from the detection viewpoint.

Unique word signaling has been initially introduced for single-antenna systems. The inherent coding across sub-carriers results in BER performance gain compared to SISO CP-OFDM [10, 11]. MIMO-like system model allows linear and maximum likelihood MIMO detection [12], which is not applicable to SISO CP-OFDM. However, exponential complexity of sphere decoding restricted its use only to UW-OFDM systems with low number of sub-carriers [12, 13].

UW-OFDM clearly outperforms CP-OFDM in the SISO case [9–11]. However, as current standards employ MIMO with M_t transmit and M_r receive physical antennas [51, 63], it makes sense to employ it in UW-OFDM too. This chapter introduces the generalization of UW-OFDM scheme to a MIMO system and compares its performance against a MIMO CP-OFDM system. The comparison is conducted for both linear and nonlinear receivers. The employment of multiple antennas increases the system size further and MIMO UW-OFDM can be regarded as a virtual massive MIMO system with hundreds of virtual antennas. It is therefore impossible to apply SD to MIMO UW-OFDM. Thus, direct comparison of ML performance cannot be done. Hence, it is resorted to quasi-ML algorithms initially introduced for massive MIMO systems [143].

First, likelihood ascent search algorithm [98, 99] is adapted to MIMO UW-OFDM.

Next, concentrating on the virtual massive MIMO interpretation of MIMO UW-OFDM, this chapter addresses the problem of memory dominance of the receiver. It has been shown that the traditional small-scale MIMO receivers are memory dominated, as different buffering memories occupy up to 50% of chip area [144]. In MIMO UW-OFDM this memory dominance is even more pronounced due to much larger system size. As discussed in the background chapter of this thesis, the aggressive technology downscaling denies the assumption of deterministic hardware operation. Therefore, the detection

will be modified such that it takes the possibility of faulty memory into account. The proposed method is developed using the statistical model of channel noise and memory errors, outlined in Sec. 2.3.4. The novelty of the proposed solution is emphasized by its ability to handle multiple memory errors, whereas previous schemes assumed single memory errors only [6]. It is to note that the proposed algorithm is suited to massive MIMO systems in the general sense. MIMO UW-OFDM is taken here as an example of a system employing several hundreds of (virtual) receive antennas.

The MIMO UW-OFDM is introduced and its performance comparison to MIMO CP-OFDM is treated in Sec. 3.1. Memory error resilient detection that builds up on the detection methods introduced for MIMO-UW OFDM in Sec. 3.1 is outlined in Sec. 3.2.

3.1 MIMO unique word OFDM

The generalization of UW-OFDM to a MIMO system is performed similarly to the CP-OFDM case (Sec. 2.1.2). There are again M_t parallel transmit data symbol vectors $\tilde{\mathbf{d}} \in \mathbb{C}^{N_d \times 1}$. Grouping to the compound transmit frequency domain data symbol vector with respect to sub-carriers and transmit antennas is done as follows:

$$\tilde{\mathbf{d}} = \begin{bmatrix} \tilde{\mathbf{d}}(0) \\ \vdots \\ \tilde{\mathbf{d}}(N_d-1) \end{bmatrix}_{N_d M_t \times 1} = \begin{bmatrix} \tilde{d}_1(0) \\ \vdots \\ \tilde{d}_{M_t}(0) \\ \\ \tilde{d}_1(1) \\ \vdots \\ \tilde{d}_{M_t}(1) \\ \\ \vdots \\ \\ \tilde{d}_1(N_d-1) \\ \vdots \\ \tilde{d}_{M_t}(N_d-1) \end{bmatrix} \tag{3.1}$$

The generation of the encoded compound frequency domain symbol vector $\tilde{\mathbf{x}} \in \mathbb{C}^{N M_t \times 1}$ is done as in Eq. 2.45. The difference to SISO case is the generator matrix $\check{\mathbf{G}}$ obtained by augmenting the SISO generator matrix to the appropriate size by the Kronecker product

with $\mathbf{I}_{M_t}$.

$$\check{\mathbf{G}} = \mathbf{G} \otimes \mathbf{I}_{M_t} = \tag{3.2}$$
$$= \begin{bmatrix} g_{11}\mathbf{I}_{M_t} & g_{12}\mathbf{I}_{M_t} & \cdots & g_{1N_d}\mathbf{I}_{M_t} \\ g_{21}\mathbf{I}_{M_t} & g_{22}\mathbf{I}_{M_t} & \cdots & g_{2N_d}\mathbf{I}_{M_t} \\ \vdots & \vdots & \vdots & \vdots \\ g_{N1}\mathbf{I}_{M_t} & g_{N2}\mathbf{I}_{M_t} & \cdots & g_{NN_d}\mathbf{I}_{M_t} \end{bmatrix}_{(N_d+N_r)M_t \times N_d M_t}$$

The encoded frequency domain compound symbol vector is therefore given as

$$\tilde{\mathbf{x}} = \check{\mathbf{G}}\tilde{\mathbf{d}} = \tag{3.3}$$
$$= \check{\mathbf{G}} \begin{bmatrix} \tilde{d}_1(0) \\ \vdots \\ \tilde{d}_{M_t}(0) \\ \\ \tilde{d}_1(1) \\ \vdots \\ \tilde{d}_{M_t}(1) \\ \\ \vdots \\ \\ \tilde{d}_1(N_d-1) \\ \vdots \\ \tilde{d}_{M_t}(N_d-1) \end{bmatrix}$$

Random initialization introduced in Sec. 2.1.2.2 produces random non-systematic generator matrices $\mathbf{G}''$ every time the optimization algorithm is run. This allows encoding each of the M_t data streams by a different generator matrix $\mathbf{G}''_m$, where $m = 1, \ldots, M_t$. In this case the compound frequency domain transmit symbol vector is obtained as

$$\tilde{\mathbf{x}} = \check{\mathbf{G}}''_s\tilde{\mathbf{d}} \tag{3.4}$$

where $\check{\mathbf{G}}''_s$ is generated as follows

$$\check{\mathbf{G}}''_s = \mathbf{G}''_1 \otimes \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + \cdots + \mathbf{G}''_{M_t} \otimes \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.5}$$

The transformation to the time domain is performed as in the CP case by the M_t parallel IFFT modules. Therefore, the compound time domain symbol vector is obtained similar

to Sec. 2.2.2 as

$$\mathbf{x} = \frac{1}{N}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\check{\mathbf{G}}\tilde{\mathbf{d}} \tag{3.6}$$

Next, the vector of non-zero UWs is added to $\mathbf{x}$, similar to the SISO case.

$$\mathbf{x}' = \mathbf{x} + \begin{bmatrix}\mathbf{0} & \mathbf{u}^H\end{bmatrix}^H \tag{3.7}$$

Here, the compound vector of non-zero UWs is given as

$$\mathbf{u} = \begin{bmatrix}\mathbf{u}(N_d)\\ \vdots \\ \mathbf{u}(N-1)\end{bmatrix}_{N_r M_t \times 1} = \begin{bmatrix} u_1(N_d)\\ \vdots \\ u_{M_t}(N_d) \\ \\ u_1(N_d+1) \\ \vdots \\ u_{M_t}(N_d+1) \\ \\ \vdots \\ \\ u_1(N-1) \\ \vdots \\ u_{M_t}(N-1)\end{bmatrix}$$

The frequency domain version of the compound unique word vector is obtained by FFT as given below

$$\tilde{\mathbf{u}} = (\mathbf{F}_N \otimes \mathbf{I}_{M_t}) \begin{bmatrix}\mathbf{0} & \mathbf{u}^H\end{bmatrix}^H \tag{3.8}$$

Therefore, the compound time domain transmit symbol vector is the following

$$\mathbf{x}' = \frac{1}{N}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})(\check{\mathbf{G}}\tilde{\mathbf{d}} + \tilde{\mathbf{u}}) \tag{3.9}$$

The latter symbol vector is transmitted over the same MIMO multipath channel, discussed in Sec. 2.2.2. The linear block convolution matrix is given in the MIMO case as

$$\mathbf{H}_{toep} = \begin{bmatrix} \mathbf{H}_0 & & & & \\ \vdots & \ddots & & & \\ \mathbf{H}_{L-1} & \cdots & \mathbf{H}_0 & & \\ & \ddots & \vdots & \ddots & \\ & & \mathbf{H}_{L-1} & \cdots & \mathbf{H}_0 \end{bmatrix}_{(N+N_r)M_r \times (N+N_r)M_t} \tag{3.10}$$

Here again, as with SISO UW-OFDM, the ISI terms fall within the duration of the UW vector $\mathbf{u}(t-1)$ of $\mathbf{x}'(t-1)$. At the receiver side, neglecting the AWGN vector, the

compound time domain receive symbol vector $\mathbf{y}(t)$ is given similar to Eq. 2.48:

$$\mathbf{y}(t) = \mathbf{H}_{toep}(t) \begin{bmatrix} \mathbf{u}(t-1) \\ \mathbf{d}(t) \\ \mathbf{u}(t) \end{bmatrix} \tag{3.11}$$

The structure of the transmitted compound OFDM symbol vector is equivalent to the signal with added cyclic prefix in case of CP-OFDM. The inclusion of $\mathbf{u}(t-1)$ is modeled by multiplication with a copy matrix $\check{\mathbf{\Theta}}_c$, similar to cyclic prefix insertion matrix in Eq. 2.81:

$$\check{\mathbf{\Theta}}_c = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{N_r} \\ \mathbf{I}_{N-N_r} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N_r} \end{bmatrix} \otimes \mathbf{I}_{M_t} \tag{3.12}$$

where $\check{\mathbf{\Theta}}_c$ is obtained from the SISO copy matrix, augmenting it to the appropriate size by the Kronecker product with the identity matrix of size M_t.

The t-th compound time domain receive symbol vector, neglecting the AWGN, is given as

$$\mathbf{y}(t) = \check{\mathbf{\Theta}}_x \mathbf{H}_{toep}(t) \check{\mathbf{\Theta}}_c \mathbf{x}'(t) \tag{3.13}$$

where $\check{\mathbf{\Theta}}_x$ is the UW extraction matrix similar to cyclic prefix removal matrix in Eq. 2.84. It is obtained from SISO UW extraction matrix by augmenting it to the appropriate size by the Kronecker product with the identity matrix of size M_r:

$$\check{\mathbf{\Theta}}_x = \begin{bmatrix} \mathbf{0}_{N \times N_r} & \mathbf{I}_N \end{bmatrix} \otimes \mathbf{I}_{M_r} \tag{3.14}$$

Clearly, the matrix given below

$$\mathbf{H}_{cir} = \check{\mathbf{\Theta}}_x \mathbf{H}_{toep} \check{\mathbf{\Theta}}_c \tag{3.15}$$

is the same block circulant matrix as in Eq. 2.86, as the UW insertion and extraction matrices transform the block convolution matrix in the same way as the CP insertion and removal matrices. Therefore, the block eigenvalue decomposition of $\mathbf{H}_{cir}$ is the same as in the CP-OFDM case.

Thereby, dropping the time index t, the compound receive frequency domain symbol vector is obtained at the receiver side by M_r parallel FFT blocks as

$$\begin{aligned} \tilde{\mathbf{y}} &= (\mathbf{F}_N \otimes \mathbf{I}_{M_r}) \mathbf{H}_{cir} \mathbf{x}' + (\mathbf{F}_N \otimes \mathbf{I}_{M_r}) \mathbf{n} \\ &= \frac{1}{N} (\mathbf{F}_N \otimes \mathbf{I}_{M_r}) \mathbf{H}_{cir} (\mathbf{F}_N^H \otimes \mathbf{I}_{M_t}) (\check{\mathbf{G}} \tilde{\mathbf{d}} + \tilde{\mathbf{u}}) + (\mathbf{F}_N \otimes \mathbf{I}_{M_r}) \mathbf{n} \end{aligned} \tag{3.16}$$

Eigenvalue decomposition of $\mathbf{H}_{cir}$ results in the following compound receive frequency domain symbol vector

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}_b \check{\mathbf{G}} \tilde{\mathbf{d}} + \tilde{\mathbf{H}}_b \tilde{\mathbf{u}} + \tilde{\mathbf{n}} \tag{3.17}$$

where $\tilde{\mathbf{H}}_b$ is the block diagonal matrix, with blocks containing MIMO flat fading channel coefficients relative to the sub-carriers, as given in Eq. 2.88.

Subtracting the known part $\tilde{\mathbf{H}}_b\tilde{\mathbf{u}}$ at the receiver side results in final MIMO UW-OFDM system equation

$$\tilde{\mathbf{y}}' = \tilde{\mathbf{H}}_b\check{\mathbf{G}}\tilde{\mathbf{d}} + \tilde{\mathbf{n}} \tag{3.18}$$

3.1.1 LMMSE detection

The system equation (Eq. 3.18) for MIMO UW-OFDM looks similar to the one of the MIMO CP-OFDM. However, the sub-carriers are not independent anymore. Recall the general form of the LMMSE estimate of the transmit frequency domain symbol vector $\tilde{\mathbf{x}}$:

$$\hat{\tilde{\mathbf{x}}} = \left(\tilde{\mathbf{H}}_b^H\mathbf{C}_{nn}^{-1}\tilde{\mathbf{H}}_b + \mathbf{C}_{xx}^{-1}\right)^{-1}\tilde{\mathbf{H}}_b^H\mathbf{C}_{nn}^{-1}\tilde{\mathbf{y}}' \tag{3.19}$$

where the noise covariance matrix is $\mathbf{C}_{nn} = E\left\{\tilde{\mathbf{n}}\tilde{\mathbf{n}}^H\right\} = N\sigma_n^2\mathbf{I}$ and the transmit symbol vector covariance matrix is $\mathbf{C}_{xx} = E\left\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^H\right\} = \sigma_d^2\check{\mathbf{G}}\check{\mathbf{G}}^H$. As in UW-OFDM only the data part $\tilde{\mathbf{d}}$ of $\tilde{\mathbf{x}}$ is of interest, its LMMSE estimate is obtained as follows:

$$\begin{aligned}
\hat{\tilde{\mathbf{d}}} &= \check{\mathbf{G}}^{-1}\left(\tilde{\mathbf{H}}_b^H\tilde{\mathbf{H}}_b + \left(\check{\mathbf{G}}\check{\mathbf{G}}^H\right)^{-1}\frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1}\tilde{\mathbf{H}}_b^H\tilde{\mathbf{y}}' \\
&= \left(\tilde{\mathbf{H}}_b^H\tilde{\mathbf{H}}_b\check{\mathbf{G}} + (\check{\mathbf{G}}^H)^{-1}\check{\mathbf{G}}^{-1}\check{\mathbf{G}}\frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1}\tilde{\mathbf{H}}_b^H\tilde{\mathbf{y}}' \\
&= \left(\check{\mathbf{G}}^H\tilde{\mathbf{H}}_b^H\tilde{\mathbf{H}}_b\check{\mathbf{G}} + \frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1}\check{\mathbf{G}}^H\tilde{\mathbf{H}}_b^H\tilde{\mathbf{y}}'
\end{aligned} \tag{3.20}$$

In case of random non-systematic generator matrices $\check{\mathbf{G}}''_m$, $m = 1, \ldots M_t$ the LMMSE estimate is of the form

$$\hat{\tilde{\mathbf{d}}} = \left(\check{\mathbf{G}}_s''^H\tilde{\mathbf{H}}_b^H\tilde{\mathbf{H}}_b\check{\mathbf{G}}''_s + \frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1}\check{\mathbf{G}}_s''^H\tilde{\mathbf{H}}_b^H\tilde{\mathbf{y}}' \tag{3.21}$$

where $\check{\mathbf{G}}''_s$ is given in Eq. 3.5.

Due to the correlation between sub-carriers introduced by generator matrix $\check{\mathbf{G}}$, the LMMSE estimate is calculated once for all sub-carriers and data streams, as opposed to the CP-OFDM case. However, it is to note that the size of the matrix to be inverted has increased from $M_r \times M_t$ to less favorable $N_dM_r \times N_dM_t$. This increase in complexity can be tackled as discussed further.

3.1.1.1 Complexity-reduced LMMSE for systematic MIMO UW-OFDM

The complexity of the LMMSE estimator for MIMO UW-OFDM is dominated by the matrix inversion in Eq. 3.20. The size of the matrix to be inverted is $N_dM_r \times N_dM_t$. Exploiting the structure of the systematic generator matrix $\check{\mathbf{G}}$, reduced-complexity version of the LMMSE is obtained similar to SISO UW-OFDM [13].

The receive frequency domain symbol vector $\tilde{\mathbf{y}}'$ has to be resorted by multiplying with transposed permutation matrix $\check{\mathbf{P}}^T$. This matrix is obtained from the SISO permutation

matrix by the Kronecker product with the identity matrix of size M_r

$$\check{\mathbf{P}}^T = \mathbf{P}^T \otimes \mathbf{I}_{M_R} \tag{3.22}$$

The resorted receive symbol vector is then given as

$$\tilde{\mathbf{y}}' = \check{\mathbf{P}}^T \frac{1}{N}(\mathbf{F}_N \otimes \mathbf{I}_{M_r})\mathbf{H}_{cir}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\check{\mathbf{G}}\tilde{\mathbf{d}} + \check{\mathbf{P}}^T\tilde{\mathbf{n}} \tag{3.23}$$

The resorted $\tilde{\mathbf{H}}_b$ is in this case defined as follows

$$\tilde{\mathbf{H}}_b = \check{\mathbf{P}}^T \frac{1}{N}(\mathbf{F}_N \otimes \mathbf{I}_{M_r})\mathbf{H}_{cir}(\mathbf{F}_N^H \otimes \mathbf{I}_{M_t})\check{\mathbf{P}} \tag{3.24}$$

After resorting, the block channel matrix $\check{\mathbf{H}}_b$ is easily decomposed into information and redundancy parts:

$$\tilde{\mathbf{H}}_b = \begin{bmatrix} \tilde{\mathbf{H}}_{b,d} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{H}}_{b,r} \end{bmatrix} \tag{3.25}$$

where $\tilde{\mathbf{H}}_{b,d} \in \mathbb{C}^{N_d M_r \times N_d M_t}$ and $\tilde{\mathbf{H}}_{b,r} \in \mathbb{C}^{N_r M_r \times N_r M_t}$.

Using the unsorted systematic generator matrix $\check{\mathbf{G}}_u$,

$$\check{\mathbf{G}}_u = \begin{bmatrix} \mathbf{I} \\ \mathbf{T} \end{bmatrix} \otimes \mathbf{I}_{M_t} \tag{3.26}$$

it can be easily shown that

$$\tilde{\mathbf{H}}_b\check{\mathbf{G}}_u = \begin{bmatrix} \tilde{\mathbf{H}}_{b,d} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{H}}_{b,r} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{T} \end{bmatrix} \otimes \mathbf{I}_{M_t} = \begin{bmatrix} \tilde{\mathbf{H}}_{b,d} \\ \tilde{\mathbf{H}}_{b,r}(\mathbf{T} \otimes \mathbf{I}_{M_t}) \end{bmatrix} \tag{3.27}$$

The matrix to be inverted from Eq. 3.20 can be reformulated as

$$\begin{aligned} &\left(\check{\mathbf{G}}_u^H\tilde{\mathbf{H}}_b^H\tilde{\mathbf{H}}_b\check{\mathbf{G}}_u + \frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1} = \\ &= \left(\tilde{\mathbf{H}}_{b,d}^H\mathbf{H}_{b,d} + (\mathbf{T} \otimes \mathbf{I}_{M_t})^H\tilde{\mathbf{H}}_{b,r}^H\tilde{\mathbf{H}}_{b,r}(\mathbf{T} \otimes \mathbf{I}_{M_t}) + \frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}\right)^{-1} \end{aligned} \tag{3.28}$$

Next, two block diagonal matrices are defined

$$\mathbf{D}_d = \tilde{\mathbf{H}}_{b,d}^H\tilde{\mathbf{H}}_{b,d} + \frac{N\sigma_n^2}{\sigma_d^2}\mathbf{I}, \qquad \mathbf{D}_d \in \mathbb{C}^{N_d M_r \times N_d M_t} \tag{3.29}$$

$$\mathbf{D}_r = \tilde{\mathbf{H}}_{b,r}^H, \qquad \mathbf{D}_r \in \mathbb{C}^{N_r M_r \times N_r M_t} \tag{3.30}$$

Applying the matrix inversion lemma [145] to the right hand side of Eq. 3.28, and renaming $\mathbf{T} \otimes \mathbf{I}_{M_t} = \check{\mathbf{T}}$, reduced complexity LMMSE estimate of the transmit symbol vector is defined as follows:

$$\hat{\tilde{\mathbf{d}}} = (\mathbf{D}_d^{-1} - \mathbf{D}_d^{-1}\check{\mathbf{T}}^H(\check{\mathbf{T}}\mathbf{D}_d^{-1}\check{\mathbf{T}}^H + \mathbf{D}_r^{-1})^{-1}\check{\mathbf{T}}\mathbf{D}_d^{-1}) \begin{bmatrix} \tilde{\mathbf{H}}_{b,d}^H & \check{\mathbf{T}}^H\tilde{\mathbf{H}}_{b,r}^H \end{bmatrix} \tilde{\mathbf{y}}' \tag{3.31}$$

The inversion of block diagonal matrices $\mathbf{D}_d$ and $\mathbf{D}_r$ is reduced to $N_d + N_r$ matrix inversions of size $M_r \times M_t$. The same number of matrix inversions is performed in the MIMO CP-OFDM LMMSE detection. Additionally, the inversion of $N_r M_r \times N_r M_t$ -sized matrix $(\check{\mathbf{T}}\mathbf{D}_d^{-1}\check{\mathbf{T}}^H + \mathbf{D}_r^{-1})^{-1}$ has to be performed. Thus, the complexity is comparable for both methods, due to the systematic structure of the UW-OFDM generator matrix. Obviously, this simplification is not applicable in the case of non-systematic MIMO UW-OFDM.

The complexities of both algorithms are compared in terms of number of matrix inversions. The results are depicted in Table 3.1.

TABLE 3.1: Complexity comparison

Algorithm	MIMO-CP LMMSE		MIMO-UW Red. Compl. LMMSE	
Operation	Size	Count	Size	Count
Inversion	$M_r \times M_t$	N	$M_r \times M_t$	$N_d + N_r = N$
Inversion	$\emptyset$	$\emptyset$	$N_r M_r \times N_r M_t$	1

3.1.2 Sphere decoding for MIMO UW-OFDM

By regarding $\tilde{\mathbf{H}}_d\mathbf{G}$ as a virtual channel matrix, SISO UW-OFDM can be regarded as a virtual MIMO system. Therefore, SD has been applied to SISO UW-OFDM detection [12]. However, the exponential complexity of SD restricted its use only to systems with a low number ($N = 24$) of sub-carriers [12], [13]. MIMO UW-OFDM can be interpreted as a virtual massive MIMO system, where the virtual channel matrix is $\tilde{\mathbf{H}}_b\check{\mathbf{G}}$. In this case the ML solution is given as

$$\hat{\tilde{\mathbf{d}}} = \underset{\tilde{\mathbf{d}} \in \mathbb{A}^{N_d M_t}}{\operatorname{argmin}} \left\| \tilde{\mathbf{y}}' - \tilde{\mathbf{H}}_b\check{\mathbf{G}}\tilde{\mathbf{d}} \right\|_2^2 \tag{3.32}$$

The size of the transmit symbol vector is increased to $N_d M_t \times 1$, leading to a very large tree and prohibitive complexity.

Therefore, SD will only be used for MIMO CP-OFDM detection in this work. It will provide a ML performance reference for MIMO-UW OFDM.

3.1.3 LAS for MIMO UW-OFDM

Since MIMO UW-OFDM can be interpreted as a virtual massive MIMO system, LAS detection algorithm can be directly applied. The MIMO UW-OFDM system model can be transformed to real numbers

$$\mathbf{y}' = \mathbf{H}\mathbf{d} + \mathbf{n} \tag{3.33}$$

with the real-value decomposition

$$\begin{bmatrix} \Re(\tilde{\mathbf{y}}') \\ \Im(\tilde{\mathbf{y}}') \end{bmatrix} = \begin{bmatrix} \Re(\tilde{\mathbf{H}}_b\check{\mathbf{G}}) & -\Im(\tilde{\mathbf{H}}_b\check{\mathbf{G}}) \\ \Im(\tilde{\mathbf{H}}_b\check{\mathbf{G}}) & \Re(\tilde{\mathbf{H}}_b\check{\mathbf{G}}) \end{bmatrix} \begin{bmatrix} \Re(\tilde{\mathbf{d}}) \\ \Im(\tilde{\mathbf{d}}) \end{bmatrix} + \begin{bmatrix} \Re(\tilde{\mathbf{n}}) \\ \Im(\tilde{\mathbf{n}}) \end{bmatrix} \tag{3.34}$$

Now the size of vectors and matrices has doubled: $\mathbf{y}' \in \mathbb{R}^{(2M_rN\times 1)}$, $\mathbf{H} \in \mathbb{R}^{(2M_rN\times 2M_tN_d)}$, $\mathbf{d} \in \mathbb{R}^{(2M_tN_d\times 1)}$, $\mathbf{n} \in \mathbb{R}^{(2M_rN\times 1)}$.

LAS sub-procedures, with necessary extensions relative to the original algorithm, are detailed in next section.

3.1.3.1 Maximum a posteriori probability LAS

The original LAS algorithm starts with the initial solution given by

$$\mathbf{d}^{(0)} = \mathcal{Q}(\hat{\mathbf{d}}_{\text{lmmse}}) \tag{3.35}$$

where $\mathcal{Q}(.)$ quantizes the LMMSE estimate to the nearest neighbor in $\mathbb{A}^{2N_dM_t}$ [143].

It will be shown in simulation reported later in Sec. 3.1.4 that the quality of the soft outputs computed according to Eq. 3.55 is not adequate for coded MIMO UW-OFDM. The performance of soft decision decoded LAS is worse than the performance of soft decision decoded LMMSE. As LAS algorithm quantizes the LMMSE estimate, it ignores the LLR provided by latter (Eq. 2.105). Therefore it makes sense to modify the initial LAS algorithm such that it takes the soft output of LMMSE as its initial solution.

As LAS is essentially an improvement of LMMSE solution, the LLR provided by LMMSE can be used as a priori information in deciding which symbols have to be updated during LAS search procedures (Sec. 2.2.3.2.2). The detection method that uses a-priori probability of the transmit symbol is referred to as maximum a posteriori probability (MAP) [146]. The MAP estimate of data symbol vector $\mathbf{d}$ is given as

$$\hat{\mathbf{d}}_{MAP} = \underset{\mathbf{d}\in\mathbb{A}^{2N_dM_t}}{\text{argmax}}\ P(\mathbf{d}|\mathbf{y}', \mathbf{H}) \tag{3.36}$$

Using Bayes theorem, Eq. 3.36 can be reformulated as

$$\begin{aligned}\hat{\mathbf{d}}_{MAP} &= \underset{\mathbf{d}\in\mathbb{A}^{2N_dM_t}}{\text{argmax}} \left(p(\mathbf{y}'|\mathbf{d}, \mathbf{H}) \frac{P(\mathbf{d})}{p(\mathbf{y}')} \right) \\ &= \underset{\mathbf{d}\in\mathbb{A}^{2N_dM_t}}{\text{argmax}} \left(p(\mathbf{y}'|\mathbf{d}, \mathbf{H}) P(\mathbf{d}) \right)\end{aligned} \tag{3.37}$$

where the second equality is given by the fact that the pdf of $\mathbf{y}'$ does not depend on $\mathbf{d}$. The first term in the second equality is exactly the likelihood in Eq. 2.111. Taking the logarithm of Eq. 3.37, the maximization is turned into minimization

$$\hat{\mathbf{d}}_{MAP} = \underset{\mathbf{d}\in\mathbb{A}^{2N_dM_t}}{\text{argmin}} \left(\frac{\|\mathbf{y}' - \mathbf{H}\mathbf{d}\|_2^2}{\sigma_n^2} - \log P(\mathbf{d}) \right) \tag{3.38}$$

where the second term is the a priori probability of the transmit vector $\mathbf{d}$. It is given as the product of probabilities of individual symbols.

$$P(\mathbf{d}) = \prod_{j=1}^{2N_dM_t} P(d_j) \tag{3.39}$$

Assuming 4-QAM with Gray mapping, the values of symbols represent the bit values, $d_{j,b} = d_j$. Symbol probabilities $P(d_j = +1)$ and $P(d_j = -1)$ are obtained from LMMSE LLR given in Eq. 2.107 as [55]

$$P(d_j = +1) = \frac{\exp(\bar{L}_j)}{1 + \exp(\bar{L}_j)} \tag{3.40}$$

$$P(d_j = -1) = \frac{1}{1 + \exp(\bar{L}_j)} \tag{3.41}$$

Hence, the initial MAP cost is given as

$$C^{(0)} = \|\mathbf{y}' - \mathbf{H}\mathbf{d}^{(0)}\|_2^2 - \log(P(\mathbf{d}^{(0)})) \tag{3.42}$$

If the a priori probabilities of the individual symbols are the same, the second term in minimization of Eq. 3.38 is neglected and the MAP cost of initial solution vector is identical to the ML cost of the original LAS

$$C^{(0)} = \|\mathbf{y}' - \mathbf{H}\mathbf{d}^{(0)}\|_2^2 = \mathbf{d}^{(0)^T}\mathbf{H}^T\mathbf{H}\mathbf{d}^{(0)} - 2\mathbf{y}^T\mathbf{H}\mathbf{d}^{(0)} \tag{3.43}$$

In 1-LAS, the search in the one symbol neighborhood of the current solution vector is performed by updating one symbol position of current solution vector per iteration. Consider the j-th symbol, $j = 1, \ldots, 2N_dM_t$, changed in $(i+1)$-th iteration. The update is formulated as

$$\mathbf{d}^{(i+1)} = \mathbf{d}^{(i)} + \gamma_j^{(i)}\mathbf{u}_j \tag{3.44}$$

where $\mathbf{u}_j$ is the unit vector with j-th element set to one. For any iteration i, it is necessary that $\mathbf{d} \in \mathbb{A}^{2N_dM_t}$. This implies that $\gamma^{(i)}$ is restricted to certain integer values. For example, in case of 16-QAM, the real-valued symbol alphabet is given as $\mathbb{A} = \{-3, -1, +1, +3\}$ and $\gamma^{(i)}$ is therefore restricted to $\{-6, -4, -2, +2, +4, +6\}$ [143]. In case of 4-QAM, the symbol alphabet is $\mathbb{A} = \{-1, +1\}$ and allowed $\gamma^{(i)}$ values are $\{-2, +2\}$.

Updating a symbol at position j results in the probability change for that symbol from $P(d_j)$ to $P(\bar{d}_j)$. Here, symbol $\bar{d}_j$ represents the sign flipped version of symbol d_j. Introducing matrix $\boldsymbol{\Upsilon} = \mathbf{H}^T\mathbf{H}$ with elements υ_{ij} and vector $\mathbf{z}^{(i)} = \mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{d}^{(i)})$, with elements $z_j^{(i)}$, the change of MAP cost from iteration i to $i+1$ is given by

$$\begin{aligned}
\Delta C_j^{i+1} &= C^{(i+1)} - C^{(i)} \\
&= \|\mathbf{y}' - \mathbf{H}(\mathbf{d}^{(i)} + \gamma_j^{(i)}\mathbf{u}_j)\|_2^2 - \log P(\mathbf{d}^{(i)} + \gamma_j^{(i)}\mathbf{u}_j) - \|\mathbf{y}' - \mathbf{H}\mathbf{d}^{(i)}\|_2^2 + \log P(\mathbf{d}^{(i)}) \\
&= {\gamma_j^{(i)}}^2 \upsilon_{jj} - 2\gamma_j^{(i)} z_j^{(i)} + \log P(d_j) - \log P(\bar{d}_j)
\end{aligned} \tag{3.45}$$

The last equality contains the difference between a priori probabilities of two possible values of the symbol which is changed at position j. The probabilities of symbols at positions different from j do not change from iteration i to iteration $i+1$, and are therefore canceled out in the subtraction. This way, MAP LAS takes the confidence in the individual symbols provided by the LMMSE initial estimate into account.

In order to decrease the the overall MAP cost, the MAP cost change should be negative. Hence, the symbol position p with minimum MAP cost difference is obtained:

$$p = \underset{j=1,\dots,2N_dM_t}{\operatorname{argmin}} \Delta C_j^{i+1} \tag{3.46}$$

If $\Delta C_p^{i+1} < 0$, vectors $\mathbf{d}^{(i)}$ and $\mathbf{z}^{(i)}$ for the $(i+1)$-th iteration are updated as follows

$$\mathbf{d}^{(i+1)} = \mathbf{d}^{(i)} + \gamma_p^{(i)} \mathbf{u}_p \tag{3.47}$$

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} - \gamma_p^{(i)} \boldsymbol{\upsilon}_p \tag{3.48}$$

where $\boldsymbol{\upsilon}_p$ is the p-th column of matrix $\boldsymbol{\Upsilon}$. If $\Delta C_p^{i+1} > 0$, then a 1-neighborhood local minimum (which may be or may be not the global minimum) is reached and the 1-LAS sub-procedure is terminated. The resulting vector $\mathbf{d}$ is the solution of 1-LAS algorithm.

The algorithm then tries to escape from the assumed local minimum and improve the MAP cost further by initiating the 2-LAS sub-procedure (Fig. 2.22). 2-LAS searches for a two-symbol update that would further increase the likelihood. There are $\binom{2N_dM_t}{2}$ two symbol position combinations. Denote the tuple of symbol positions to be updated as $t = \{j, q\}$. The update rule for the $i+1$ iteration can then be written as

$$\mathbf{d}^{(i+1)} = \mathbf{d}^{(i)} + \gamma_j^{(i)} \mathbf{u}_j + \gamma_q^{(i)} \mathbf{u}_q \tag{3.49}$$

The MAP cost difference for the two-symbol update is written as

$$\begin{aligned} &\Delta C_t^{i+1}(\gamma_j^{(i)}, \gamma_q^{(i)}) = C^{(i+1)} - C^{(i)} = \\ &= {\gamma_j^{(i)}}^2 \upsilon_{jj} + {\gamma_q^{(i)}}^2 \upsilon_{qq} - 2\gamma_j^{(i)} \gamma_q^{(i)} \upsilon_{jq} - 2\gamma_j^{(i)} z_j^{(i)} - 2\gamma_q^{(i)} z_q^{(i)} + \\ &+ \log P(d_j) - \log P(\bar{d}_j) + \log P(d_q) - \log P(\bar{d}_q) \end{aligned} \tag{3.50}$$

In order to find the optimum values the brute-force method is to evaluate [143]

$$(\dot{\gamma}_j^{(i)}, \dot{\gamma}_q^{(i)}) = \underset{j,q}{\operatorname{argmin}} \Delta C_t^{i+1}(\gamma_j^{(i)}, \gamma_q^{(i)}) \tag{3.51}$$

In case of 4-QAM, however, search for optimum $\dot{\gamma}_j^{(i)}$ is not necessary, as it may take only values from $\{-2, +2\}$, depending on the sign of $d_j^{(i)}$. The MAP cost differences are computed for all $t = \{j, q\}$-tuples and the tuple $o = \{p, r\}$ with the minimum cost difference is obtained

$$o = \underset{t}{\operatorname{argmin}} \Delta C_t^{m+1}(\gamma_j^{(i)}, \gamma_q^{(i)}) \tag{3.52}$$

Finally, if $\Delta C_o^{i+1}(\gamma_p^{(i)}, \gamma_r^{(i)}) < 0$ the update rule for $\mathbf{d}^{(i+1)}$, $\mathbf{z}^{(i+1)}$ vectors is given by

$$\mathbf{d}^{(i+1)} = \mathbf{d}^{(i)} + \gamma_p^{(i)} \mathbf{u}_p + \gamma_r^{(i)} \mathbf{u}_r \tag{3.53}$$

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} - (\gamma_p^{(i)} \boldsymbol{\upsilon}_p + \gamma_r^{(i)} \boldsymbol{\upsilon}_r) \tag{3.54}$$

2-LAS terminates after a single iteration. If the 2-symbol update improved the MAP

cost, the 1-LAS sub-procedure is initiated again (Fig. 2.22). Otherwise, 3-LAS sub-procedure starts a 3-neighborhood search. The computations are similar to the ones in 2-LAS and can be derived in a straightforward way. If the iteration of 3-LAS is able to improve the MAP cost, 1-LAS is initiated again. If not, the algorithm terminates and the current solution vector $\mathbf{d}$ is declared the final solution.

3.1.3.1.1 Soft output generation

Soft output generation for the original LAS is proposed in [143]. The confidence in each bit is formulated as

$$R_{j,b} \approx \frac{\|\mathbf{y}' - \mathbf{H}\hat{\mathbf{d}}_j^{b=1}\|_2^2 - \|\mathbf{y} - \mathbf{H}\hat{\mathbf{d}}_j^{b=0}\|_2^2}{\|\mathbf{h}_j\|_2^2} \tag{3.55}$$

where $\hat{\mathbf{d}}_j^{b=1}$ and $\hat{\mathbf{d}}_j^{b=0}$ are the LAS solution vectors with b-th bit of symbol j set to 1 or 0, respectively, $\mathbf{h}_j$ is the j-th column of $\mathbf{H}$. It is to note that the soft output in Eq. 3.55 represents a rather coarse approximation of max-log LLR in Eq. 2.103, as other vectors belonging to $\chi_{j,b}^1$ and $\chi_{j,b}^0$ are simply neglected.

The derived soft value is efficiently reformulated in terms of $\mathbf{z}$ and $\boldsymbol{\Upsilon}$, which are readily available upon the termination of the LAS algorithm. Note that $\hat{\mathbf{d}}_j^{b=0}$ and $\hat{\mathbf{d}}_j^{b=1}$ differ only in the j-th entry, and hence

$$\hat{\mathbf{d}}_j^{b=1} = \hat{\mathbf{d}}_j^{b=0} + \gamma_j \mathbf{u}_j \tag{3.56}$$

As $\hat{\mathbf{d}}_j^{b=1}$ and $\hat{\mathbf{d}}_j^{b=0}$ are available, γ_j is obtained from Eq. 3.56. Substituting Eq. 3.56 in Eq. 3.55 yields

$$\begin{aligned} R_{j,b}\|\mathbf{h}_j\|_2^2 &= \|\mathbf{y}' - \mathbf{H}\hat{\mathbf{d}}_j^{b=0} - \gamma_{j,b}\mathbf{h}_i\|_2^2 - \|\mathbf{y} - \mathbf{H}\hat{\mathbf{d}}_j^{b=1}\|_2^2 \\ &= \gamma_j^2\|\mathbf{h}_j\|_2^2 - 2\gamma_{j,b}\mathbf{h}_j^T(\mathbf{y}' - \mathbf{H}\hat{\mathbf{d}}_j^{b=0}) \\ &= -\gamma_j^2\|\mathbf{h}_j\|_2^2 - 2\gamma_{j,b}\mathbf{h}_j^T(\mathbf{y}' - \mathbf{H}\hat{\mathbf{d}}_j^{b=1}) \end{aligned} \tag{3.57}$$

In case of 4-QAM and real-valued system, it holds $\hat{d}_{j,b} = \hat{d}_j$. If $\hat{d}_j = 0$ then $\hat{\mathbf{d}}_{j=0} = \hat{\mathbf{d}}$, and dividing Eq. 3.57 by $\|\mathbf{h}_j\|_2^2$ yields

$$R_j = \gamma_j^2 - 2\gamma_j \frac{z_j}{\upsilon_{jj}} = 4 + 4\frac{z_j}{\upsilon_{jj}} \tag{3.58}$$

If $\hat{d}_j = 1$, then $\hat{\mathbf{d}}_{j=1} = \hat{\mathbf{d}}$, then

$$R_j = -\gamma_j^2 - 2\gamma_j \frac{z_i}{\upsilon_{jj}} = -4 + 4\frac{z_i}{\upsilon_{jj}} \tag{3.59}$$

For proposed MAP LAS, the max-log MAP LLR for each symbol/bit is given as

$$L_j = \min_{\mathbf{d}\in\chi_j^{(-1)}} \left(\frac{\|\mathbf{y}' - \mathbf{H}\mathbf{d}\|_2^2}{\sigma_n^2} - \log P(\mathbf{d}) \right) - \min_{\mathbf{d}\in\chi_j^{(+1)}} \left(\frac{\|\mathbf{y}' - \mathbf{H}\mathbf{d}\|_2^2}{\sigma_n^2} - \log P(\mathbf{d}) \right) \tag{3.60}$$

where $\chi_j^{(-1)}$ and $\chi_j^{(+1)}$ are sets of transmit symbol vectors with symbol at position j being -1 or $+1$.

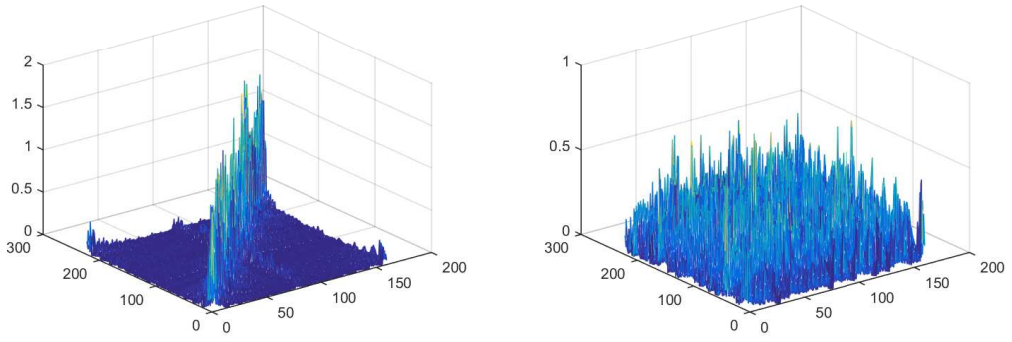
Equation 3.60 requires computation of $2^{2N_d M_t}$ Euclidean distances. In order to reduce the computational complexity, a following approximation is proposed (Eq. 3.61): In case MAP LAS changes the symbol, its hard-decision value is taken as the soft output. In case the symbol is not changed, the LLR provided by LMMSE initial solution is used.

$$\dot{L}_j = \begin{cases} d_j & \text{when } sign(d_j) \neq sign(\bar{L}_j) \\ \bar{L}_j & \text{when } sign(d_j) = sign(\bar{L}_j) \end{cases} \tag{3.61}$$

3.1.3.2 LAS performance improvement using $\check{\mathbf{G}}''_s$

As discussed in Sec. 2.1.2.2, non-systematic generator matrices $\mathbf{G}'$ and $\mathbf{G}''$ exhibit band and full structures respectively. The MIMO version of these matrices share the same structure as depicted in Fig. 3.1. Therefore, it is to expect that MIMO UW-OFDM system with $\check{\mathbf{G}}''$ outperforms the one with $\check{\mathbf{G}}'$ in the uncoded scenario.

In a coded system, the system with $\check{\mathbf{G}}'$ will perform better than $\check{\mathbf{G}}''$. In an uncoded



(A) $\check{\mathbf{G}}'$ (B) $\check{\mathbf{G}}''$

FIGURE 3.1: MIMO UW-OFDM non-systematic generator matrices

system, the BER performance of LAS can be improved when the M_t data streams are coded with different $\mathbf{G}''_m$, yielding a compound generator matrix $\check{\mathbf{G}}''_s$. The LAS algorithm regards the virtual channel matrix $\mathbf{H}_b\check{\mathbf{G}}''$ as a true massive MIMO channel matrix. It has been shown that LAS performance degrades when the channel matrix is correlated [143]. Therefore, for LAS it is important that the entries of the matrix are as much uncorrelated as possible. This is not the case for UW-OFDM, as generator matrix $\mathbf{G}''$ introduces correlations between diagonal blocks of $\mathbf{H}_b$. Therefore, by using generator matrix $\check{\mathbf{G}}''_s$, the introduced correlation in $\mathbf{H}_b\check{\mathbf{G}}''_s$ is less, as this matrix is made of M_t distinct $\mathbf{G}''$ (Eq. 3.5). Hence, it is to expect that using $\check{\mathbf{G}}''_s$ would bring additional LAS performance improvement.

TABLE 3.2: MIMO system parameters

OFDM system		802.11n	UW
FFT size	N	64	64
data subcarriers	N_d	52	40
guard interval/ redundant subcarriers	N_g/ N_r	16	16
redundant sub-carrier indices	$\mathcal{I}_r$	{}	{2, 6, 10, 14, 17, 21, 24, 26, 38, 40, 43, 47, 50, 54, 58, 62}
zero subcarriers	N_z	8	8
zero subcarrier indices	$\mathcal{I}_z$	{0, 29, 30,. . . , 35}	
transmit antennas	M_t	4	4
receive antennas	M_r	4	4

3.1.4 Performance comparison

Simulation of MIMO CP- and UW-OFDM systems is performed in order to obtain BER results. Next, the gains in dB are compared for MIMO CP- and UW-OFDM systems in case of LMMSE and ML detection at BER $= 10^{-6}$.

3.1.4.1 Simulation parameters

Parameters of UW-OFDM system are where possible picked same as in CP-OFDM based 802.11n standard. The system parameters are summarized in Table 3.2.

The recorded 10,000 multipath channel realizations are used in all simulations. The channel coefficients are obtained from standardized 802.11n channel model (model C) [75]. The channel is assumed constant for 100 OFDM symbol vectors. Modulation is 4-QAM with Gray mapping. In case of a coded system, information bits are encoded by (133, 171) convolutional code with constraint length 7 and code rate 1/2 (Sec. 2.1.2.1). At the receiver side, the detected bits are decoded by soft-decision Viterbi decoder implemented in a Matlab routine.

3.1.4.2 Simulation results

The BER results for LMMSE detection are depicted in Fig. 3.2 and 3.3. MIMO UW-OFDM clearly outperforms 802.11n in both uncoded and coded systems. In an uncoded system, using non-systematic generator matrix $\check{\mathbf{G}}''$ provides maximum gain over 802.11n. In a coded system, the advantage of the UW system with generator matrix $\check{\mathbf{G}}'$ over 802.11n is most pronounced. The performance of these generator matrices in coded and uncoded systems coincides with the SISO OFDM case [11]. The system with systematic generator matrix $\check{\mathbf{G}}$ performs worse than the systems with non-systematic generator matrices $\check{\mathbf{G}}'$ and $\check{\mathbf{G}}''$ in both coded and uncoded scenarios.

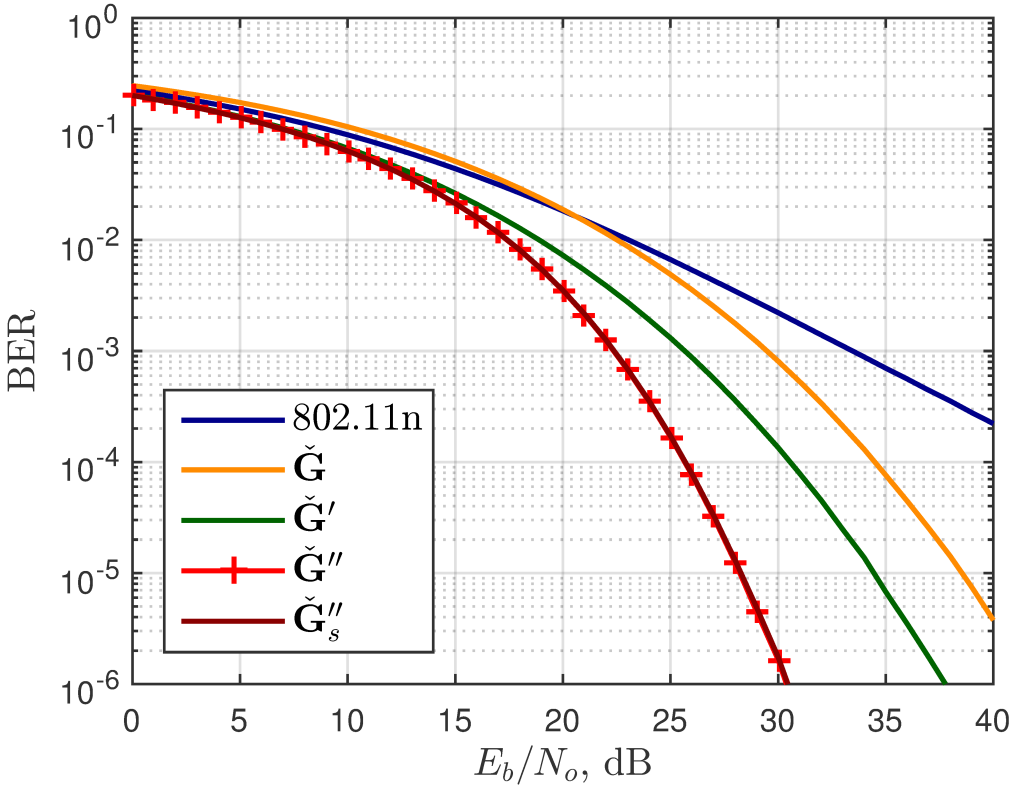


FIGURE 3.2: LMMSE uncoded performance

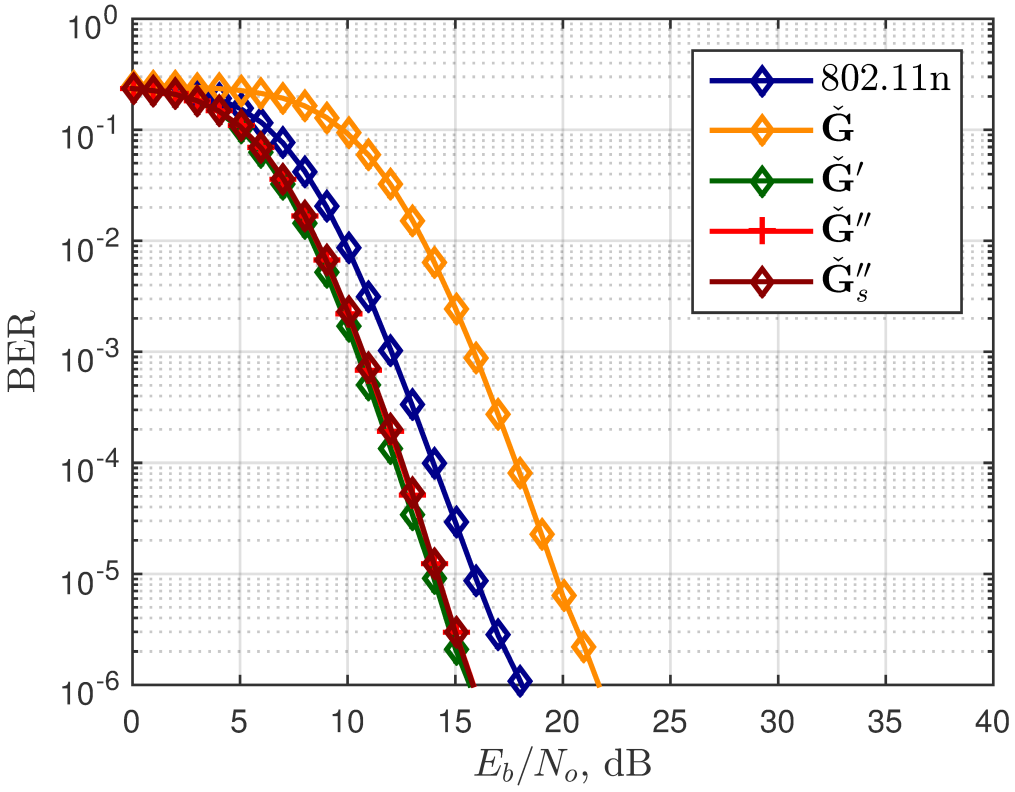


FIGURE 3.3: LMMSE coded performance

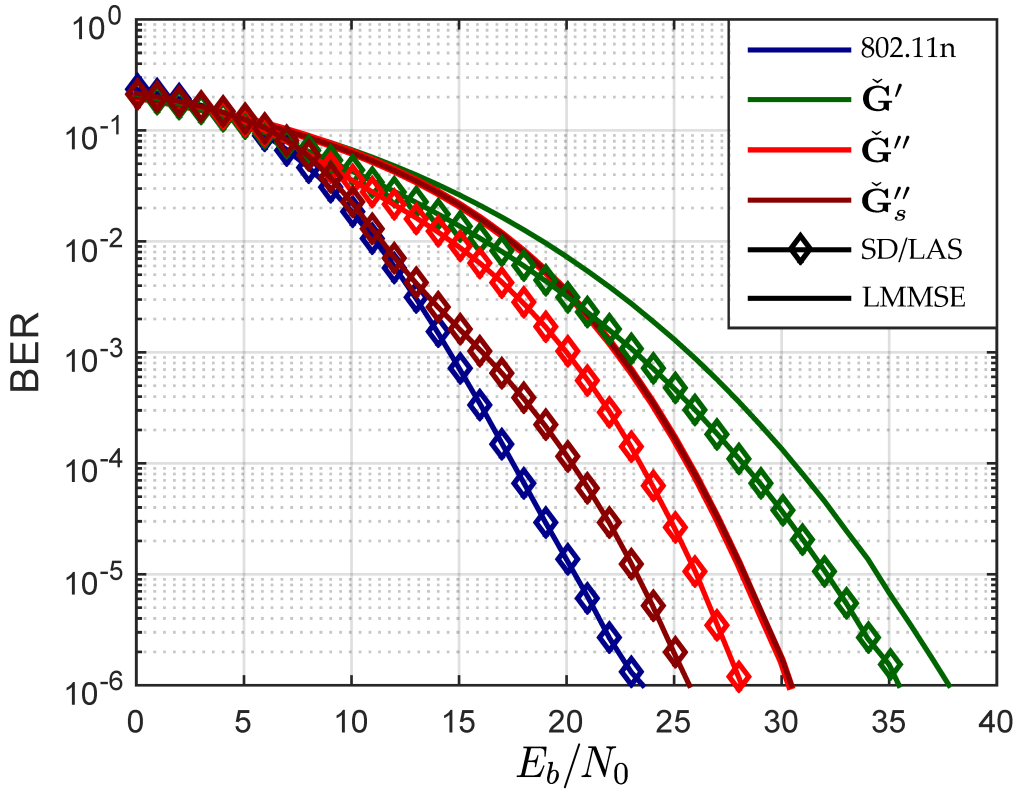


FIGURE 3.4: 1-LAS vs LMMSE, uncoded

Figure 3.4 depicts the gain due to 1-LAS compared to the LMMSE initial solution. It can be observed that UW system with $\check{\mathbf{G}}''$ approaches sphere decoded 802.11n closer than the one with $\check{\mathbf{G}}'$, due to the better performing initial solution. Individual encoding of data streams with $\check{\mathbf{G}}''_s$ further improves 1-LAS performance. It should be noted that the performance of LMMSE initial solutions for $\check{\mathbf{G}}''$ and $\check{\mathbf{G}}''_s$ is exactly the same.

Figure 3.5 depicts performance of three versions of LAS algorithm against sphere decoded 802.11n in an uncoded system. It can be observed that the performance of LAS improves with the number of stages, and with 3-LAS, 802.11n is outperformed by 6 dB at BER of 10^{-6}. Unfortunately, the complexity penalty associated with 3-LAS is substantial. 3-LAS simulation takes two orders of magnitude more time than the SD 802.11n simulation on the same machine. The reason is the size of $S_3(\check{\mathbf{d}})$ neighborhood (Eq. 2.126) that is getting very large for parameters given in Table 3.2.

Figure 3.6 illustrates the problem with original LAS soft output computation for the coded system (Eq. 3.55). In this case, the performance of 1-LAS degrades with respect to the LMMSE solution. As previously mentioned, ignoring LLR provided by LMMSE causes this behavior. The proposed MAP 1-LAS (Eq. 3.61) improves the performance of the initial LMMSE solution.

Finally, Fig. 3.7 depicts performance of LAS using generator matrix $\check{\mathbf{G}}'$ with respect to SD 802.11n performance. It can be observed that, again, performance of LAS improves with the number of stages. However, performance of SD 802.11n cannot be achieved, and there is still a loss of 0.6 dB at BER of 10^{-6} for 3-LAS.

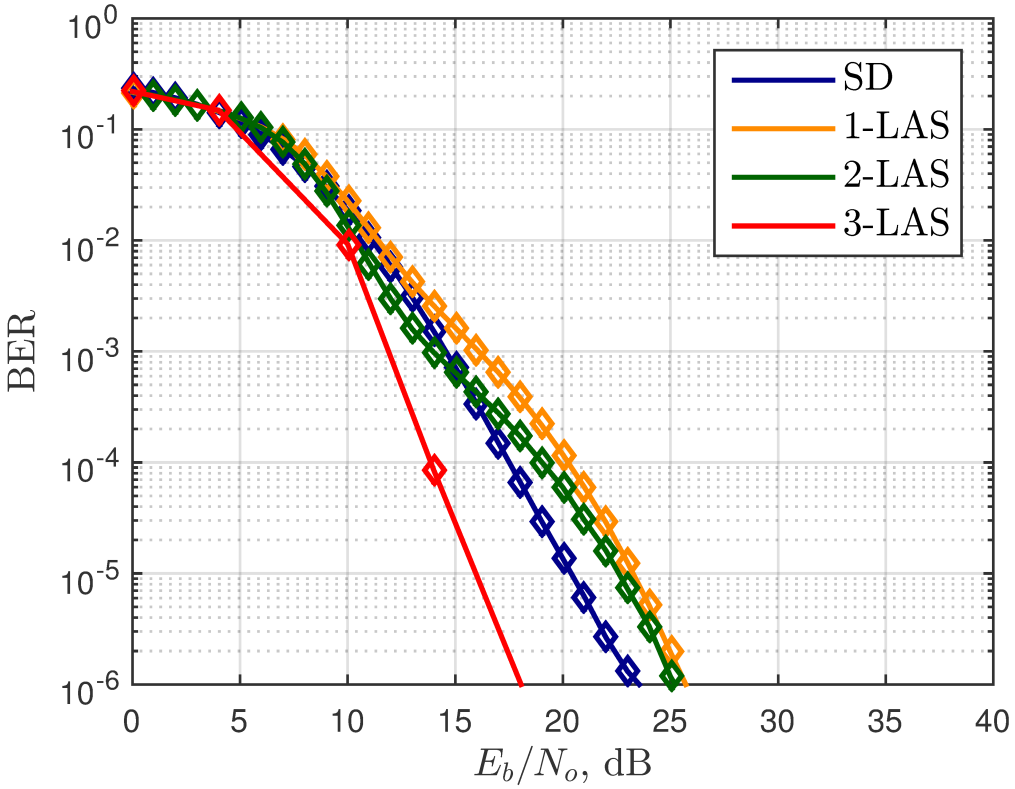


FIGURE 3.5: M-LAS vs SD, $\check{\mathbf{G}}''_s$, uncoded

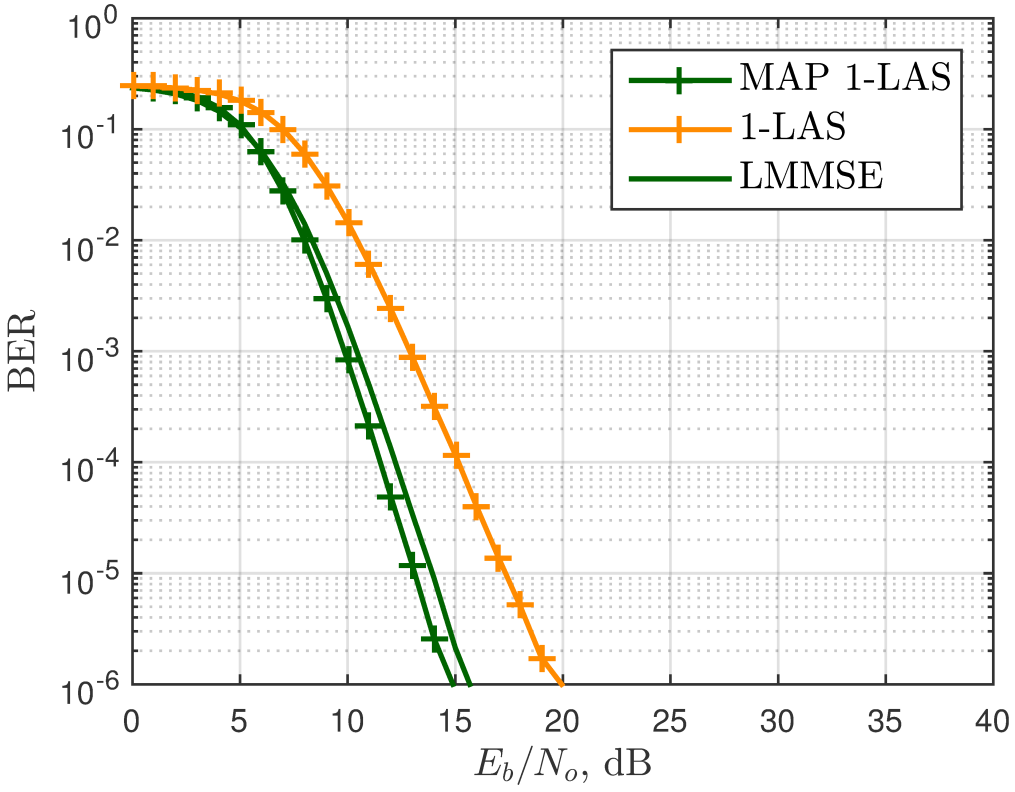


FIGURE 3.6: Proposed MAP 1-LAS vs 1-LAS, $\check{\mathbf{G}}'$, coded

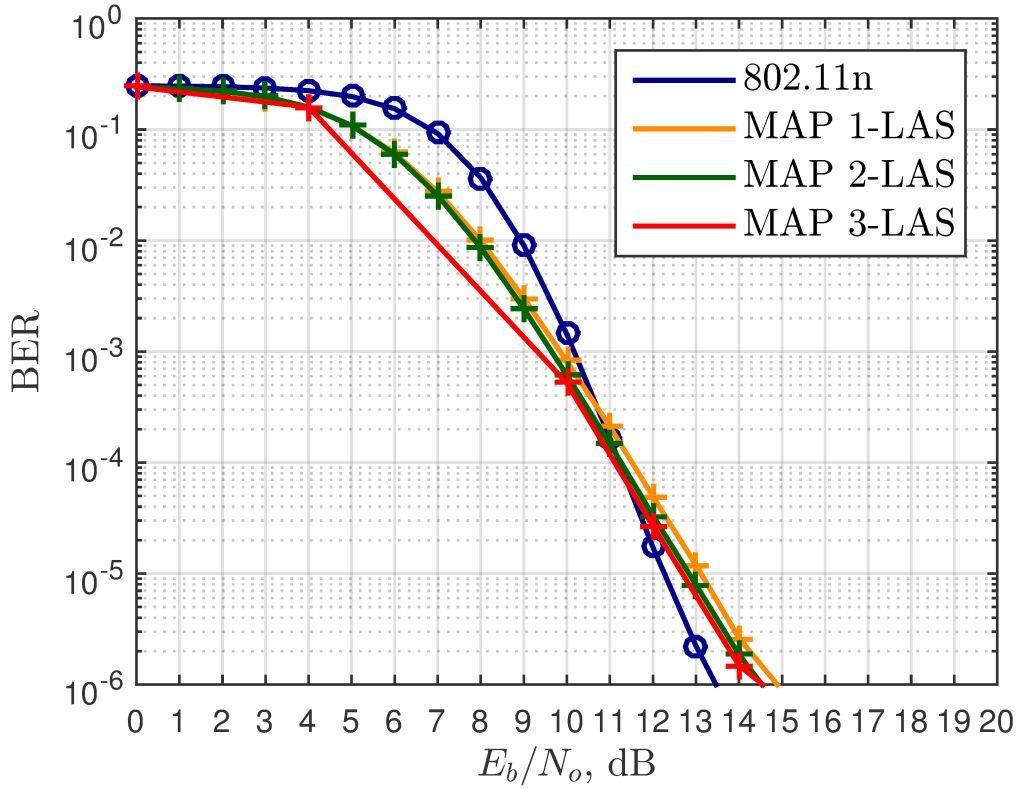


FIGURE 3.7: M-LAS vs SD, $\check{\mathbf{G}}'$, coded

3.1.4.3 Complexity comparison

The complexity of the detection algorithms discussed previously is compared in terms of run-time on the same machine. LMMSE has been implemented as a Matlab routine, whereas nonlinear algorithms are implemented in C connected to Matlab through MEX interface [147]. It could be argued that the C implementation should be faster than the Matlab implementation. However, Matlab uses highly optimized matrix-vector libraries and is able to keep up with C implementation. Therefore, the run-time comparison of linear and non-linear algorithms is considered fair.

The run-time of MIMO CP-OFDM LMMSE detection over the whole range of E_b/N_0 values for one channel realization and 100 receive vectors is used as a unit measure. The run-time of other algorithms is given relative to this measure. The results are summarized in Table 3.3.

TABLE 3.3: Run-time comparison

	MIMO CP-OFDM (802.11n)	MIMO UW-OFDM
LMMSE	1	1.08
SD	2.22	∅
1-LAS	∅	1.3
2-LAS	∅	4.0
3-LAS	∅	309.0

The complexity of LMMSE MIMO-UW OFDM is therefore almost the same as the complexity of LMMSE MIMO CP-OFDM. In order to achieve better than SD performance in uncoded scenario, 3-LAS requires two orders of magnitude more time. In coded scenario, even with this complexity increase, 3-LAS still performs slightly worse than SD. 1-LAS shows little complexity increase relative to LMMSE detected MIMO CP-OFDM, and requires almost twice less time than SD. It also is able to get relatively close to SD in terms of performance in both uncoded and coded scenarios, as shown in simulation results.

3.2 Memory error resilient detection

It has been shown in Sec. 3.1 that the LMMSE detection is attractive for MIMO UW-OFDM, since its complexity grows linearly with the number of antennas. In this section the LMMSE will be modified to take the memory errors discussed in Sec. 2.3.4 into account.

Consider a general complex number MIMO system equation

$$\mathbf{y}_c = \mathbf{H}_c\mathbf{x}_c + \mathbf{n}_c \tag{3.62}$$

The system model in Eq. 3.62 can be decomposed to real-valued numbers

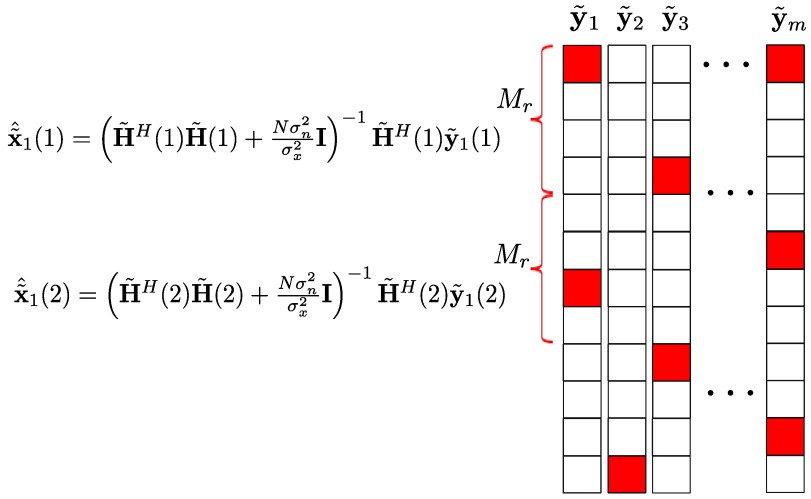
$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{3.63}$$

using the real-value decomposition given in Eq. 3.64

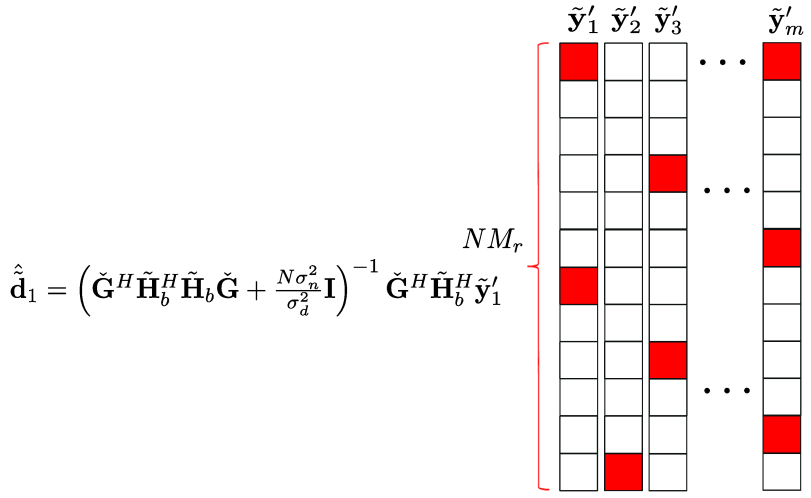
$$\begin{pmatrix} \Re\{\mathbf{y}_c\} \\ \Im\{\mathbf{y}_c\} \end{pmatrix} = \begin{pmatrix} \Re\{\mathbf{H}_c\} & -\Im\{\mathbf{H}_c\} \\ \Im\{\mathbf{H}_c\} & \Re\{\mathbf{H}_c\} \end{pmatrix} \begin{pmatrix} \Re\{\mathbf{x}_c\} \\ \Im\{\mathbf{x}_c\} \end{pmatrix} + \begin{pmatrix} \Re\{\mathbf{n}_c\} \\ \Im\{\mathbf{n}_c\} \end{pmatrix} \tag{3.64}$$

It is to note that this is not a compulsory step. However, now all operations considered further are real-valued, at the cost of doubling the size of vectors and matrices.

The receive symbol vector $\mathbf{y}$ is stored in a buffering memory prior to detection (Fig. 2.32). As the channel is constant for the duration of multiple symbol vectors, the memory can be regarded as a matrix (Fig. 2.33). The number of columns (m) equals the number of symbol vectors, received during the coherence time of the channel. Number of rows equals $M_r \times N$, for both MIMO CP- and UW-OFDM. The memory may be faulty and



(A) Memory addressing in MIMO CP-OFDM



(B) Memory addressing in MIMO UW-OFDM

FIGURE 3.8: Addressing faulty memory in LMMSE

some vector elements may contain single bit flips according to statistical model outlined in Sec. 2.3.4.

As illustrated in Fig. 3.8a, the single-flip single element assumption [126] holds for MIMO CP-OFDM. The linear detection is performed on N distinct systems, where each receive vector is of size $M_r \times 1$.

In MIMO UW-OFDM, the transmit symbol vector estimate is computed for the whole compound receive symbol vector. Therefore, as shown in Fig. 3.8b, multiple errors will affect the estimate computation. Therefore, the single-flip single element assumption does not hold for MIMO UW-OFDM. This assumption must be relaxed and the memory error resilient detection has to handle the single-flip, multiple element scenario.

First, the memory error resilient detection that builds up on the LMMSE estimator will be derived for the single-flip single element case and then extended to handle multiple single bit flips.

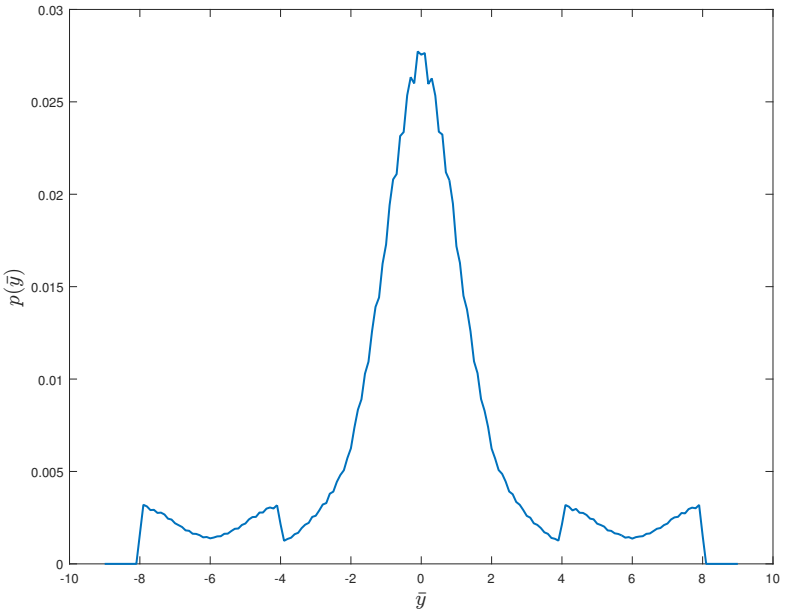


FIGURE 3.9: Pdf of receive vector element

The LMMSE estimate in Eq. 2.95 is optimal in minimum mean square sense when the transmit symbol vector $\mathbf{x}$ and the noise vector $\mathbf{n}$ are independent and Gaussian distributed [83]. Due to memory errors, this assumption does not hold anymore. Figure 3.9 depicts the distribution of an element of the receive symbol vector $\bar{\mathbf{y}}$ stored in the [4].[8] fixed point format. Clearly, this distribution is not Gaussian. The difference to Gaussian bell curve is most prominent at the tails of the depicted distribution. Therefore, in presence of memory errors, the LMMSE estimate is expected to produce a poor result.

It is known that any distribution can be well approximated by a mixture of Gaussian distributions [148, 149]. Hence, the framework of Gaussian mixture (GM) model can be used as a starting point for derivation of a suitable detection scheme that takes memory errors into account. The pdf of a Gaussian mixture distributed random vector $\mathbf{r}$ is given as [150]

$$p(\mathbf{r}) = \sum_{m=1}^{M} P_m N(\mu_{\mathbf{r}}^m, \mathbf{C}_{\mathbf{rr}}^m) \tag{3.65}$$

where $\mu_{\mathbf{r}}^m$ is the mean of m-th component, $\mathbf{C}_{\mathbf{rr}}^m$ is the covariance matrix of m-th component and P_m is the probability of drawing the m-th component from available M components. Obviously, $\sum_{m=1}^{M} P_m = 1$.

At first glance, it makes sense to regard the joint effect of AWGN and memory errors as a GM distributed noise vector $\bar{\mathbf{n}}$, with component covariance matrices $\sigma_{n^m}^2 \mathbf{I}$. As transmit

symbol vector $\mathbf{x}$ and noise vector $\bar{\mathbf{n}}$ are independent, their joint pdf can be written as

$$p(\mathbf{x}, \bar{\mathbf{n}}) = N(\mu_{\mathbf{x}}, \mathbf{C}_{\mathbf{xx}}) \sum_{m=1}^{M} P_m N(\mu_{\bar{\mathbf{n}}}^m, \mathbf{C}_{\bar{\mathbf{n}}\bar{\mathbf{n}}}^m) \tag{3.66}$$

$$= \sum_{m=1}^{M} P_m N(\mu^m, \mathbf{C}^m) \tag{3.67}$$

Thus, the joint pdf of $\mathbf{x}$ and $\bar{\mathbf{n}}$ is GM distributed as shown in Eq. 3.67 [151], with

$$\mu^m = \begin{bmatrix} \mu_{\mathbf{x}} \\ \mu_{\bar{\mathbf{n}}}^m \end{bmatrix}, \mathbf{C}^m = \begin{bmatrix} \mathbf{C}_{\mathbf{xx}} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\mathbf{n}\bar{\mathbf{n}}}^m \end{bmatrix} \tag{3.68}$$

A following relation can be derived from the system equation [150]:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{H}\mathbf{x} + \bar{\mathbf{n}} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{H} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \bar{\mathbf{n}} \end{bmatrix} = \bar{\mathbf{H}} \begin{bmatrix} \mathbf{x} \\ \bar{\mathbf{n}} \end{bmatrix} \tag{3.69}$$

Equation 3.69 shows that $\begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix}$ is related to $\mathbf{x}$ through a linear transformation. This implies that if $\mathbf{x}$ and $\bar{\mathbf{n}}$ are jointly GM distributed, then so are $\mathbf{x}$ and $\mathbf{y}$ [83]. Therefore, the joint distribution of $\mathbf{x}$ and $\mathbf{y}$ is

$$p(\mathbf{y}, \mathbf{x}) = \sum_{m=1}^{M} P_m N(\bar{\mathbf{H}}\mu^m, \bar{\mathbf{H}}\mathbf{C}^m\bar{\mathbf{H}}^T) \tag{3.70}$$

where $\bar{\mathbf{H}}\mu^m$ is given as

$$\bar{\mathbf{H}}\mu^m = \begin{bmatrix} \mu_{\mathbf{y}}^m \\ \mu_{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{H}\mu_{\mathbf{x}} + \mu_{\bar{\mathbf{n}}}^m \\ \mu_{\mathbf{x}} \end{bmatrix} \tag{3.71}$$

and

$$\bar{\mathbf{H}}\mathbf{C}^m\bar{\mathbf{H}}^T = \begin{bmatrix} \mathbf{C}_{\mathbf{yy}}^m & \mathbf{C}_{\mathbf{yx}}^m \\ \mathbf{C}_{\mathbf{xy}}^m & \mathbf{C}_{\mathbf{xx}}^m \end{bmatrix} = \begin{bmatrix} \mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}_{\bar{\mathbf{n}}\bar{\mathbf{n}}}^m & \mathbf{H}\mathbf{C}_{\mathbf{xx}} \\ \mathbf{C}_{\mathbf{xx}}\mathbf{H}^T & \mathbf{C}_{\mathbf{xx}} \end{bmatrix} \tag{3.72}$$

Since the joint distribution is available, it is possible to derive the conditional mean $E\{\mathbf{x}|\mathbf{y}\}$ which is the MMSE estimator [83]. The marginal pdf of $\mathbf{y}$ is [152]

$$p(\mathbf{y}) = \sum_{m=1}^{M} P_m p^m(\mathbf{y}) \tag{3.73}$$

where $p^m(\mathbf{y})$ is Gaussian with mean $\mu_{\mathbf{y}}^m$ and covariance matrix $\mathbf{C}_{\mathbf{yy}}^m$.

Joint distribution and conditional distribution are connected by Bayes' rule [83]:

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{y})} &= \frac{\sum_{m=1}^{M} P_m p^m(\mathbf{y},\mathbf{x})}{\sum_{m=1}^{M} P_m p^m(\mathbf{y})} \\ &= \frac{\sum_{m=1}^{M} P_m p^m(\mathbf{y}) p^m(\mathbf{x}|\mathbf{y})}{\sum_{m=1}^{M} P_m p^m(\mathbf{y})} \\ &= \sum_{m=1}^{M} \alpha_m(\mathbf{y}) p^m(\mathbf{x}|\mathbf{y}) \end{aligned} \tag{3.74}$$

where

$$\begin{aligned} \alpha_m(\mathbf{y}) &= \frac{P_m p^m(\mathbf{y})}{\sum_{m=1}^{M} P_m p^m(\mathbf{y})} \\ &= \frac{P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}^m_{\bar{\mathbf{n}}\bar{\mathbf{n}}}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{y}-(\mathbf{H}\mu_{\mathbf{x}}+\mu^m_{\bar{\mathbf{n}}}))^T (\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}^m_{\bar{\mathbf{n}}\bar{\mathbf{n}}})^{-1} (\mathbf{y}-(\mathbf{H}\mu_{\mathbf{x}}+\mu^m_{\bar{\mathbf{n}}}))}}{\sum_{m=1}^{M} P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}^m_{\bar{\mathbf{n}}\bar{\mathbf{n}}}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{y}-(\mathbf{H}\mu_{\mathbf{x}}+\mu^m_{\bar{\mathbf{n}}}))^T (\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}^m_{\bar{\mathbf{n}}\bar{\mathbf{n}}})^{-1} (\mathbf{y}-(\mathbf{H}\mu_{\mathbf{x}}+\mu^m_{\bar{\mathbf{n}}}))}} \end{aligned} \tag{3.75}$$

The coefficient $\alpha_m(\mathbf{y})$ can be regarded as the probability that $\bar{\mathbf{n}}$ is drawn from component pdf m, given the receive vector $\mathbf{y}$. Obtaining the mean of the conditional distribution $p(\mathbf{x}|\mathbf{y})$:

$$\begin{aligned} E\{\mathbf{x}|\mathbf{y}\} &= \int \mathbf{x} p(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\ &= \int \mathbf{x} \sum_{m=1}^{M} \alpha_m(\mathbf{y}) p^m(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\ &= \sum_{m=1}^{M} \alpha_m(\mathbf{y}) \int \mathbf{x} p^m(\mathbf{x}|\mathbf{y}) d\mathbf{x} \end{aligned} \tag{3.76}$$

As $p^m(\mathbf{y}|\mathbf{x})$ is conditional density of joint Gaussian $p^m(\mathbf{y},\mathbf{x})$, $p^m(\mathbf{y}|\mathbf{x})$ is also Gaussian. The integrals of the individual m components are then the LMMSE estimators of that particular m-th component as given in Eq. 2.95.

Therefore, the MMSE estimate for a Gaussian mixture is given as

$$\hat{\mathbf{x}} = \sum_{m=1}^{M} \alpha_m(\mathbf{y}) (\mu_{\mathbf{x}} + \mathbf{C}_{\mathbf{xx}}\mathbf{H}^T (\mathbf{H}\mathbf{C}_{\mathbf{xx}}\mathbf{H}^T + \mathbf{C}^m_{\bar{\mathbf{n}}\bar{\mathbf{n}}})^{-1} (\mathbf{y} - (\mathbf{H}\mu_{\mathbf{x}} + \mu^m_{\bar{\mathbf{n}}})) \tag{3.77}$$

The MMSE estimator for a GM is nonlinear, as the coefficients $\alpha_m(\mathbf{y})$ are non-linear in $\mathbf{y}$. Taking into account that transmit symbol vector and the m noise components are zero mean with $\mathbf{C}_{\mathbf{xx}} = \sigma_x^2\mathbf{I}$ and $\mathbf{C}_{\bar{\mathbf{n}}\bar{\mathbf{n}}} = \sigma_{n^m}^2\mathbf{I}$ the nonlinear MMSE (NMMSE) estimate

is given in a more convenient form as

$$\hat{\mathbf{x}} = \sum_{m=1}^{M} \alpha_m(\mathbf{y}) \sigma_x^2 \mathbf{I}\mathbf{H}^T (\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_{n^m}^2 \mathbf{I})^{-1} \mathbf{y} \tag{3.78}$$

with coefficients $\alpha_m(\mathbf{y})$

$$\alpha_m(\mathbf{y}) = \frac{P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_{n^m}^2 \mathbf{I}|^{\frac{1}{2}}} e^{-w_m}}{\sum_{m=1}^{M} P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_{n^m}^2 \mathbf{I}|^{\frac{1}{2}}} e^{-w_m}} \tag{3.79}$$

where the exponents w_m are given as

$$w_m = \frac{1}{2} \mathbf{y}^T (\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_{n^m}^2 \mathbf{I})^{-1} \mathbf{y} \tag{3.80}$$

It is to observe that if there is only one noise component, $M = 1$, the coefficient $\alpha_1(\mathbf{y})$ equals one and the MMSE estimate coincides with the LMMSE estimate in Eq. 2.95.

3.2.1 NMMSE-based single error correction

The general NMMSE estimator is next adapted to mitigate the effect of memory errors. First, the attention is restricted to single bit flips within the integer part, as they cause the largest error values. In this case the mixture component probabilities P_m are: the probability to have no memory errors $P(k = 0)$ and d equal probabilities $P(k = 1)$ to have a single bit flip at one of of the integer bit positions j, $j \in \{l-1, \ldots, f\}$. Therefore, there would be $d + 1$ noise components: nominal channel AWGN noise $\mathbf{n}$ in case there are no memory errors and d components where each noise term includes the bit flip at bit position j, $j \in \{l - 1, \ldots, f\}$.

The problem with this approach is that according to memory error model in Sec. 2.3.4, the value of e_i^j actually depends on the value of y_i^j. Thus, it is not possible to separate out e_i^j from $\bar{y}_i$ and form $\bar{n}_i = n_i + e_i^j$. Therefore, the same channel AWGN noise vector with covariance matrix $\sigma_n^2 \mathbf{I}$ is present in all $d + 1$ mixture components. The first component assumes receive vector to be memory error free ($\bar{\mathbf{y}} = \mathbf{y}$), while remaining components assume that some elements of receive vector are affected by bit flips at position j,

$$\bar{\mathbf{y}}^j = \mathbf{y} + e^j \mathbf{u} \tag{3.81}$$

where e^j are the error values given in Eq. 2.144. Vector $\mathbf{u}$ contains ones at i-th position for which the bit flip at bit position j occurs and zero otherwise. First, the assumption of having single bit flip in just a single element i of the receive vector taken in [6] is considered valid. Therefore, the receive vector $\bar{\mathbf{y}}$ may contain single element i affected by a single bit flip at position j, $j \in \{l - 1, \ldots, f\}$. This implies that vector $\mathbf{u}$ is a unit vector.

Finally the mixture contains: receive vector $\mathbf{y}$ with no memory errors with component probability $P(k = 0)$ and d components where receive vector $\bar{\mathbf{y}}^j$ has a bit flip at position j, $j \in \{l - 1, \ldots, f\}$ at some element i with same component probability $P(k = 1)$. Since possible error values are known, the position i of error-affected vector element can

be identified. When an element of the receive vector contains a flip in the integer part, the pdf of $\bar{\mathbf{y}}$ will deviate from the nominal multivariate Gaussian distribution given in the nominator of the NMMSE coefficient $\alpha_{(m=1)}(\bar{\mathbf{y}})$ in Eq. 3.79. The exponent $w_{(m=1)}$ given in Eq. 3.80 will assume a large value and the coefficient $\alpha_1(\bar{\mathbf{y}})$ will be close to zero.

The receiver has the potentially error-affected vector at its disposal, but does not know whether the error has occurred, nor the affected element of the vector. Therefore, for each possible bit flip position j (corresponding to mixture components, $m > 1$), the single bit flip correction values β_i^j are computed for all i elements of the receive vector. Since the value of the assumed bit flip depends on the value of the receive vector, the correction values are obtained as

$$\beta_i^j = \begin{cases} (2\bar{y}_i^j - 1)2^{(j-f)} & \text{when } j = l-1 \\ (-2\bar{y}_i^j + 1)2^{(j-f)} & \text{when } j \neq l-1 \end{cases} \tag{3.82}$$

The exponents w_i^j are obtained with applied single-bit corrections

$$w_i^j = \frac{1}{2}(\bar{\mathbf{y}} + \beta_i^j \mathbf{u}_i)^T (\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_n^2 \mathbf{I})^{-1} (\bar{\mathbf{y}} + \beta_i^j \mathbf{u}_i) \tag{3.83}$$

where $\mathbf{u}_i$ is the unit vector containing one at position i and zeros at other positions. When the applied single-bit correction indeed corrects the error, the deviation from the nominal Gaussian density would decrease and the respective w_i^j will decrease too. Otherwise, the correction actually introduces an error, forcing respective w_i^j to increase. By finding the minimal exponent w_i^j for a given $j, j \in \{l-1, \ldots, f\}$, the index i of the receive vector element where the bit flip at position j occurred is identified.

Next, the coefficients $\alpha_m(\bar{\mathbf{y}})$ are obtained with minimal exponent values. It is to note that as the division by the sum of nominators in Eq. 3.79 is just scaling, the nominator of coefficient and the coefficient itself will be used interchangeably.

$$\alpha_m(\mathbf{y}) = P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\mathbf{H}\sigma_x^2 \mathbf{I}\mathbf{H}^T + \sigma_n^2 \mathbf{I}|^{\frac{1}{2}}} e^{-w_{min}^j} \tag{3.84}$$

Finally, the NMMSE estimate is computed by Eq. 3.78 with component coefficients computed previously. The single error correction algorithm is summarized in Alg. 2 for MIMO CP-OFDM. Real-valued frequency domain vectors and matrices are assumed. Its detailed operation is outlined next:

- The algorithm is executed sequentially for all N sub-carriers.
- The bit values $\bar{\bar{y}}_i^j$ of the integer part of all elements $\bar{\bar{y}}_i$ of receive vector $\bar{\bar{y}}$ are extracted by BITGET function in line 5 of Alg. 2.
- For each integer part bit position j of all elements i of receive vector $\bar{\bar{\mathbf{y}}}$, the correction values are computed by Eq. 3.82, assuming that a single-flip occurred at exactly this position.
- For each integer part bit position j and all elements i of $\bar{\bar{\mathbf{y}}}$, compute exponents w_i^j by Eq. 3.83 with correction values applied.

Algorithm 2 NMMSE-based single error correction for MIMO CP-OFDM

for $k = 1, \dots, N$ **do**
  for $j = f, \dots, l-1$ **do**
    for $i = 1, \dots, 2M_r$ **do**
      $\bar{\tilde{y}}_i^j(k) = \text{BITGET}(\bar{\tilde{\mathbf{y}}}(k), j)$
    end for
  end for
  for $j = f, \dots, l-1$ **do**
    for $i = 1, \dots, 2M_r$ **do**
      if $j = l-1$ **then**
        $\beta_i^j = (2\bar{\tilde{y}}_i^j(k) - 1)2^{(j-f)}$
      else
        $\beta_i^j = (-2\bar{\tilde{y}}_i^j(k) + 1)2^{(j-f)}$
      end if
    end for
  end for
  for $j = f, \dots, l-1$ **do**
    for $i = 1, \dots, 2M_r$ **do**
      $w_i^j = \frac{1}{2}(\bar{\tilde{\mathbf{y}}}(k) + \beta_i^j \mathbf{u}_i)^T (\tilde{\mathbf{H}}(k)\sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2 \mathbf{I})^{-1} (\bar{\tilde{\mathbf{y}}}(k) + \beta_i^j \mathbf{u}_i)$
    end for
  end for
  for all j **do**
    $w_{min}^j = \min_i w_i^j$
  end for
  for all m **do**
    $$\alpha_m(\bar{\tilde{\mathbf{y}}}(k)) = \frac{P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\tilde{\mathbf{H}}(k)\sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2 \mathbf{I}|^{\frac{1}{2}}} e^{-w_{min}^j}}{\sum_{m=1}^{(d+1)} P_m \frac{1}{(\sqrt{2\pi})^{2M_r} |\tilde{\mathbf{H}}(k)\sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2 \mathbf{I}|^{\frac{1}{2}}} e^{-w_{min}^j}}$$
  end for
  for $j = f, \dots, l-1$ **do**
    $w_{min}^j \to \beta_{s_m}^j$
  end for
  $$\hat{\tilde{\mathbf{x}}}(k) = \sum_{m=1}^{d+1} \alpha_m(\bar{\tilde{\mathbf{y}}}(k)) \sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) (\tilde{\mathbf{H}}(k)\sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2 \mathbf{I})^{-1} (\bar{\tilde{\mathbf{y}}}(k) + \beta_{s_m}^j \mathbf{u}_{s_m})$$
end for

- For each integer bit position j, obtain minimum exponent w_{min}^j. Recall that the j-th bit flip position corresponds to the mixture component $m > 1$. For $m = 1$, minimization is not required, since $\bar{\mathbf{y}}$ is assumed memory error free and $w_{(m=1)}$ is obtained by Eq. 3.80.
- Compute component coefficients $\alpha_m(\bar{\tilde{\mathbf{y}}})$ with obtained w_{min}^j and $w_{(m=1)}$ by Eq. 3.84.
- Finally, the NMMSE estimate is computed by Eq. 3.78 with correction values $\beta_{s_m}^j$ applied, where s_m are the indexes of vector elements for which minimum exponent values w_{min}^j were obtained.

The inverse of $\tilde{\mathbf{H}}(k)\sigma_x^2 \mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2 \mathbf{I}$ and its determinant can be precomputed for all N sub-carriers for the duration when the channel matrix $\tilde{\mathbf{H}}_b$ is constant. The correction values β_i^j can be stored in a look-up table, because their values are predefined for a specific

fixed point format. The remaining operation are simply matrix-vector multiplications, except the exponentiation in Eq. 3.79.

It has been found by numerical simulation that the optimal fixed point format for MIMO CP-OFDM is [4].[8]. The component probabilities are

$$P_1 = P(k=0) = (1-P_e)^4$$
$$P_{2\ldots5} = P(k=1) = P_e(1-P_e)^3$$

For example, if $j = l-1 = 11$ and $\bar{\tilde{y}}_i^j(k) = 1$, the assumed correct value of the sign bit is $\tilde{y}_i^j(k) = 0$. This implies that the bit flip in the sign bit from 0 to 1 has happened, and the error value is $-2^{(j-f)} = -2^3$. Therefore, the correction value is computed as $+2^3$. In case the error has really occurred at this bit position of this receive vector element, the corresponding w_i^j will be the minimum for this j and the corrections applied to j-th bit position of other elements only introduce the error.

The NMMSE estimator will be

$$\begin{aligned}\hat{\tilde{\mathbf{x}}}(k) =& \alpha_1(\bar{\tilde{\mathbf{y}}}(k))\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)+\sigma_n^2\mathbf{I})^{-1}\bar{\tilde{\mathbf{y}}}(k)+\\ &\alpha_2(\bar{\tilde{\mathbf{y}}}(k))\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)+\sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k)+\beta_{s_2}^{11}\mathbf{u}_{s_2})+\\ &\alpha_3(\bar{\tilde{\mathbf{y}}}(k))\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)+\sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k)+\beta_{s_3}^{10}\mathbf{u}_{s_3})+\\ &\alpha_4(\bar{\tilde{\mathbf{y}}}(k))\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)+\sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k)+\beta_{s_4}^{9}\mathbf{u}_{s_4})+\\ &\alpha_5(\bar{\tilde{\mathbf{y}}}(k))\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)+\sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k)+\beta_{s_5}^{8}\mathbf{u}_{s_5})\end{aligned} \tag{3.85}$$

where $\mathbf{u}_{s_m}$ is the unit vector with one at position s_m and zero at all other positions. The exponent values $w_i^{(j\neq11)}$ for the bit positions other than $j = 11$ will be all very large, due to the single-flip single element assumption. The corrections introduced for these bit positions will only introduce the actual error. Therefore, the coefficients $\alpha_m(\bar{\mathbf{y}})$ will for $j \neq 11$ will all be near zero, effectively leaving only the LMMSE estimate with corrected receive symbol vector $(\bar{\mathbf{y}}(k) + \beta_{s_2}^{11}\mathbf{u}_{s_2})$ in Eq. 3.85.

3.2.1.1 log-NMMSE

It is convenient to transform the computation of coefficients to log domain:

$$\begin{aligned}\alpha_m(\bar{\mathbf{y}}) &= \log\left(P_m\frac{1}{(\sqrt{2\pi})^{2M_r}|\mathbf{H}\sigma_x^2\mathbf{I}\mathbf{H}^T+\sigma_n^2\mathbf{I}|^{\frac{1}{2}}}e^{-w_{min}^j}\right)\\ &= \log(P_m) - \log\left((\sqrt{2\pi})^{2M_r}|\mathbf{H}\sigma_x^2\mathbf{I}\mathbf{H}^T+\sigma_n^2\mathbf{I}|^{\frac{1}{2}}\right) - w_{min}^j\end{aligned} \tag{3.86}$$

The first two terms have to be computed once for the particular channel matrix $\mathbf{H}$.

Further, it is not necessary to compute NMMSE estimate as given in Eq. 3.85. Instead, maximum coefficient $\alpha_{max}(\bar{\mathbf{y}})$ is found:

$$\alpha_{max}(\bar{\mathbf{y}}) = \max_m \alpha_m(\bar{\mathbf{y}}) \tag{3.87}$$

Algorithm 3 log-NMMSE for MIMO CP-OFDM

for $k = 1, \ldots, N$ **do**
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2M_r$ **do**
      $\bar{\tilde{y}}_i^j(k) = \text{BITGET}(\bar{\tilde{\mathbf{y}}}(k), j)$
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2M_r$ **do**
      if $j = l-1$ **then**
        $\beta_i^j = (2\bar{\tilde{y}}_i^j(k) - 1)2^{(j-f)}$
      else
        $\beta_i^j = (-2\bar{\tilde{y}}(k)_i^j + 1)2^{(j-f)}$
      end if
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2M_r$ **do**
      $w_i^j = \frac{1}{2}(\bar{\tilde{\mathbf{y}}}(k) + \beta_i^j \mathbf{u}_i)^T(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k) + \beta_i^j \mathbf{u}_i)$
    end for
  end for
  for all j **do**
    $w_{min}^j = \min_i w_i^j$
  end for
  for all m **do**
    $\alpha_m(\bar{\tilde{\mathbf{y}}}(k)) = \log(P_m) - \log\left((\sqrt{2\pi})^{2M_r}|\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2\mathbf{I}|^{\frac{1}{2}}\right) - w_{min}^j$
  end for
  $\alpha_{max}(\bar{\tilde{\mathbf{y}}}(k)) = \max_m \alpha_m(\bar{\mathbf{y}}(k))$
  $\alpha_m^{max}(\bar{\tilde{\mathbf{y}}}(k)) \rightarrow \beta_s^z$
  $\hat{\tilde{x}}(k) = \sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k)(\tilde{\mathbf{H}}(k)\sigma_x^2\mathbf{I}\tilde{\mathbf{H}}^T(k) + \sigma_n^2\mathbf{I})^{-1}(\bar{\tilde{\mathbf{y}}}(k) + \beta_s^z \mathbf{u}_s)$
end for

The maximum coefficient identifies the single error that caused the maximal deviation from the nominal Gaussian pdf. The value of this error is given by the maximal coefficient index m that corresponds to the bit flip position j. The vector element i where this error has occurred is given by the index for which the minimal exponent was obtained for the maximal coefficient.

The LMMSE estimate corresponding to $\alpha_{max}(\bar{\mathbf{y}})$ is declared the final solution.

$$\hat{\mathbf{x}} = \sigma_x^2\mathbf{I}\mathbf{H}^T(\mathbf{H}\sigma_x^2\mathbf{I}\mathbf{H}^T + \sigma_n^2\mathbf{I})^{-1}(\bar{\mathbf{y}} + \beta_s^z\mathbf{u}) \tag{3.88}$$

where β_s^z is the correction value corresponding to index s of vector element for which minimum exponent value w_{min}^j was obtained for $\alpha_{max}(\bar{\mathbf{y}})$.

The proposed log-NMMSE for MIMO CP-OFDM is outlined in Alg. 3, where the lines different to Alg. 2 are highlighted, while the unchanged lines are shaded gray.

3.2.2 Iterative NMMSE-based multiple error correction

In case of MIMO UW-OFDM, or in massive MIMO in general, the assumption of single error within the receive symbol vector does not hold. The generalization of the algorithms developed in Sec. 3.2.1 to the multiple error scenario is performed next.

Again, the mixture is assumed to contain $d+1$ components, where the first component assumes receive vector to be memory error free, while the remaining components assume that some elements of receive vector are affected by bit flips at position j, as given by Eq. 3.81. Vector $\mathbf{u}$ contains ones at i-th position for which the bit flip at bit position j occurs and zero otherwise. The simplifying assumption of having single bit flip in just a single element i of the receive vector taken in [6] cannot be fulfilled in MIMO UW-OFDM due to large memory size required. Therefore, the receive vector $\bar{\mathbf{y}}$ may contain *multiple elements* i affected by a single bit flip at position j, $j \in \{l-1, \ldots, f\}$. This implies that vector $\mathbf{u}$ may contain *multiple* ones.

Similarly, the mixture contains: receive vector $\bar{\mathbf{y}}$ with no memory errors with component probability $P(k=0)$ and d components where receive vector $\bar{\mathbf{y}}^j$ has a bit flip at position j, $j \in \{l-1, \ldots, f\}$ at some elements i with same component probability $P(k=1)$. Since possible error values are known, the positions i of error-affected vector elements can be identified *one by one*. When an element of the receive vector contains a flip in the integer part, the pdf of $\bar{\mathbf{y}}$ will deviate from the nominal multivariate Gaussian distribution given in the nominator of the NMMSE coefficient $\alpha_{(m=1)}(\bar{\mathbf{y}})$ in Eq. 3.79. The exponent $w_{(m=1)}$ given in Eq. 3.80 will assume a large value and the coefficient $\alpha_1(\bar{\mathbf{y}})$ will be close to zero.

The receiver has the potentially error-affected vector at its disposal, but does not know whether or *how many error(s)* have occurred, nor the affected *element(s)* of the vector. Therefore, for each possible bit flip position j (corresponding to mixture components, $m > 1$), the single-bit flip correction values β_i^j are computed for all i elements of the receive vector. Since the value of the assumed bit flip depends on the value of the receive vector, the correction values are obtained by Eq. 3.82.

The exponents w_i^j are obtained with applied single-bit corrections by Eq. 3.83. When the applied single-bit correction indeed corrects the error, the deviation from the nominal Gaussian density would decrease and the respective w_i^j will decrease too. Otherwise, the correction actually introduces an error, forcing respective w_i^j to increase. By finding the minimal exponent w_i^j for a given $j, j \in \{l-1, \ldots, f\}$, the index i of the receive vector element where the bit flip at position j occurred is identified.

Next, the coefficients $\alpha_m(\bar{\mathbf{y}})$ with minimal exponent values are obtained. The direct computation of coefficients by Eq. 3.79 reveals a numeric instability that is caused by the multiple errors within the receive vector. Assume that $\bar{\mathbf{y}}$ contains two bit flips: one at $(l-1)$-th bit position in element s and the other at $(l-2)$-th bit position in element z. The minimum exponent value for $(l-1)$-th bit flip position will be obtained with the bit flip corrected, however, due to the remaining bit flip in element z, the value of $w_{min}^{(l-1)}$ may still be large. This would result in the zero coefficient for $(l-1)$-th bit flip position. The same would happen for $w_{min}^{(l-2)}$, due to influence of bit flip in element s. This way all coefficients would be computed to zero, rendering the algorithm useless.

The direct way to overcome this is to compute the minimum of the w_{min}^{j} instead of computing the coefficients:

$$w_{min} = \min_{m} w_{min}^{j} \tag{3.89}$$

The index of w_{min} would then identify the bit flip value and the position of the single-flip value that caused the largest deviation from the nominal Gaussian pdf. The problem with this direct approach is that the component probabilities P_m are not taken into account. Therefore, the performance of this version of the iterative algorithm will be suboptimal, as the simulation results in Sec. 3.2.4 will indicate.

The better way to avoid the numeric issue due to multiple errors is to compute the coefficients in log domain by Eq. 3.86. In this case, the maximum coefficient identifies the single error that caused the maximal deviation from the nominal Gaussian pdf. The value of this error is given by the maximal coefficient index m that corresponds to the bit flip position j. The vector element i where this error has occurred is given by the index for which the minimal exponent was obtained for the maximal coefficient.

This way the correction is performed iteratively starting from the memory error which caused the largest deviation and correcting one error per iteration. The iterative algorithm with error position identification by Eq. 3.89 is summarized in Alg. 4 and the iterative algorithm with coefficient computation in log domain is summarized in Alg. 5 for MIMO UW-OFDM. Real-valued frequency domain vectors and matrices are assumed. The function and respective differences of the two algorithms are highlighted next.

- the initial receive vector $\bar{\tilde{\mathbf{y}}}'$ is read from the buffering memory.
- Next, the main loop is executed until maximum number of iterations m_{it} is reached.
- Within an iteration, first the bit values $\bar{\tilde{y}}_i'^{j}$ of the integer part of all elements $\bar{\tilde{y}}_i'$ of receive vector $\bar{\tilde{\mathbf{y}}}'$ are extracted by BITGET function in line 5 of Alg. 4, Alg. 5.
- For each integer part bit position j of all elements i of receive vector $\bar{\tilde{\mathbf{y}}}'$, the correction values are computed by Eq. 3.82, assuming that a single flip occurred at exactly this position.
- For each integer part bit position j and all elements i of $\bar{\tilde{\mathbf{y}}}$, exponents w_i^j by Eq. 3.83 with correction values applied are computed.
- For each integer bit position j, minimum exponent is obtained. Recall that the j-th bit flip position corresponds to the mixture component $m > 1$. For $m = 1$, minimization is not required, since $\bar{\tilde{\mathbf{y}}}'$ is assumed memory error free and $w_{(m=1)}$ is obtained by Eq. 3.80.
- In Alg. 4 compute w_{min} by Eq. 3.89. In Alg. 5 compute component coefficients $\alpha_m(\bar{\tilde{\mathbf{y}}}')$ with obtained w_{min}^{j} and $w_{(m=1)}$ in log domain by Eq. 3.86.
- In Alg. 4, if w_{min} corresponds to $m = 1$, the algorithm is terminated as all errors are assumed to have been corrected and LMMSE estimate of the current $\bar{\tilde{\mathbf{y}}}'^{(it)}$ is output (Eq. 3.90). In Alg. 5 the maximum component coefficient is obtained. If $\alpha_{max}(\bar{\mathbf{y}})$

corresponds to $m = 1$, the algorithm is terminated as all errors are assumed to have been corrected and LMMSE estimate of the current $\bar{\tilde{\mathbf{y}}}'^{(it)}$ is output (Eq. 3.90).

$$\hat{\tilde{\mathbf{d}}} = \sigma_d^2 \mathbf{I}\check{\mathbf{G}}^T \mathbf{H}_b^T (\mathbf{H}_b \check{\mathbf{G}} \sigma_d^2 \mathbf{I}\check{\mathbf{G}}^T \mathbf{H}_b^T + \sigma_n^2 \mathbf{I})^{-1} \bar{\tilde{\mathbf{y}}}'^{(it)} \tag{3.90}$$

- Otherwise, the correction value β_s^z corresponding to $\alpha_{max}(\bar{\tilde{\mathbf{y}}}')$ in Alg. 5 or to w_{min} in Alg. 4, where z indicates the bit flip position within the s-th vector element where the memory error has occurred, is applied to $\bar{\tilde{\mathbf{y}}}'$

$$\bar{\tilde{\mathbf{y}}}'^{(it+1)} = \bar{\tilde{\mathbf{y}}}'^{(it)} + \beta_s^z \tag{3.91}$$

- The subsequent iteration is started with $\bar{\tilde{\mathbf{y}}}'^{(it+1)}$
- In the last iteration, the LMMSE estimate with the last found correction value is output as the final solution

$$\hat{\tilde{\mathbf{d}}} = \sigma_d^2 \mathbf{I}\check{\mathbf{G}}\mathbf{H}_b^T (\mathbf{H}_b \check{\mathbf{G}} \sigma_d^2 \mathbf{I}\check{\mathbf{G}}^T \mathbf{H}_b^T + \sigma_n^2 \mathbf{I})^{-1} (\bar{\tilde{\mathbf{y}}}'^{(m_{it})} + \beta_s^z \mathbf{u}_s) \tag{3.92}$$

The number of iterations m_{it} can be picked arbitrarily and is the design parameter that trades off computational complexity and performance.

It is worth to mention that the [4].[8] fixed point format does not fit the range of values in MIMO UW-OFDM. It has been found by numeric experiment that the fixed point format optimal for MIMO-UW is [6].[8]. In this format, the possible error values are $\{\pm 2^5, \pm 2^4, \pm 2^3, \pm 2^2, \pm 2^1, \pm 2^0\}$ with probability $P(k = 1) = P_e(1 - P_e)^5$. It is obvious that the largest possible memory error value is larger in case of MIMO UW-OFDM compared to MIMO CP-OFDM.

3.2.3 Resilient likelihood ascent search

Any ML based detection algorithm can be modified to incorporate knowledge of memory errors [6]. Here, this modification is performed for MIMO UW-OFDM LAS algorithm from Sec. 3.1.3 following the approach in [6]. The derivation will be performed for MIMO UW-OFDM (real-valued system in Eq. 3.33). Although it has been argued that the single-flip single element assumption is not valid for MIMO UW-OFDM, the derivation will be first performed under this assumption for clarity of exposition.

As the attention is restricted to errors in integer part only, the likelihood for any element of the receive vector after the memory $\bar{\mathbf{y}}'$ can be written as

$$p(\bar{y}_i'|\mathbf{d}) = P(k = 0)p(y_i', k = 0|\mathbf{d}) + P(k = 1) \sum_{j=d}^{l-1} p^j(y_i', k = 1|\mathbf{d}) \tag{3.93}$$

Assuming that elements of receive symbol vector contain independent Gaussian noise and memory errors, the likelihood for $\bar{\mathbf{y}}'$ can be expressed as

$$p(\bar{\mathbf{y}}'|\mathbf{d}) = \prod_{i=1}^{2N_d M_t} p(\bar{y}_i'|\mathbf{d}) \tag{3.94}$$

Algorithm 4 Iterative NMMSE for MIMO UW-OFDM

Initialize $\bar{\tilde{\mathbf{y}}}'^{(it=1)} = \bar{\tilde{\mathbf{y}}}'$
for $it = 1, \ldots, m_{it}$ **do**
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      $\tilde{\tilde{y}}_i'^j = \text{BITGET}(\bar{\tilde{\mathbf{y}}}', j)$
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      Compute β_i^j by Eq. 3.82
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      Compute w_i^j by Eq. 3.83
    end for
  end for
  for all j **do**
    $w_{min}^j = \underset{i}{\text{argmin}}\, w_i^j$
  end for
  Compute w_{min} by Eq. 3.89
  if $it \neq m_{it}$ **then**
    if $m = 1$ **then**
      Output LMMSE estimate of $\bar{\tilde{\mathbf{y}}}'^{(it)}$ (Eq. 3.90)
    else
      $w_{min} \rightarrow \beta_s^z$
      Update $\bar{\tilde{\mathbf{y}}}'^{(it+1)}$ by Eq. 3.91
    end if
  else
    $w_{min} \rightarrow \beta_s^z$
    Output LMMSE estimate with final correction, Eq. 3.92
  end if
end for

Plugging Eq. 3.93 into Eq. 3.94:

$$p(\bar{\mathbf{y}}'|\mathbf{d}) = \prod_{i=1}^{2N_d M_t} \left(P(k=0)p(y_i', k=0|\mathbf{d}) + P(k=1) \sum_{j=d}^{l-1} p^j(y_i', k=1|\mathbf{d}) \right) \tag{3.95}$$

In [6] it is assumed that as $P(k=1)$ is much smaller than $P(k=0)$, terms of second order or higher of $P(k=1)$ in Eq. 3.95 can be ignored, as their contribution is negligible. Therefore, it is assumed that the vector contains either one bit flip or none (actually valid only for MIMO CP-OFDM).

Algorithm 5 Iterative log-NMMSE for MIMO UW-OFDM

Initialize $\bar{\tilde{\mathbf{y}}}'^{(it=1)} = \bar{\tilde{\mathbf{y}}}'$
for $it = 1, \ldots, m_{it}$ **do**
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      $\tilde{\bar{y}}_i'^{j} =$ BITGET$(\bar{\tilde{\mathbf{y}}}', j)$
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      Compute β_i^j by Eq. 3.82
    end for
  end for
  for $j = f, \ldots, l-1$ **do**
    for $i = 1, \ldots, 2NM_r$ **do**
      Compute w_i^j by Eq. 3.83
    end for
  end for
  for all j **do**
    $w_{min}^j = \underset{i}{\operatorname{argmin}}\, w_i^j$
  end for
  for $m = 1, \ldots, d+1$ **do**
    Compute $\alpha_m(\bar{\tilde{\mathbf{y}}}')$ by Eq. 3.86
  end for
  $\alpha_{max}(\bar{\tilde{\mathbf{y}}}') = \underset{m}{\operatorname{argmax}}\, \alpha_m(\bar{\tilde{\mathbf{y}}}')$
  if $it \neq m_{it}$ **then**
    if $m = 1$ **then**
      Output LMMSE estimate of $\bar{\tilde{\mathbf{y}}}^{(it)}$ (Eq. 3.90)
    else
      $\alpha_{max}(\bar{\tilde{\mathbf{y}}}') \rightarrow \beta_s^z$
      Update $\bar{\tilde{\mathbf{y}}}'^{(it+1)}$ by Eq. 3.91
    end if
  else
    $\alpha_{max}(\bar{\tilde{\mathbf{y}}}') \rightarrow \beta_s^z$
    Output LMMSE estimate with final correction, Eq. 3.92
  end if
end for

The likelihood is then expressed as follows

$$p(\bar{\mathbf{y}}'|\mathbf{d}) = P^{2N_dM_t}(k=0) \prod_{i=1}^{2N_dM_t} p(y_i', k=0|\mathbf{d}) + \\ + P(k=1)P^{2N_dM_t-1}(k=0) \sum_{s=1}^{2N_dM_t} \left(\sum_{j=d}^{l-1} p(y_{is}|\mathbf{d}) \prod_{i=1, i\neq s}^{2N_dM_t} p^j(y_i', k=0|\mathbf{d}) \right) \tag{3.96}$$

Furthermore,

$$p(\bar{\mathbf{y}}'|\mathbf{d}) = \frac{P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}} \exp\left(-\frac{\|\mathbf{Hd}-\mathbf{y}'\|_2^2}{N\sigma_n^2}\right) + \\ + \frac{P(k=1)P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}} \sum_{s=1}^{2N_dM_t}\sum_{j=d}^{l-1} \exp\left(-\frac{\|\mathbf{Hd}-\mathbf{y}'\pm 2^{(j-f)}\mathbf{u}\|_2^2}{N\sigma_n^2}\right) \tag{3.97}$$

As given in Eq. 2.112 the maximization yields the ML estimate. Therefore, in case of the likelihood given in Eq. 3.97 the maximization becomes

$$\hat{\mathbf{d}} = \underset{\mathbf{d}\in\mathbb{A}^{2N_dM_t}}{\operatorname{argmax}} \log(p(\bar{\mathbf{y}}'|\mathbf{d})) \tag{3.98}$$

where

$$\log(p(\bar{\mathbf{y}}'|\mathbf{d})) = \log\left(\frac{P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}} \exp\left(-\frac{\|\mathbf{Hd}-\mathbf{y}'\|_2^2}{N\sigma_n^2}\right) + \right. \\ \left. + \frac{P(k=1)P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}} \sum_{s=1}^{2N_dM_t}\sum_{j=d}^{l-1} \exp\left(-\frac{\|\mathbf{Hd}-\mathbf{y}'\pm 2^{j-f}\mathbf{u}\|_2^2}{N\sigma_n^2}\right)\right) \tag{3.99}$$

Next, Eq. 3.99 can be further reformulated as

$$\log(p(\bar{\mathbf{y}}'|\mathbf{d})) = \log\left(\exp\left(\frac{\log c_0 - \|\mathbf{Hd}-\mathbf{y}'\|_2^2}{N\sigma_n^2}\right) + \right. \\ \left. + \sum_{s=1}^{2N_dM_t}\sum_{j=d}^{l-1} \exp\left(\frac{\log c_1 - \|\mathbf{Hd}-\mathbf{y}'\pm 2^{(j-f)}\mathbf{u}\|_2^2}{N\sigma_n^2}\right)\right) \tag{3.100}$$

where

$$\log c_0 = N\sigma_n^2 \log\left(\frac{P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}}\right) \tag{3.101}$$

$$\log c_1 = N\sigma_n^2 \log\left(\frac{P(k=1)P^{2N_dM_t}(k=0)}{(\pi N\sigma_n^2)^{2N_dM_t}}\right) \tag{3.102}$$

Using max-log approximation the ML estimate is the given as

$$\hat{\mathbf{d}} = \underset{\mathbf{d}\in\mathbb{A}^{2M_tN_d}, j=f,\dots,l-1}{\operatorname{argmax}} \left(\log c_0 - \|\bar{\mathbf{y}}'-\mathbf{Hd}\|_2^2, \log c_1 - \|(\bar{\mathbf{y}}'\pm 2^{(j-f)}\mathbf{u}-\mathbf{Hd})\|_2^2\right) \tag{3.103}$$

Therefore, the respective minimization in case of memory errors is given as

$$\hat{\mathbf{d}} = \underset{\mathbf{d}\in\mathbb{A}^{2M_tN_d}, j=f,\dots,l-1}{\operatorname{argmin}} \left(\|\bar{\mathbf{y}}'-\mathbf{Hd}\|_2^2 - \log c_0, \|\bar{\mathbf{y}}'\pm 2^{(j-f)}\mathbf{u}-\mathbf{Hd}\|_2^2 - \log c_1\right) \tag{3.104}$$

Recall that in original LAS, the ML cost function after m-th iteration is given in Eq. 3.43. Now, looking at the minimization in Eq. 3.104, the ML cost function after m-th iteration

in case there are no memory errors is given as

$$C_0^m = \|\bar{\mathbf{y}}' - \mathbf{H}\mathbf{d}^{(i)}\|_2^2 - \log c_0 = \mathbf{d}^{(m)^T}\mathbf{H}^T\mathbf{H}\mathbf{d}^{(m)} - 2\bar{\mathbf{y}}'^T\mathbf{H}\mathbf{d}^{(m)} - \log c_0 \tag{3.105}$$

In case of memory errors the ML cost function after m-th iteration is of the form

$$C_1^m = \|(\bar{\mathbf{y}}' \pm 2^{(j-f)}\mathbf{u}) - \mathbf{H}\mathbf{d}^{(m)}\|_2^2 - \log c_1 = \mathbf{d}^{(m)^T}\mathbf{H}^T\mathbf{H}\mathbf{d}^{(m)} - 2(\bar{\mathbf{y}} \pm 2^{(j-f)}\mathbf{u})^T\mathbf{H}\mathbf{d}^{(m)} - \log c_1 \tag{3.106}$$

The cost difference C_0^{m+1} is therefore same as in original LAS:

$$\Delta C_0^{m+1} = C_0^{m+1} - C_0^m = \gamma_i^{(m)^2}\upsilon_{ii} - 2\gamma_i^{(m)}z_i^{(m)} \tag{3.107}$$

where $\mathbf{z}^{(m)} = \mathbf{H}^T(\mathbf{y}' - \mathbf{H}\mathbf{d}^{(m)})$, $z_i^{(m)}$ is the i-th entry of $\mathbf{z}^{(m)}$, and υ_{ii} is the (i,i) - th entry of the $\boldsymbol{\Upsilon} = \mathbf{H}^T\mathbf{H}$ matrix. The cost difference C_1^{m+1} is given as

$$\Delta C_1^{m+1} = C_1^{m+1} - C_1^m = \gamma_i^{(m)^2}\upsilon_{ii} - 2\gamma_i^{(m)}\bar{z}_i^{(m)} \tag{3.108}$$

where $\bar{\mathbf{z}}^{(m)} = \mathbf{H}^T((\bar{\mathbf{y}} \pm 2^{(j-f)}) - \mathbf{H}\mathbf{d}^{(m)})$, $\bar{z}_i^{(m)}$ is the i-th entry of $\bar{\mathbf{z}}^{(m)}$.

It can be concluded that the number of cost differences to be evaluated per iteration is increased from $2N_dM_t$ in original LAS to $(d+1)2N_dM_t$ in single-flip single element resilient version. This represents a severe increase in complexity, taking into account the assumption of a single error within the receive symbol vector. As single-flip multiple element is the true situation in MIMO UW-OFDM, the complexity of this approach is daunting.

As LAS is just an extension of LMMSE in MIMO UW-OFDM, it makes more sense to run iterative log-NMMSE first to correct most of the errors and thereafter run original LAS on top of it. In this case, the complexity increase is only from log-NMMSE and LAS could improve the BER further.

3.2.4 Simulation results

The BER results are obtained for both 802.11n and MIMO UW-OFDM system with parameters from Table 3.2 in a coded scenario. The bit flip probabilities are set to $P_e = 10^{-3}$ and $P_e = 10^{-4}$. In coded scenario, MIMO UW-OFDM system is simulated with $\check{\mathbf{G}}'$, as it provides the best performance. In uncoded scenario, MIMO UW-OFDM is generated with $\check{\mathbf{G}}_s''$ matrix, since it provides additional gain compared to $\check{\mathbf{G}}''$ when LAS is employed.

Simulation results for MIMO CP-OFDM system under memory errors are shown in Fig. 3.10, 3.11, 3.12, 3.13. The baseline LMMSE curve, where the memory is completely error free is plotted in blue for reference. The LMMSE result under memory errors with P_e bit flip probability is shown in red.

It can be observed that in the low SNR range, channel errors outweigh memory errors and therefore the BER curves are identical to the memory error free case. As the SNR increases, the memory errors outweigh the channel errors and introduce an error floor. This floor is independent of SNR and depends solely on P_e.

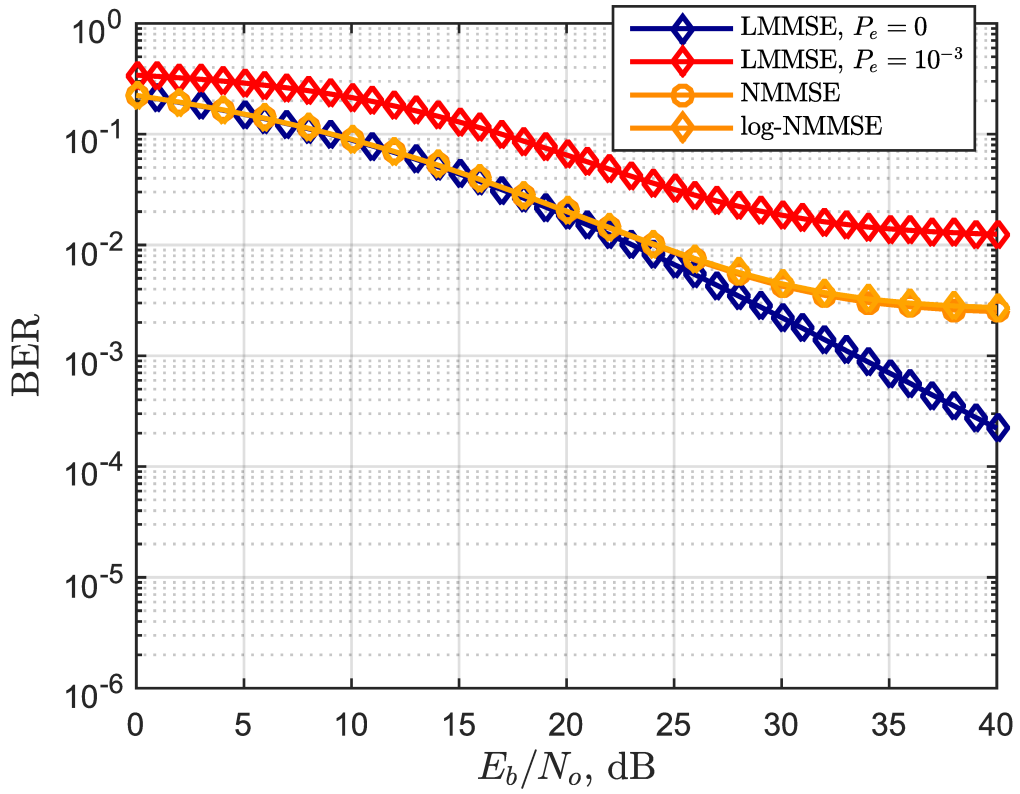


FIGURE 3.10: MIMO CP-OFDM, uncoded BER performance, $P_e = 10^{-3}$

Obviously, with lower P_e, the error floor is rising at larger SNR value and settles at lower BER value. The proposed NMMSE-based algorithms are able to lower the error floor and bring the BER performance curve closer to the memory error free case. It should be noted that the performance of the NMMSE and log-NMMSE algorithms (Alg. 2, Alg. 3) is identical under the single-flip, single element assumption. In coded transmission, the gain of the proposed algorithms is increasing and with $P_e = 10^{-4}$ the performance approaches that of the memory error free case (Fig. 3.13).

The BER results for MIMO UW-OFDM system and proposed versions of iterative NMMSE-based multiple error correction algorithms (Alg. 4, Alg. 5) are shown in Fig. 3.14, 3.15, 3.16, 3.17.

It can be observed that the impact of memory errors is in general stronger than for MIMO CP-OFDM due to multiple memory errors in the receive vector. Also, as discussed previously, iterative log-NMMSE (Alg. 5) is performing better than the iterative NMMSE (Alg. 4). As has been discussed, the existence of multiple errors leads to the numerical issue in iterative NMMSE. The values of exponents assume large values even with single error corrected. This leads to all zero $\alpha_m(\bar{\mathbf{y}}')$ coefficients. Therefore NMMSE cannot be computed in that case and simple LMMSE is used instead. In that case, only a subset of memory errors gets corrected. The performance of iterative NMMSE therefore does not depend on the maximum number of iterations, m_{it}. By transforming the computation to log domain, this numeric problem is resolved and iterative error correction is possible with log-NMMSE (Alg. 5). Here, the number of performed iterations has a decisive impact. Figure 3.15 shows that doubling the number of iterations reduces

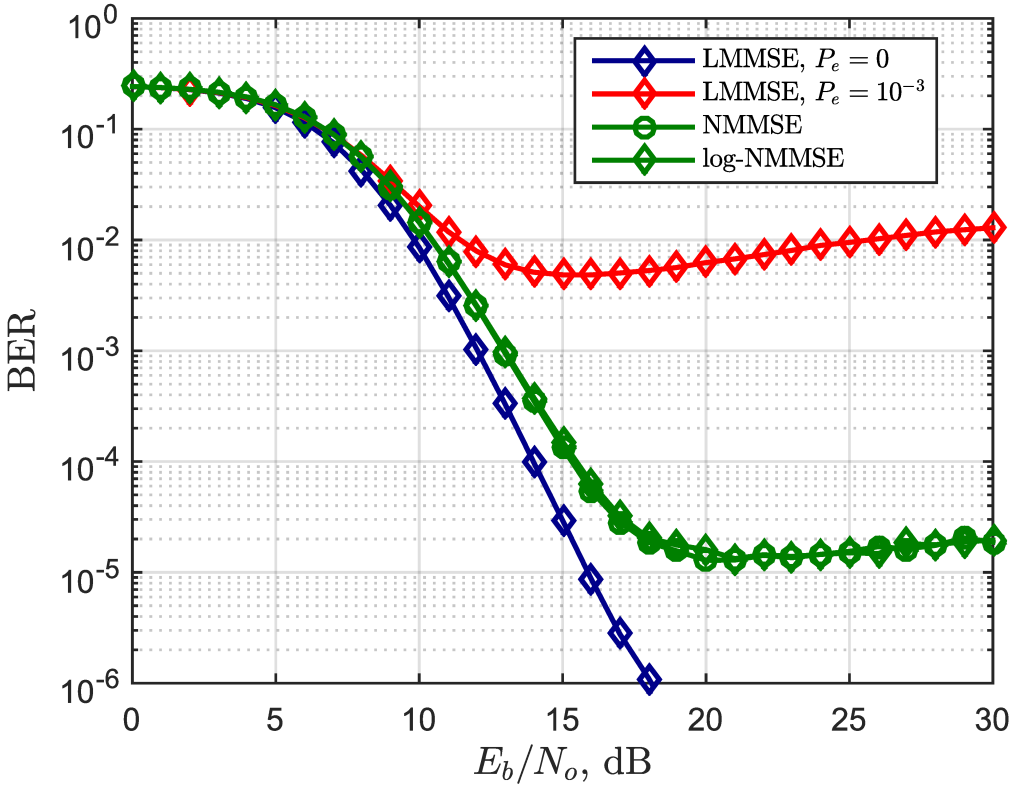


FIGURE 3.11: MIMO CP-OFDM, coded BER performance, $P_e = 10^{-3}$

the error floor by two orders of magnitude for $P_e = 10^{-3}$. For $P_e = 10^{-4}$, as shown in Fig. 3.17, with $m_{it} = 6$, there is an improvement of about 0.4 dB over $m_{it} = 3$ at BER $= 10^{-6}$.

The results of applying LAS on top of proposed memory error resilient algorithms for coded scenario are shown in Fig. 3.18, 3.19, 3.20, 3.21. It can be concluded that applying LAS on top of both iterative NMMSE and iterative log-NMMSE introduces a slight improvement in low SNR region, where the channel errors still outweigh the memory errors. However, in high SNR region, the improvement due to LAS vanishes. With $P_e = 10^{-4}$ and $m_{it} = 6$, as shown in Fig. 3.21, log-NMMSE is able to correct most of the memory errors in the high SNR region. LAS is then able to remove the remaining errors in the high SNR regime, such that the BER curve matches the one of the memory error free case.

3.3 Summary

Section 3.1 of this chapter addressed the generalization of UW-OFDM to the MIMO case. It has been shown shown that simple quasi-ML algorithms initially introduced for massive MIMO systems are applicable to MIMO UW with minor modifications. Simulation results indicate that MIMO UW-OFDM outperforms MIMO CP-OFDM in case of linear detection in both coded and uncoded cases. In case of quasi-ML detection

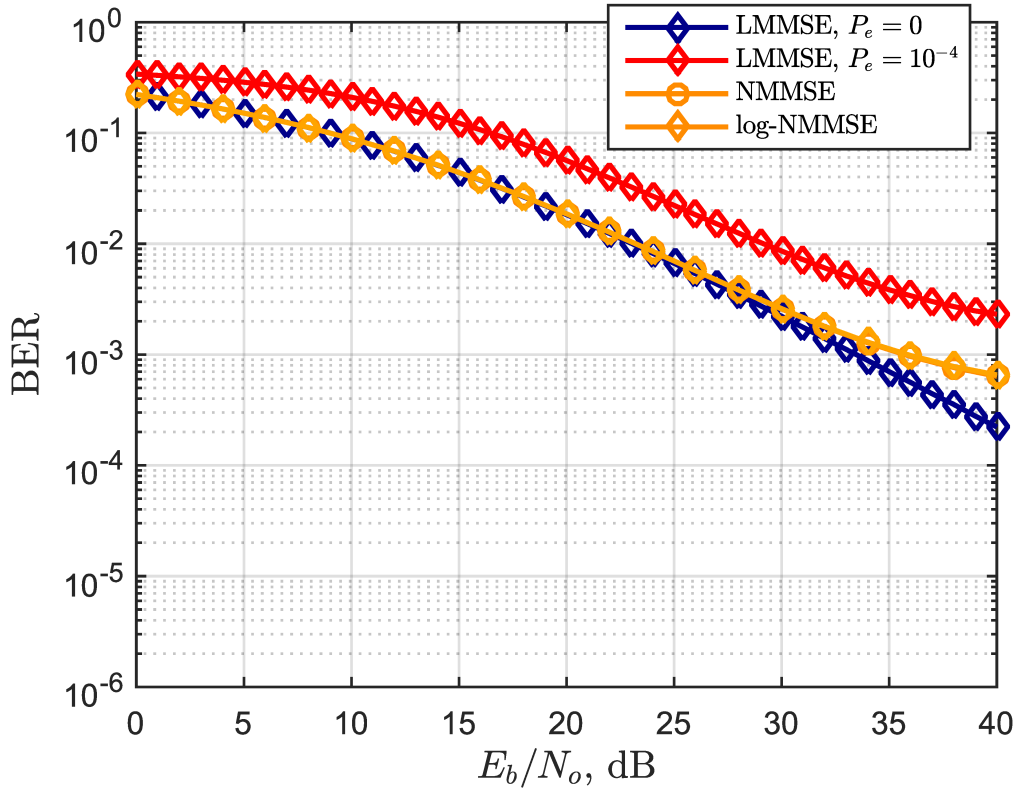


FIGURE 3.12: MIMO CP-OFDM, uncoded BER performance, $P_e = 10^{-4}$

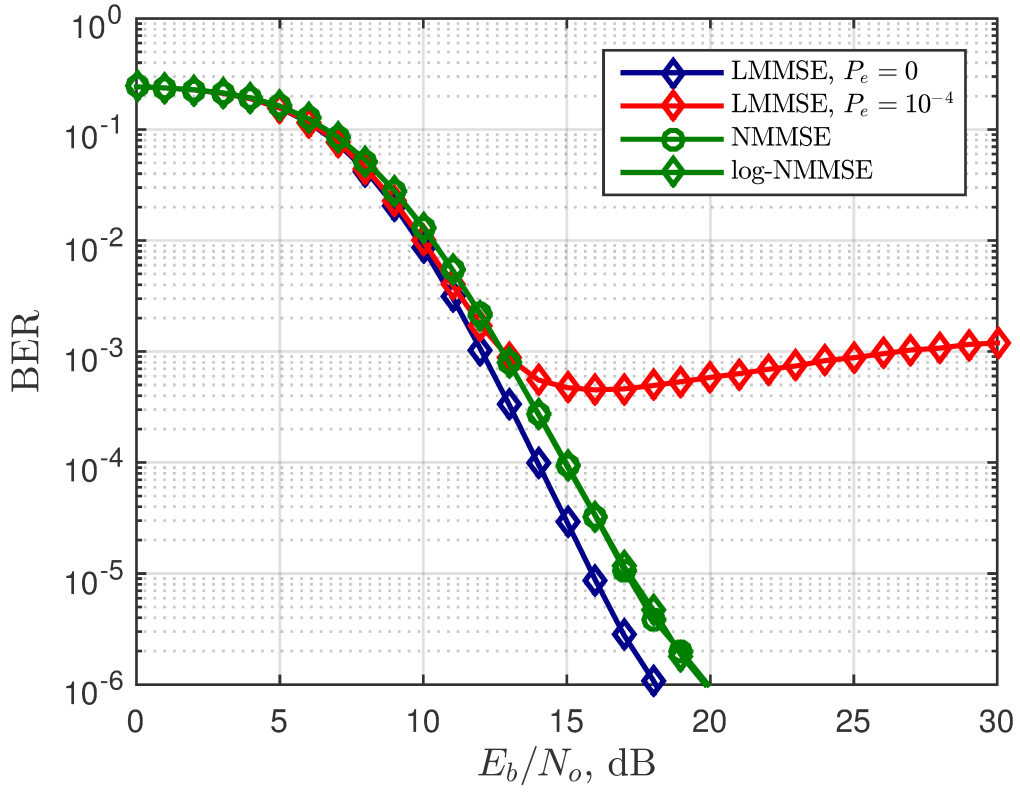


FIGURE 3.13: MIMO CP-OFDM, coded BER performance, $P_e = 10^{-4}$

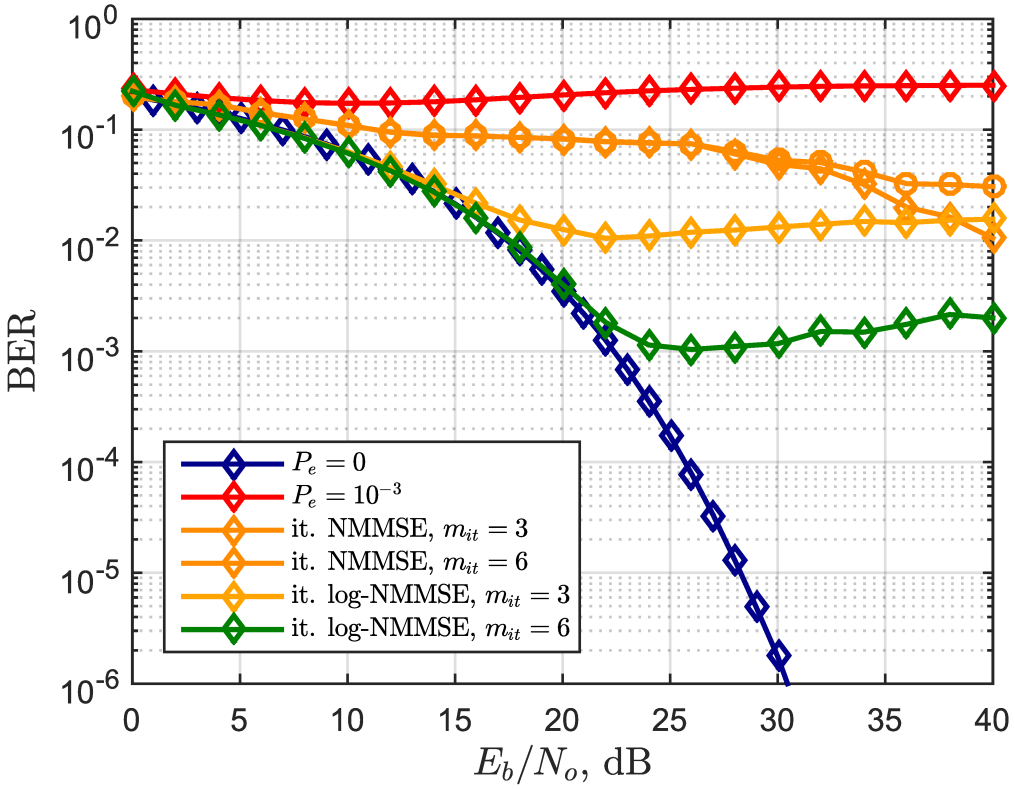


FIGURE 3.14: MIMO UW-OFDM, $\check{\mathbf{G}}_s''$, uncoded BER performance, $P_e = 10^{-3}$

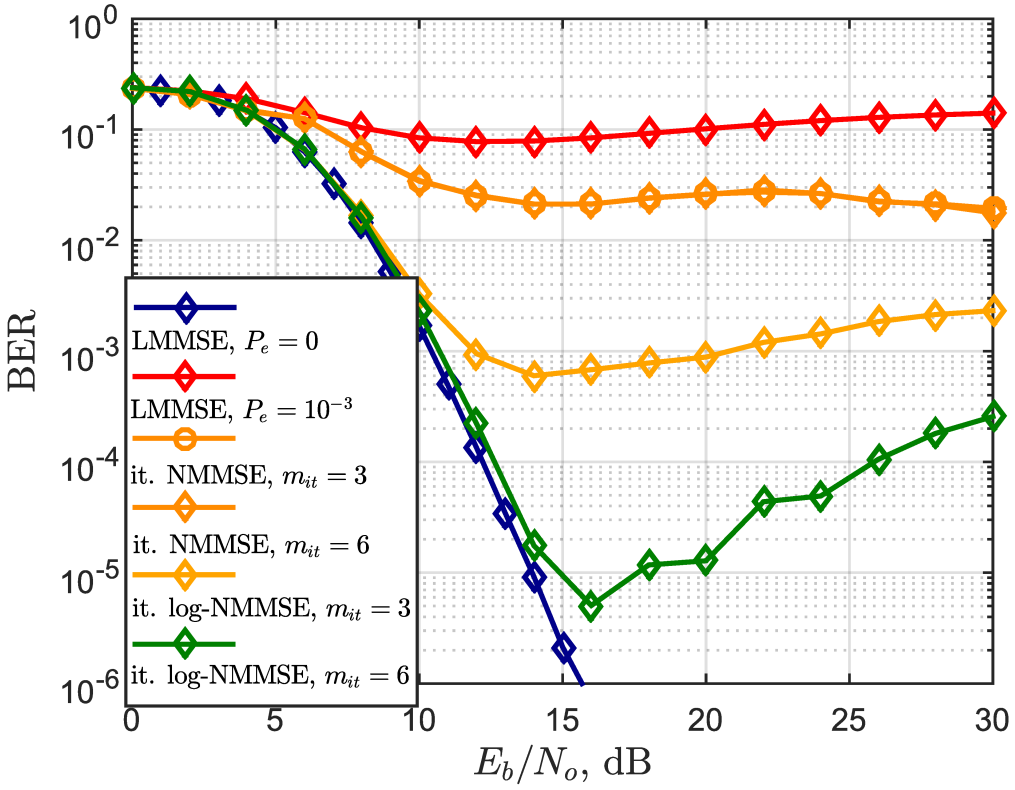


FIGURE 3.15: MIMO UW-OFDM, $\check{\mathbf{G}}'$, coded BER performance, $P_e = 10^{-3}$

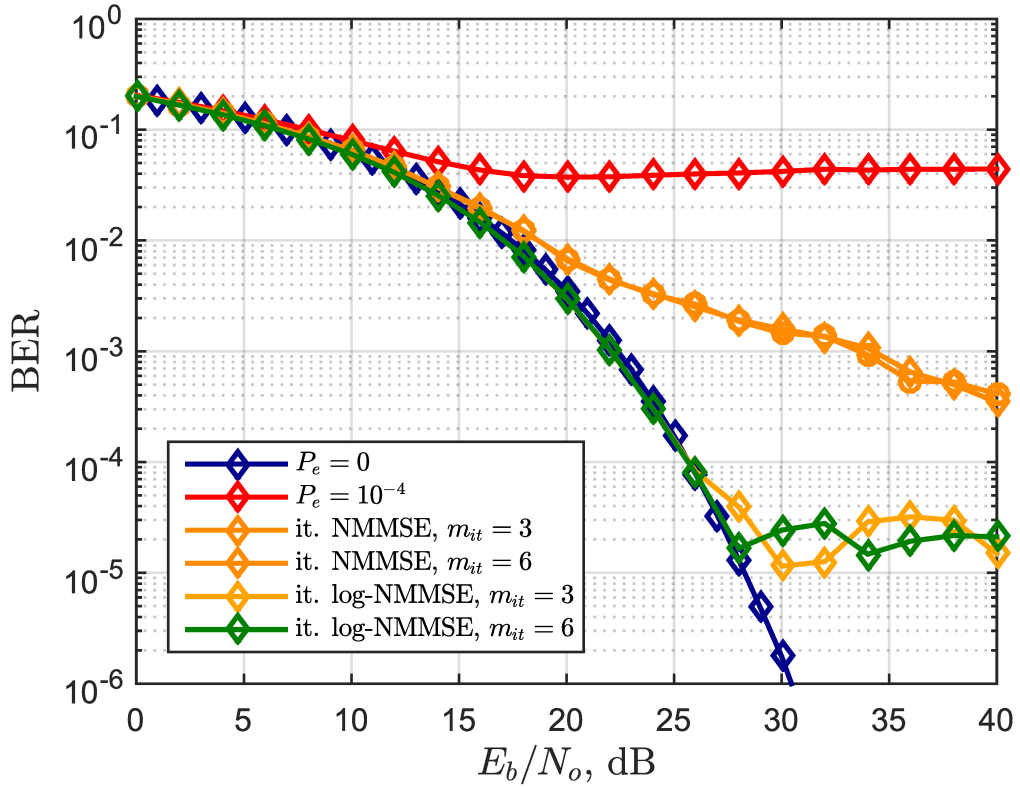


FIGURE 3.16: MIMO UW-OFDM, $\check{\mathbf{G}}_s''$, uncoded BER performance, $P_e = 10^{-4}$

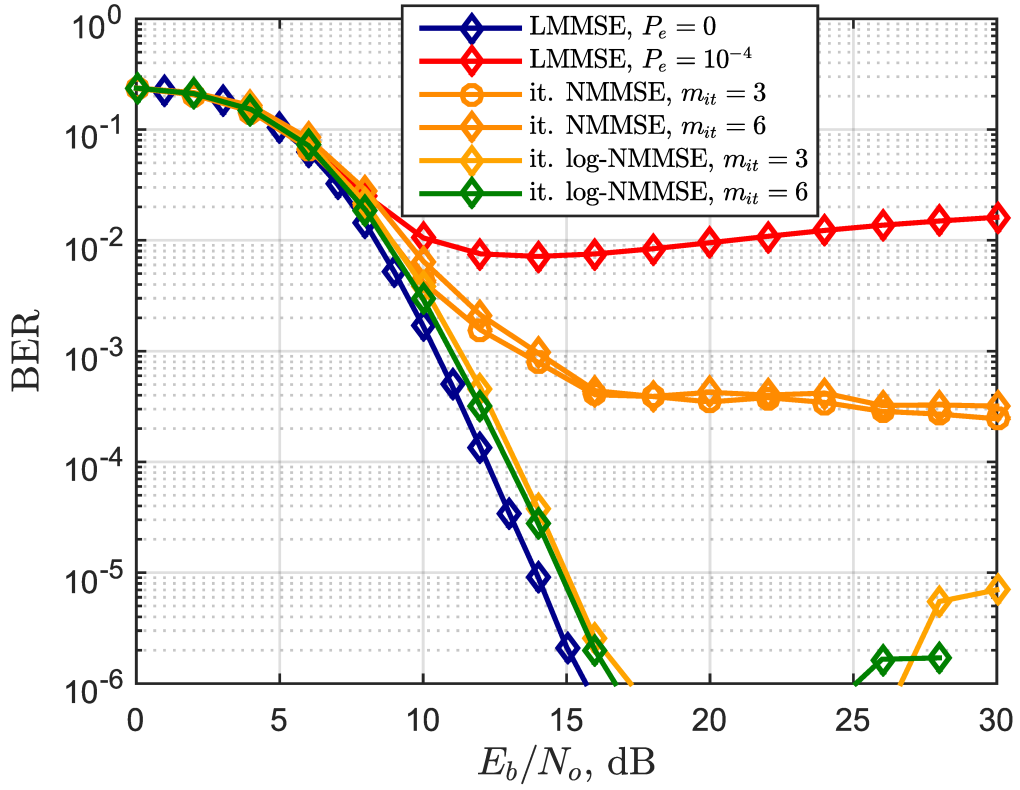


FIGURE 3.17: MIMO UW-OFDM, $\check{\mathbf{G}}'$, coded BER performance, $P_e = 10^{-4}$

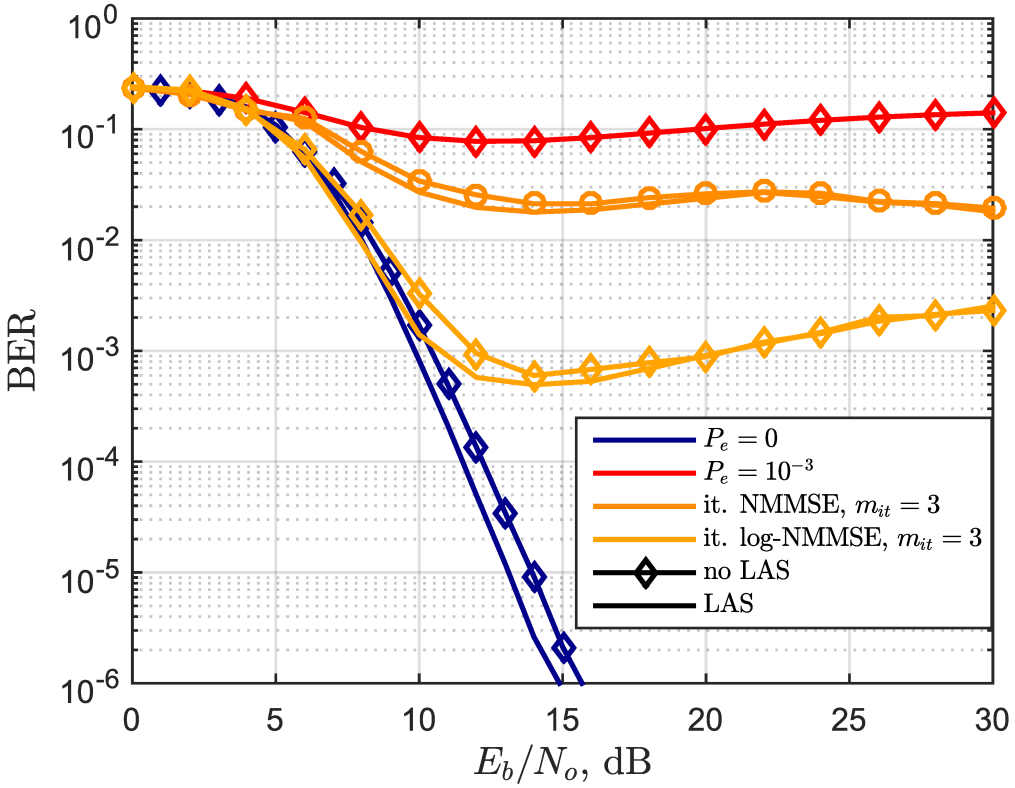


FIGURE 3.18: LAS (NMMSE), coded BER performance, $P_e = 10^{-3}$, $m_{it} = 3$

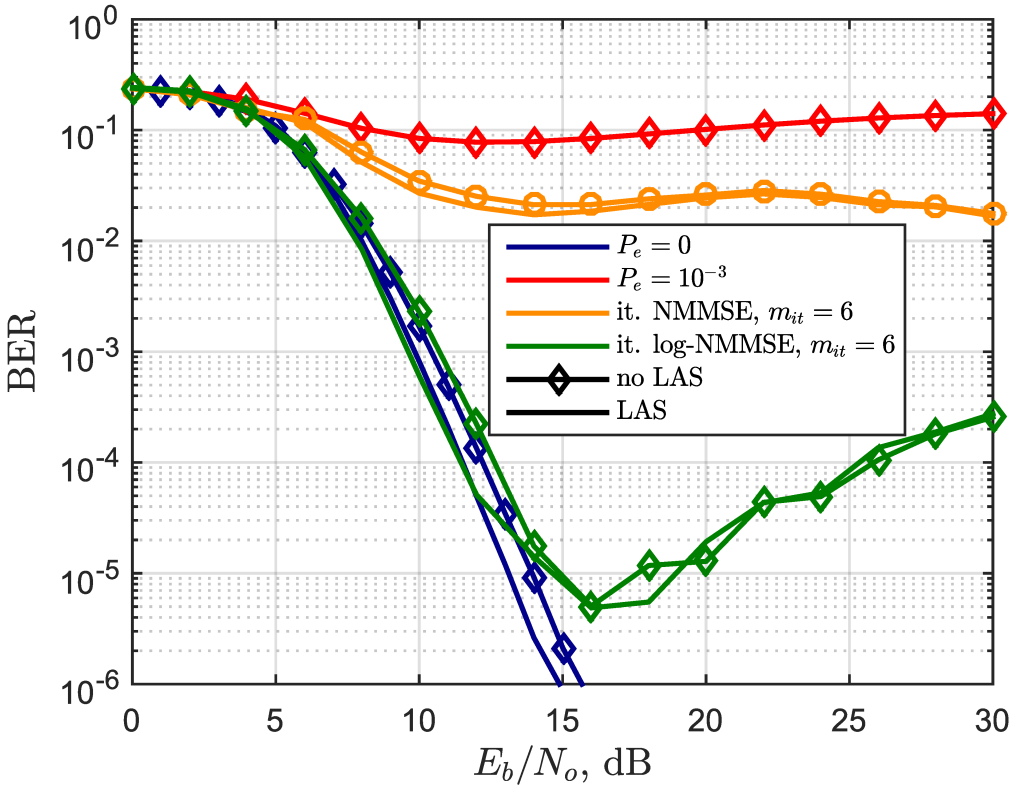


FIGURE 3.19: LAS (NMMSE), coded BER performance, $P_e = 10^{-3}$, $m_{it} = 6$

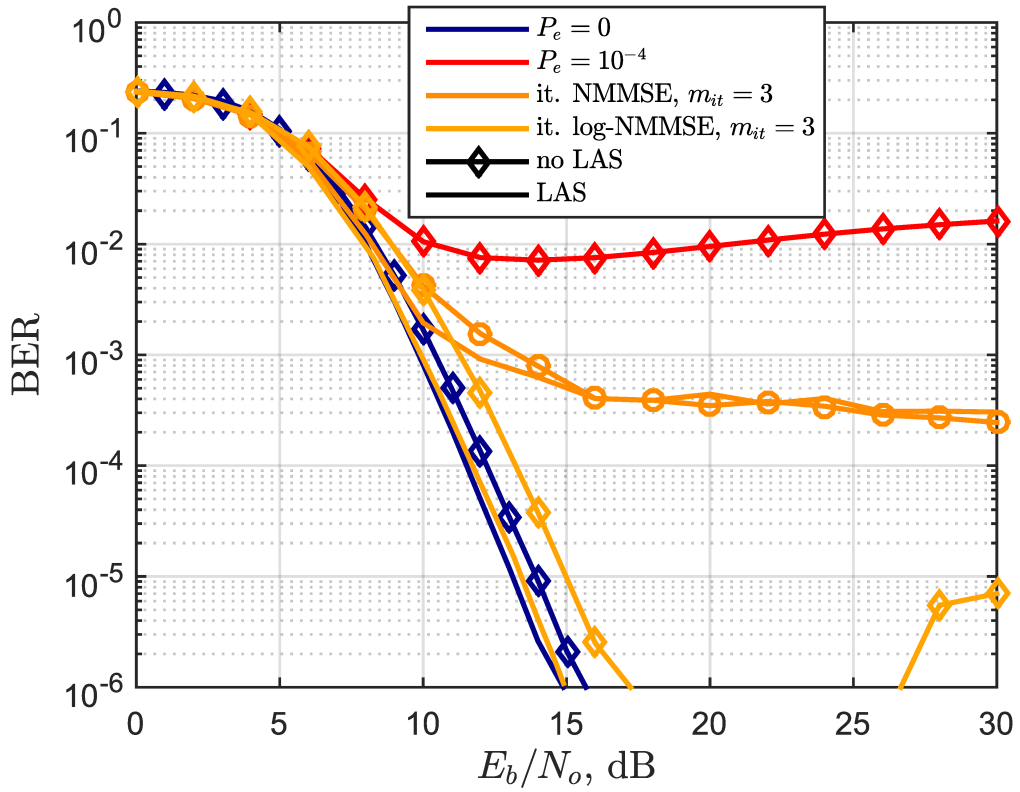


FIGURE 3.20: LAS (NMMSE), coded BER performance, $P_e = 10^{-4}$, $m_{it} = 3$

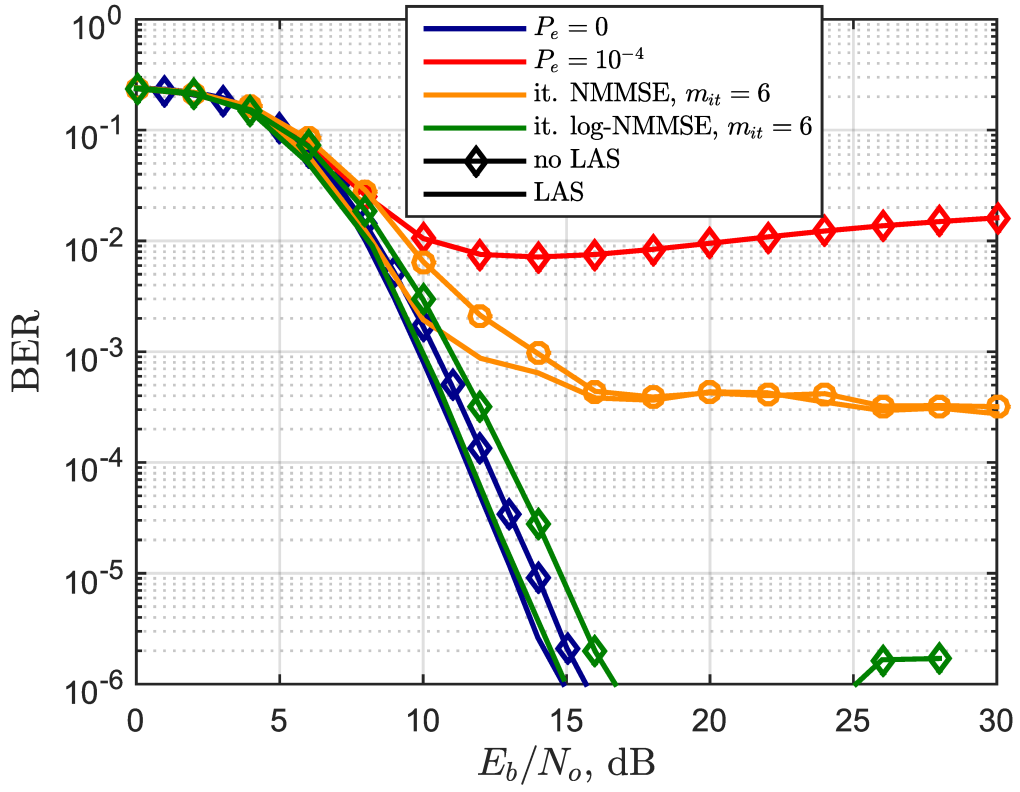


FIGURE 3.21: LAS (NMMSE), coded BER performance, $P_e = 10^{-4}$, $m_{it} = 6$

MIMO UW-OFDM is able to outperform SD of MIMO CP-OFDM in uncoded case. In coded system ML MIMO CP-OFDM performance cannot be achieved.

It can be concluded that MIMO UW-OFDM can be used as an intermediate solution between LMMSE MIMO CP-OFDM and ML MIMO CP-OFDM in a system employing four transmit and receive antennas. In case of 1-LAS, MIMO UW-OFDM exhibits lower complexity than ML MIMO CP-OFDM and is able to achieve close to ML MIMO CP-OFDM performance in both coded and uncoded setups. Obviously, the above conclusions are made without consideration of hardware implementation issues.

Proposed NMMSE iterative memory error resilient detection significantly reduces the impact of memory errors on the system performance in terms of BER. The impact of memory errors is more pronounced for MIMO UW-OFDM, since it represents a virtual massive MIMO system. Here, multiple memory errors within the receive vector have to be dealt with, opposed to MIMO CP-OFDM, where only single errors exist in the receive vector. Running LAS on top of the proposed log-NMMSE is able to remove the impact of memory errors completely for $P_e = 10^{-4}$.

The proposed modified detection algorithms include memory error detection and correction and can be regarded as a dynamic redundancy solution. Opposed to static memory error correcting codes, which have to be implemented in hardware, consuming area and power, proposed algorithms use available matrix-vector multiplication. Depending on the current value of SNR and the memory error rate, it could be chosen dynamically between standard LMMSE and proposed NMMSE-based algorithm. For example, in case of low memory error rate ($P_e < 10^{-6}$), LMMSE estimate is computed. In case of high memory error rate ($P_e \leq 10^{-4}$), the NMMSE-based detection is activated in order to reduce the impact of memory errors on the BER.

It is to note that there exist SNR regions (lower SNR range), where the channel errors still outweigh the memory errors. The proposed detection can be deactivated in that region. As SNR increases, the memory errors start to outweigh the channel errors and proposed detection should be activated. In future work it would be reasonable to obtain a threshold value of SNR for a given value of P_e. It should be also stressed that for high P_e and high SNR range, an error floor is inevitable even with proposed detection. However, as the memory errors are considered to be the result of VOS, it should be taken into account that this BER floor is traded off for a memory power reduction of 50% for $P_e = 10^{-3}$ [6].

Chapter 4

Reliability analysis of QR decomposition

QR decomposition (QRD) encountered in Sec. 2.2.3.2 enables the tree based search used in sphere decoding. In the virtual massive MIMO context, the QR decomposition can be used for computation of matrix inverse [153] in the LMMSE estimate. Since hardware implementation of MIMO UW-OFDM is not yet available, QR decomposition is treated in the sphere decoding context. An implementation of a MIMO CP-OFDM closed loop sphere decoding [97] is used in simulations within this chapter.

The QR decomposition can be performed by three distinct algorithm families, namely Gram-Schmidt orthogonalization process, Givens rotations and Householder reflections [57]. These algorithms perform the same task, differ however in underlying linear algebraic transformations and the number of required arithmetic operations. Gram-Schmidt process produces a unitary matrix from the input matrix by transforming it to an orthonormal basis. The upper triangular matrix, required for tree search is obtained as a side product. Givens rotations zeroes an element of the input matrix one by one by series of plane rotations until an upper triangular form is obtained. The unitary matrix is obtained as a side product. The Householder transformation tries to zero out the most elements of each column vector by reflection operations. The upper-triangular matrix is derived after each Householder transform matrix is applied to each column sequentially. A good comparison of these algorithms in terms of numerical complexity is found in [154]. In terms of hardware implementation, the Householder reflections reveal the highest complexity, since the Householder transformation matrices have to be multiplied sequentially in order to obtain the unitary matrix. Modified Gram-Schmidt process is used for MIMO channel preprocessing due to its numerical stability, compared to classical Gram-Schmidt process which is prone to rounding errors [155]. Givens rotations possess excellent numeric stability due to unitary rotations involved. For large matrices, the inherent parallelism of Givens rotations makes the algorithm more suitable for hardware implementation compared to other methods [154].

First, this chapter recapitulates the Givens rotations algorithm. Next, the basic hardware implementation, namely the systolic array along with its variants, is discussed. In terms of MIMO, the optimum performance is obtained if the QRD incorporates MMSE

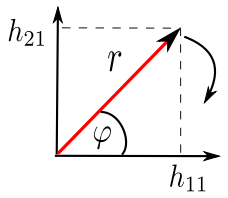


FIGURE 4.1: Rotation

sorting of the matrix columns [156]. However, sorting does not allow the direct application of the systolic array architecture. The architecture of a new version of the MMSE sorted Givens rotations algorithm based on conventional logic is next introduced [44]. The architecture is targeted for FPGA and synthesis results are discussed.

Finally, this chapter addresses the reliability analysis of combinational logic components of the implemented QR decomposition algorithm. Fault injection is performed in order to identify the the computational modules with the strongest impact of the overall performance of sphere decoding in terms of FER [45].

4.1 Givens rotations

As mentioned previously, Givens rotations seeks to null out the elements of the input matrix in order to obtain the upper-triangular form [157]. Consider a 1-by-M_t input matrix $\mathbf{H}$,

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{M_T} \\ h_{21} & h_{22} & \dots & h_{M_T} \end{bmatrix}$$

In order to bring it to upper-triangular form, h_{21} element has to be nulled. This is equivalent to rotating the vector $\begin{bmatrix} h_{11} & h_{21}^* \end{bmatrix}^H$ along the angle φ as shown in Fig. 4.1. Thereby, along with nulling the h_{21} coordinate, the h_{11} coordinate becomes equal to the norm r of the vector $\begin{bmatrix} h_{11} & h_{21}^* \end{bmatrix}^H$.

In matrix form, the rotation is defined by the unitary matrix $\mathbf{Q}^H$ [57],

$$\mathbf{Q}^H = \begin{bmatrix} \cos\varphi & \sin^*\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \tag{4.1}$$

where $(*)$ is the complex conjugate operation.

In general form, the rotation is given as

$$\mathbf{Q}^H\mathbf{H} = \mathbf{R} \tag{4.2}$$

or, in case of an example 1-by-M_t matrix

$$\begin{bmatrix} \cos\varphi & \sin^*\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} h_{11} & h_{12} & \dots & h_{M_T} \\ h_{21} & h_{22} & \dots & h_{M_T} \end{bmatrix} = \begin{bmatrix} r & h'_{12} & \dots & h'_{M_T} \\ 0 & h'_{22} & \dots & h'_{M_T} \end{bmatrix}$$

The sine and cosine of the rotation angle φ, as well as the vector norm are given as

$$\begin{aligned}\cos\varphi &= \frac{h_{11}}{r}\\ \sin\varphi &= \frac{h_{21}}{r}\\ r &= \sqrt{h_{11}^2 + |h_{21}|^2}\end{aligned}$$

The matrix multiplication in Eq. 4.2 also updates the remaining elements of the two rows involved in the rotation

$$h'_{1,2:\,M_T} = \cos\cdot h_{1,2:\,M_T} + \sin^*\cdot h_{2,2:\,M_T} \tag{4.3}$$
$$h'_{2,2:\,M_T} = -\sin\cdot h_{1,2:\,M_T} + \cos\cdot h_{2,2:\,M_T} \tag{4.4}$$

For the arbitrary input matrix $\mathbf{H}$, the matrix elements are nulled one by one until upper-triangular form $\mathbf{R}$ is obtained. The nulling of i,j -th element of $\mathbf{H}$, is performed by the corresponding unitary matrix $\mathbf{Q}^H(i,j)$. The later is formed from the identify matrix $\mathbf{I}$ of the appropriate size where the required elements are set as follows:

$$\begin{aligned}q^H_{jj}(i,j) &= \cos\varphi\\ q^H_{ji}(i,j) &= \sin^*\varphi\\ q^H_{ij}(i,j) &= -\sin\varphi\\ q^H_{ii}(i,j) &= \cos\varphi\end{aligned}$$

The resulting rotation matrix is given in general form as

$$\mathbf{Q}^H(i,j) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0\\ \vdots & \ddots & \vdots & & \vdots & & \vdots\\ 0 & \dots & \cos\varphi & \dots & \sin^*\varphi & \dots & 0\\ \vdots & & \vdots & \ddots & \vdots & & \vdots\\ 0 & \dots & -\sin\varphi & \dots & \cos\varphi & \dots & 0\\ \vdots & & \vdots & & \vdots & \ddots & \vdots\\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix} \tag{4.5}$$

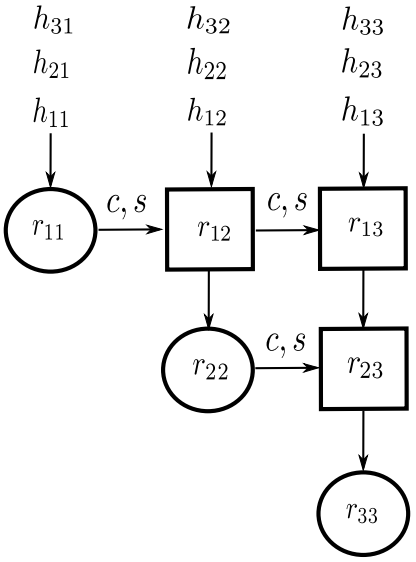


FIGURE 4.2: Systolic array

The input matrix is brought to upper-triangular form by sequential multiplication with element rotation matrices $\mathbf{Q}^H(i,j)$. Consider an example of 4-by-4 initial matrix

$$\mathbf{H} = \begin{bmatrix} \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \end{bmatrix} \xrightarrow{\mathbf{Q}^H(2,1)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ \star & \star & \star & \star \\ \star & \star & \star & \star \end{bmatrix} \xrightarrow{\mathbf{Q}^H(3,1)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ \star & \star & \star & \star \end{bmatrix} \xrightarrow{\mathbf{Q}^H(4,1)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix} \tag{4.6}$$

$$\xrightarrow{\mathbf{Q}^H(3,2)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & \star & \star & \star \end{bmatrix} \xrightarrow{\mathbf{Q}^H(4,2)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & \star & \star \end{bmatrix} \xrightarrow{\mathbf{Q}^H(4,3)} \begin{bmatrix} \star & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \\ 0 & 0 & 0 & \star \end{bmatrix}$$

The overall unitary matrix $\mathbf{Q}^H$ is obtained by sequentially multiplying the individual rotation matrices

$$\mathbf{Q}^H = \mathbf{Q}_n^H \dots \mathbf{Q}_2^H \mathbf{Q}_1^H \tag{4.7}$$

where n is the overall number of required rotations.

4.1.1 Systolic array

The inherent parallelism of Givens rotations is exploited by the systolic array architecture [158]. The systolic array is depicted in Fig. 4.2. The array takes the form of the resulting upper-triangular matrix. The input matrix is fed to the top row of the array row wise, starting from the first row. After the decomposition is performed, the elements of the resulting upper-triangular matrix $\mathbf{R}$ are stored within the registers of the array elements.

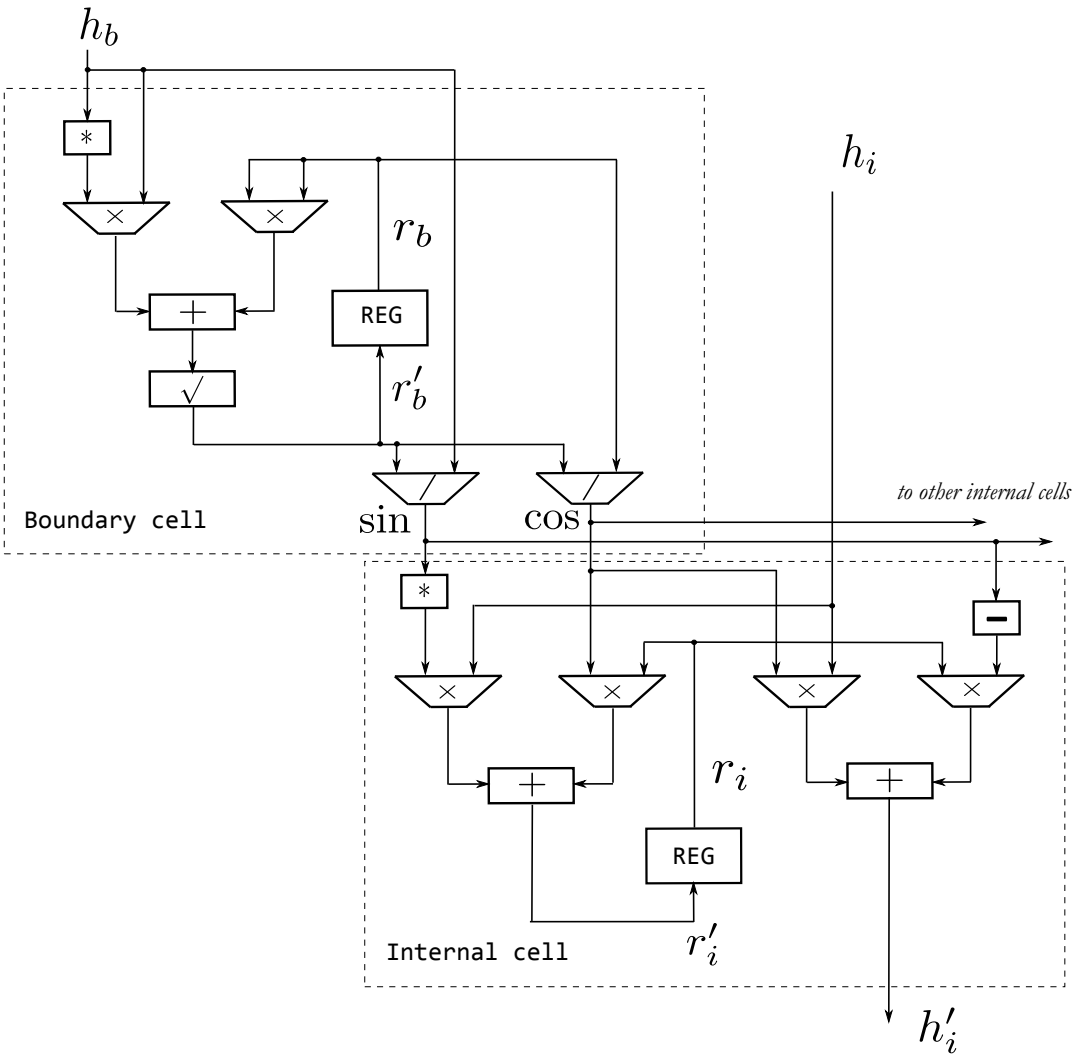


FIGURE 4.3: Boundary and internal cells of the systolic array

The array contains two types of elements: the boundary cell (denoted by a circle) and the internal cell (denoted by a square). A boundary cell computes the r_{ii} diagonal element of $\mathbf{R}$. It also computes the $\sin\varphi$ and $\cos\varphi$ and passes them to the internal cell. An internal cell computes the update of the remaining non-diagonal elements of the current row, $r_{ij}, i < j$ (Eq. 4.4).

Consider the architecture of the boundary cell, depicted in Fig. 4.3. As the first row of the matrix, $\begin{bmatrix} h_{11} & h_{12} & h_{13} \end{bmatrix}$ is fed to the array, the new value r'_b of the diagonal element is computed as

$$r'_b = \sqrt{(r_b^2 + |h_b|^2)} \tag{4.8}$$

During the first cycle, the value r_b stored in the register is still zero (array has been initialized) and the value at the input of the register equals the norm of the complex diagonal element: $r'_b = |h_{11}|$. Therefore, the boundary element stores the diagonal value

normalized to a real number. The rotation parameters are computed as follows:

$$s = \sin\varphi = \frac{h_b}{r'_b} \tag{4.9}$$

$$c = \cos\varphi = \frac{r_b}{r'_b} \tag{4.10}$$

In the first cycle, $s = h_{11}/|h_{11}|$ and $c = 0$, because $r_b = 0$. The rotation parameters c and s are then passed to the internal cells.

The architecture of the internal cell is depicted in Fig. 4.3. The internal cells gets as its inputs the off-diagonal element h_i of the input matrix row and the rotation parameters c and s. The new off-diagonal row element r'_i at the input of the register is computed as

$$r'_i = cr_i + s^* h_i \tag{4.11}$$

As $c = 0$ in the first cycle, $r'_i = \frac{h^*_{11}}{|h_{11}|}h_i$, where $h_i = h_{12}, h_{13}$. The value that is passed to the next row of the array, h'_i, is computed as

$$h'_i = ch_i - sr_i \tag{4.12}$$

during the first cycle this value equals zero, due to $c = 0$ and $r_i = 0$. Therefore, in the first cycle, the diagonal element of the input row is transformed to a real number and the remaining row elements are normalized by the conjugate of the diagonal element divided by its norm. No actual nulling has been yet performed. This architecture however allows the rotation matrix given by Eq. 4.1, where the $\cos\varphi$ assumes real values. This also implies that the right hand side multiplier and the right hand side divider of the boundary cell are real value circuits. The remaining multipliers and dividers as well as all computational elements of the internal cell are complex value circuits.

During the second cycle, the second row of the matrix, $\begin{bmatrix} h_{21} & h_{22} & h_{23} \end{bmatrix}$ is fed to the top row of the array. The boundary cell now computes the final diagonal element of the resulting upper-triangular matrix by Eq. 4.8. Obviously, $r'_b = \sqrt{|h_{11}|^2 + |h_{21}|^2}$. The rotation parameters are computed by Eq. 4.10 and passed the to internal cells. The internal cells of the first row update the remaining elements of the first row of the upper-triangular matrix by Eq. 4.11. The updated elements of the second row are computed by Eq. 4.12 and passed to the second row of the array, where the normalization of the diagonal element and the remaining elements of the second input row is performed.

During the third cycle, the third row of the input matrix is fed to the top row of the array. The top row performs nulling of the element h_{31}, whereas the second row of the array performs nulling of the element h_{32}. The third row computes the diagonal element r_{33} of $\mathbf{R}$.

The array therefore requires as many cycles as there are matrix rows to output the upper-triangular matrix. It can be observed that the array cells include square root and division operations that are quite costly in hardware. A number of array modifications have been proposed that try to remove these costly operations. The most prominent modifications are Squared Givens Rotations [159] and Modified Squared Givens Rotations [160] that exclude the square root operation and square root and division free Givens rotations proposed by the authors of [161]. Unfortunately all mentioned methods violate the

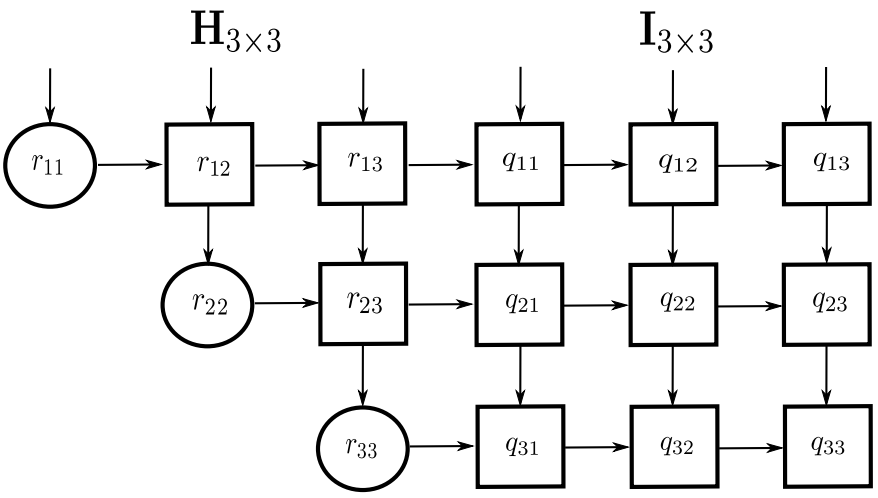


FIGURE 4.4: Systolic array with unitary matrix

unitarity of resulting $\mathbf{Q}$ and are therefore not applicable for QR decomposition in the context of sphere decoding. Multiplying the noise vector with a non-unitary matrix in Eq. 2.114 may lead to a severe noise enhancement.

The discussed systolic array keeps $\mathbf{R}$ in the registers of its cells. However, it does not produce the overall unitary matrix $\mathbf{Q}^H$. This matrix can be easily obtained using the following identity

$$\mathbf{Q}_n^H \dots \mathbf{Q}_2^H \mathbf{Q}_1^H \mathbf{I} = \mathbf{Q}^H \tag{4.13}$$

In terms of the systolic array this relation is implemented by adding the respective number of internal cells. The augmented array for the complete decomposition is depicted in Fig. 4.4. The additional internal cells are fed with an identity matrix of the respective size. Since the rotation parameters c and s are passed in each row from the boundary cell to all internal cells, the array simply produces the elements of $\mathbf{Q}^H$ in the registers of the additional internal cells. The functionality of the array elements is not affected.

4.2 MMSE sorted Givens rotations

In Sec. 2.2.3.2 it has been shown that the QR decomposition of the channel matrix (Eq. 2.61) yields a triangular system of equations, where the i-th symbol of the receive vector $\bar{\mathbf{y}}$ is given as

$$\bar{y}_i = r_{ii}x_i + \sum_{j=i+1}^{M_t} r_{ij}x_j + \bar{n}_i \tag{4.14}$$

Since $\bar{y}_{M_t}$ is totally free of interference from subsequent levels, its value divided by $\frac{1}{r_{M_t M_t}}$ is used to obtain the estimate of x_i. Proceeding further through the tree, interference free estimates of the transmit vector $\mathbf{x}$ are obtained, given decisions on each level are correct. The SNR of i-th level is determined by $|r_{ii}|^2$, assuming $\sigma_x = \mathbf{I}_{M_t}$ [156]. It therefore makes sense to start with the symbol with largest r_{ii}, in order to minimize error propagation. In that sense the columns of $\mathbf{R}$ and $\mathbf{Q}$ have to be sorted appropriately.

The approach in Eq. 4.14 does not take the noise statistics into account [156]. The LMMSE estimate incorporates the noise variance. In order to incorporate MMSE criterion into the QR decomposition, an augmented channel matrix is introduced

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H}^T & \sigma_n \mathbf{I} \end{bmatrix} \tag{4.15}$$

Then, Givens rotations operate on the composite matrix

$$\mathbf{Z}^{(0)} = \begin{bmatrix} \mathbf{H} & \mathbf{I} \\ \sigma_n \mathbf{I} & 0 \end{bmatrix} \tag{4.16}$$

such that the sequence of rotations, performed by unitary rotation matrices $\mathbf{Q}_i^H$ yields

$$\mathbf{Z}^{(N)} = \mathbf{Q}_n^H \cdots \mathbf{Q}_1^H \mathbf{Z}^{(0)} = \begin{bmatrix} \mathbf{R} & \mathbf{Q}_a^H \\ 0 & \mathbf{Q}_c^H \end{bmatrix} \tag{4.17}$$

where $\mathbf{R}$ is the desired upper triangular matrix, and $\mathbf{Q}_a^H$ is the overall rotation matrix.

In terms of sphere decoding, MMSE criterion and proper sorting would decrease the computational complexity [162]. The MMSE sorted Givens rotations algorithm is summarized in Alg. 6. As discussed previously, the last diagonal element of $\mathbf{R}$, $r_{M_t M_t}$ should be the largest after the decomposition to allow optimal detection order. The problem is that QRD starts from the first diagonal element, r_{11}. Therefore, r_{ii} is minimized in the order it is computed $(1, \ldots, M_t)$ instead of being maximized in the order of detection $M_t, \ldots, 1$ [163]. Before computing r_{11}, the column norms of $\bar{\mathbf{H}}$ are obtained, and the column with the minimum norm is placed first (lines 6 and 7 of Alg. 6). The column norms do not have to be computed again before obtaining r_{22}. they are updated iteratively (line 10 of Alg. 6).

Algorithm 6 MMSE sorted Givens rotations algorithm

1: $\mathbf{Z} = \mathbf{Z}^{(0)}$, $\mathbf{p} = \begin{bmatrix} 1 \ldots M_T \end{bmatrix}$
2: **for** $j = 1, \ldots, M_T$ **do**
3: $\quad \nu_j = \|\mathbf{h}_j\|$ ▷ initial column norms
4: **end for**
5: **for** $i = 1, \ldots, M_T$ **do**
6: $\quad k = \arg\min_{j=i,\ldots,M_T} \nu_j$
7: $\quad$ exchange columns i and k in permutation vector $\mathbf{p}$ and in the first $M_R + i - 1$ rows of $\bar{\mathbf{H}}$
8: $\quad$ perform Givens Rotations using matrices such that rows $i + M_R, \ldots, i + 1$ of column $\mathbf{z}_i$ become zero
9: $\quad$ **for** $j = i, \ldots, M_T$ **do**
10: $\quad\quad \nu_j = \nu_j - \|z_{ij}\|^2$ ▷ update column norms
11: $\quad$ **end for**
12: **end for**

4.2.1 Hardware implementation

The main drawback of sorting is that the systolic array architecture can not be applied directly. After nulling all elements of current column, the columns need to be resorted, based on their updated norms. Therefore, $\mathbf{Z}$ has to be stored in the memory and updated after nulling of elements under a diagonal element is completed. Therefore, the rotation processing can be performed by just a single line of systolic array.

It is open to decide how the Givens rotations are computed. Alg. 7 depicts Givens rotations which follow the operation of systolic array cells in Fig. 4.3. Alg. 7 assumes rotation matrix in Eq. 4.1. With this algorithm, a single rotation requires two cycles: one for transforming the complex diagonal element to real domain and row normalization, and second for the actual rotation.

Algorithm 7 Givens rotations 1

$\mathbf{R} = \mathbf{H}$, $\mathbf{Q}^H = \mathbf{I}$
for $j = 1, \ldots, M_T$ **do**
  $l = \sqrt{r_{jj} \cdot r_{jj}^*}$ ▷ normalize diagonal entry
  $n = r_{jj}^*/l$
  $r_{j(j\colon M_T)} = r_{j(j\colon M_T)} \cdot n$ ▷ normalize row j
  $\mathbf{Q}^H = \mathbf{Q}^H(j,j) \cdot \mathbf{Q}^H$, $\mathbf{Q}_{jj}^H(j,j) = n$
  for $i = j+1, \ldots, M_R$ **do**
    $l = \sqrt{r_{jj}^2 + r_{ij} \cdot r_{ij}^*}$
    $\cos\varphi = r_{jj}/l$
    $\sin\varphi = r_{ij}/l$
    $r_{j(j\colon M_T)} = \cos\varphi \cdot r_{j(j\colon M_T)} + \sin^*\varphi \cdot r_{i(j\colon M_T)}$ ▷ perform rotation
    $r_{i(j\colon M_T)} = -\sin\varphi \cdot r_{j(j\colon M_T)} + \cos\varphi \cdot r_{i(j\colon M_T)}$
    $\mathbf{Q}^H = \mathbf{Q}^H(i,j) \cdot \mathbf{Q}^H$ ▷ update rotation matrix
  end for
end for

The transformation of diagonal element to real domain can be omitted by using the rotation matrix

$$\mathbf{Q}^H = \begin{bmatrix} \cos^*\varphi & \sin^*\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \tag{4.18}$$

The corresponding Givens rotations algorithm is summarized in Alg. 8. Here, with one line of systolic array, one rotation is performed in a single cycle. This speed up is gained at the cost of having all computational blocks complex-valued. Only the last diagonal element has to be transformed to real domain, as it remains complex after matrix multiplications by $\mathbf{Q}^H(i,j)$. Assuming MMSE sorting, this is not a problem due to the structure of $\bar{\mathbf{H}}$. For example, consider augmented matrix $\bar{\mathbf{H}}$, obtained from 4x4

channel matrix $\mathbf{H}$:

$$\bar{\mathbf{H}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \\ \sigma_n & 0 & 0 & 0 \\ 0 & \sigma_n & 0 & 0 \\ 0 & 0 & \sigma_n & 0 \\ 0 & 0 & 0 & \sigma_n \end{bmatrix} \tag{4.19}$$

Nulling the σ_n in row $2M_T$ also normalizes the resulting diagonal element r_{44}. Of course, care has to be taken as sorting could change the columns, such that all elements in rows below row M_T in column M_T are zero. In case if rotations are performed on each element, without checking if it is already zero, diagonal element will be transformed to real domain anyway. If such check is performed and respective rotation is skipped, a rotation counter can be used for the last column. If the counter is zero, just a single rotation has to be performed on any element of the column M_t in rows below row M_T and the diagonal element r_{44} will be transformed to real domain.

Algorithm 8 Givens rotations 2

1: $\mathbf{R} = \mathbf{H}$, $\mathbf{Q}^H = \mathbf{I}$
2: **for** $j = 1, \ldots, M_T$ **do**
3: **for** $i = j + 1, \ldots, M_R$ **do**
4: $l = \sqrt{r_{jj} \cdot r_{jj}^* + r_{ij} \cdot r_{ij}^*}$
5: $\cos\varphi = r_{jj}/l$
6: $\sin\varphi = r_{ij}/l$
7: $r_{j(j\colon M_T)} = \cos^*\varphi \cdot r_{j(j\colon M_T)} + \sin^*\varphi \cdot r_{i(j\colon M_T)}$ ▷ perform rotation
8: $r_{i(j\colon M_T)} = -\sin\varphi \cdot r_{j(j\colon M_T)} + \cos\varphi \cdot r_{i(j\colon M_T)}$
9: $\mathbf{Q}^H = \mathbf{Q}^H(i,j) \cdot \mathbf{Q}^H$ ▷ update rotation matrix
10: **end for**
11: **end for**
12:

Alg. 8 is preferable over Alg. 7 due to less required cycles. However, complex division operations that are costly in hardware are still required. In order to trade off divisions for multiplications, inverse square root version of Givens rotations can be performed. The respective algorithm is summarized in Alg. 9.

With fixed point arithmetic, the inverse square root operation can be efficiently implemented using only shift operations and additions/subtractions [164]. Alg. 9 will be used for rotation processing implementation.

4.2.1.1 Real-time requirements

In order to provide truly real-time compliant implementation, the QRD must satisfy the stringent requirements posed by the current standards. Since the channel matrix evolves over time, the QRD must be computed at a sufficient rate. The time span during which

Algorithm 9 Givens rotations using inverse square root

1: $\mathbf{R} = \mathbf{H}$, $\mathbf{Q}^H = \mathbf{I}$
2: **for** $j = 1, \ldots, M_T$ **do**
3: **for** $i = j + 1, \ldots, M_R$ **do**
4: $l^2 = r_{jj} \cdot r_{jj}^* + r_{ij} \cdot r_{ij}^*$
5: $inv\_sqrt = \frac{1}{\sqrt{l^2}}$; $l = l^2 \cdot inv\_sqrt$
6: $\cos\varphi = r_{jj} \cdot inv\_sqrt$; $\sin\varphi = r_{ij} \cdot inv\_sqrt$
7: Perform Rotation
8: $r_{j(j:\,M_T)} = \cos\varphi^* \cdot r_{j(j:\,M_T)} + \sin\varphi^* \cdot r_{i(j:\,M_T)}$
9: $r_{i(j:\,M_T)} = -\sin\varphi \cdot r_{j(j:\,M_T)} + \cos\varphi \cdot r_{i(j:\,M_T)}$
10: Update Rotation Matrix $\mathbf{Q}^H$
11: **end for**
12: **end for**

$\mathbf{H}$ is valid (coherence time) is calculated by $t_{\text{coh}} = c/(v_r \cdot f)$, where c is the speed of light, v_r is the relative speed of the receiver, and f is the frequency. In LTE standards [63], $f = 2.4$ GHz and v_r is either 250 or 500 km/h, implying the coherence time of 1.8 ms or 0.9 ms for $v_r = 250$ km/h or $v_r = 500$ km/h, respectively. During the coherence time QRD must be performed for each of 1021 sub-carriers. This yields real-time constraints of 1.763 μs or 0.881 μs for the complete QRD [164].

4.2.1.2 Architecture

The developed hardware architecture operates on fixed point numbers of the format [**M**].[**N**] = [4].[8], which represents the best tradeoff between word length and numeric accuracy, obtained from numeric experiments.

As already mentioned, systolic array cannot be used directly as MMSE sorted Givens rotations requires column sorting. Therefore, the architecture incorporates a central memory element to store the composite matrix $\mathbf{Z}$, which is updated after the subsequent rotations are performed. It also contains *INNER PRODUCT* block required for initial column norm computation. The result is stored in *NORM VECTOR* memory. This memory is updated by the *NORM UPDATE* block after column sorting is performed. The *PERMUTATION VECTOR* block updates and stores the permutation vector to keep track of the performed sorting.

The rotation processing is implemented by a single line of a systolic array. Compared to the line of the classical systolic array in Fig. 4.4, which processes one matrix row at a time, the current implementation is able to perform one nulling operation on two rows of $\mathbf{Z}$ at the same time. Figure 4.5 depicts the QRD circuit architecture for $M_t = M_r = 4$.

The line of systolic array, used for rotation processing, contains two types of elements: the *BOUNDARY CELL*, which computes the new diagonal element and $\cos\varphi$ and $\sin\varphi$ values, and the *INTERNAL CELL*, which updates the remaining row elements. The architectures of the boundary and internal cells are depicted in Fig. 4.6. They are implemented according to Alg. 9. The boundary cell contains only real value processing elements, since norm squared of the complex number in line 4 of Alg. 9 can be computed by real-value operations. The internal cell contains complex value adders and multipliers.

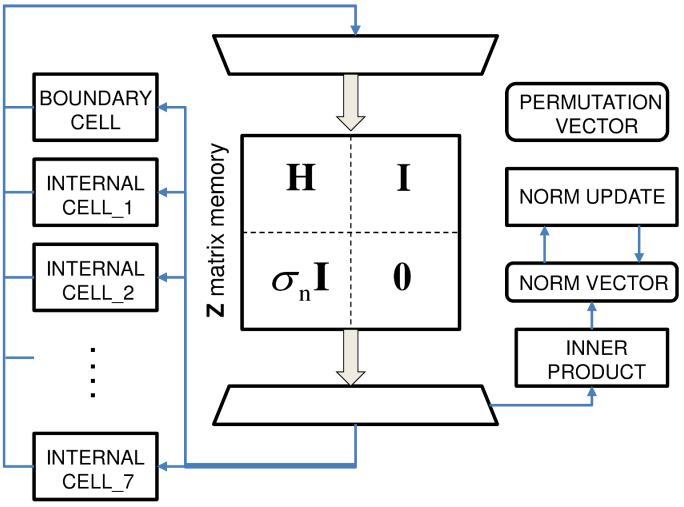


FIGURE 4.5: MMSE sorted Givens QRD architecture

Each complex multiplier contains four real multipliers and two real adders. The $\cos\varphi$ and $\sin\varphi$ values are written into the complex-valued registers between the boundary and internal cells in order to shorten the combinational path. Hence, one rotation, or nulling of one element of $\bar{\mathbf{H}}$, takes two cycles.

The state machine of the implemented MMSE sorted Givens rotations is depicted in Fig. 4.7. It comprises following states:

IDLE - In this state the circuit is expecting the *start* signal. When start is high, $\mathbf{H}$ is written row-wise to the matrix memory. This write takes 4 cycles for the example

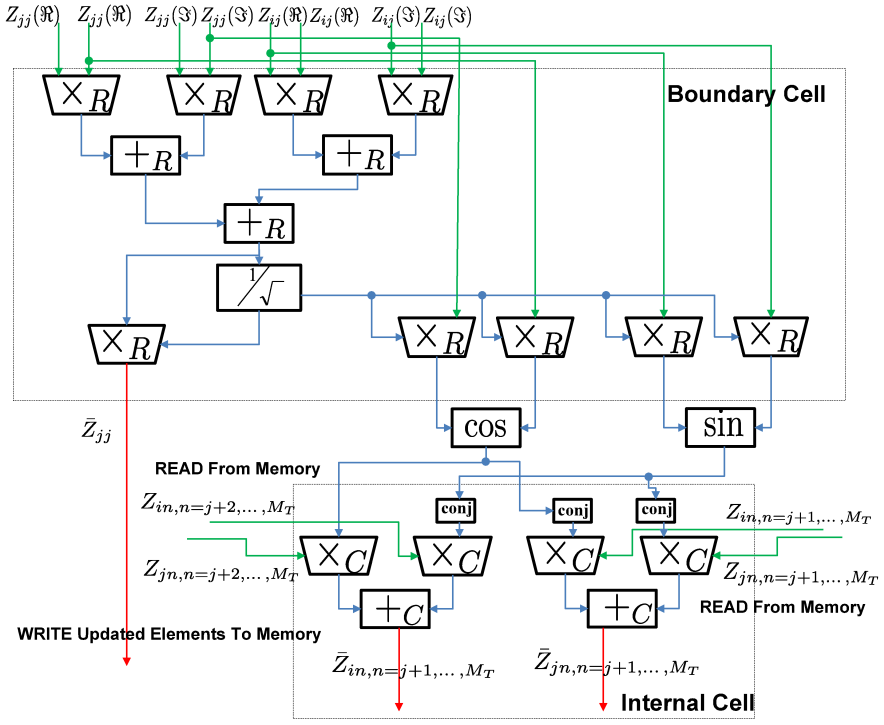


FIGURE 4.6: Rotation processing

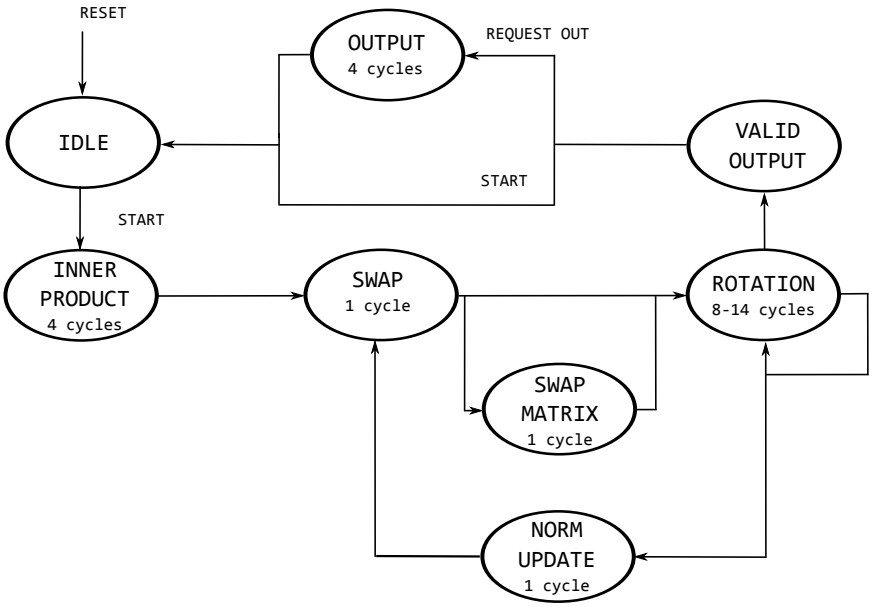


FIGURE 4.7: State machine of QR decomposition

$M_t = M_r = 4$ system. Other matrix memory regions are initialized according to Eq. 4.16. Thereafter, the state machine changes to the INNER PRODUCT state.

INNER PRODUCT - This state is held for M_r cycles. In every cycle a column of the matrix $\bar{\mathbf{H}}$ is read out from the matrix memory and its norm is computed. Therefore, the matrix memory is switched to column mode. Thereafter, the state machine changes to SWAP state.

SWAP - This state lasts for one cycle. Here, the column norms are compared, and the permutation vector is obtained and stored in the *PERMUTATION VECTOR* register. Thereafter, the state machine moves to SWAP MATRIX state, in case the column of the matrix have to be permuted. If it is not the case, the state machine moves directly to the ROTATION state.

SWAP MATRIX - The matrix column are permuted in this state in a single cycle, according to the permutation vector obtained in the SWAP state. The matrix memory is switched to the two column read and write mode.

ROTATION - In this state the actual nulling of the matrix element is performed. The state lasts a variable number of cycles. The boundary cell computes the new value of the diagonal element in one cycle. In the next cycle the internal cells compute the new values of the off-diagonal elements. The state is held as long as all elements under the diagonal element of the current column are zeroed out. Thereafter, the state machine moves to NORM UPDATE state if further permutation of columns is possible. If not, but other columns are to be rotated, the state machine moves again to the ROTATION state. Otherwise, the state machine moves to VALID OUTPUT state. The matrix memory is set to the two rows read and write mode.

NORM UPDATE - In this state the column norms are updated in a single cycle. Thereafter, the state machine moves to SWAP state.

VALID OUTPUT - In this state, the result of the QR decomposition is available in the matrix memory. If the resulting matrices are requested (*request out* signal), the state machine changes to the OUTPUT state. In case the start signal is applied, the state machine moves to the IDLE state without outputting the result of the decomposition.

OUTPUT - In this state the upper triangular and orthogonal matrices are read out column-wise and the state machine moves to IDLE state. The read out takes again M_t number of cycles.

The number of cycles required to zero out elements of one columns varies from 8 cycles for the M_t-th column and 14 cycles for the first column of $\bar{\mathbf{H}}$. In order to decrease the number of cycles required by rotation, it is checked if a value to be zeroed out already equals zero. In such a case, the rotation in the current row is skipped. This way, six cycles can be saved, and for $M_t \times M_r$ matrix $\mathbf{H}$ with $M_t = M_r = 4$, the overall number of cycles required for complete rotation equals $(22 - 6) \cdot 2 + 6 = 38$. It is illustrated by the following example:

$$\bar{\mathbf{H}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \\ \sigma_n & 0 & 0 & 0 \\ 0 & \sigma_n & 0 & 0 \\ 0 & 0 & \sigma_n & 0 \\ 0 & 0 & 0 & \sigma_n \end{bmatrix} \xrightarrow[cycles]{4*2+3} \begin{bmatrix} h & h & h & h \\ 0 & h & h & h \\ 0 & h & h & h \\ 0 & h & h & h \\ 0 & h & h & h \\ 0 & \sigma_n & 0 & 0 \\ 0 & 0 & \sigma_n & 0 \\ 0 & 0 & 0 & \sigma_n \end{bmatrix}$$

$$\xrightarrow[cycles]{4*2+2} \begin{bmatrix} h & h & h & h \\ 0 & h & h & h \\ 0 & 0 & h & h \\ 0 & 0 & h & h \\ 0 & 0 & h & h \\ 0 & 0 & h & h \\ 0 & 0 & \sigma_n & 0 \\ 0 & 0 & 0 & \sigma_n \end{bmatrix} \xrightarrow[cycles]{4*2+1} \begin{bmatrix} h & h & h & h \\ 0 & h & h & h \\ 0 & 0 & h & h \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & \sigma_n \end{bmatrix} \xrightarrow[cycles]{4*2} \begin{bmatrix} h & h & h & h \\ 0 & h & h & h \\ 0 & 0 & h & h \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The actual sorting of columns of matrix $\bar{\mathbf{H}}$ takes 2 cycles (one for the search for the column with the smallest norm, and one for the actual column swap). However, the number of swaps is variable, as it could possibly occur, that the columns of $\bar{\mathbf{H}}$ are initially in the proper order. This leads to the variable number of cycles for the whole QRD, between 55 cycles in absence of swaps and 58 cycles if all three possible swaps were required.

TABLE 4.1: Implementation using multicycle constraints

Multi-cycles	Cycles per rotation	Cycles per QRD	STRATIX III		STRATIX V	
			F_{max} [MHz]	Execution time [μs]	F_{max} [MHz]	Execution time [μs]
0	2	55 - 58	37.54	1.54*	49.17	1.17*
1	3	71 - 74	67.47	1.09*	98.9	0.75**
2	4	88 - 91	74.28	1.22*	102.99	0.88**
3	5	103 - 106	80.69	1.31*	99.81	1.06*

TABLE 4.2: Implementation using register stages

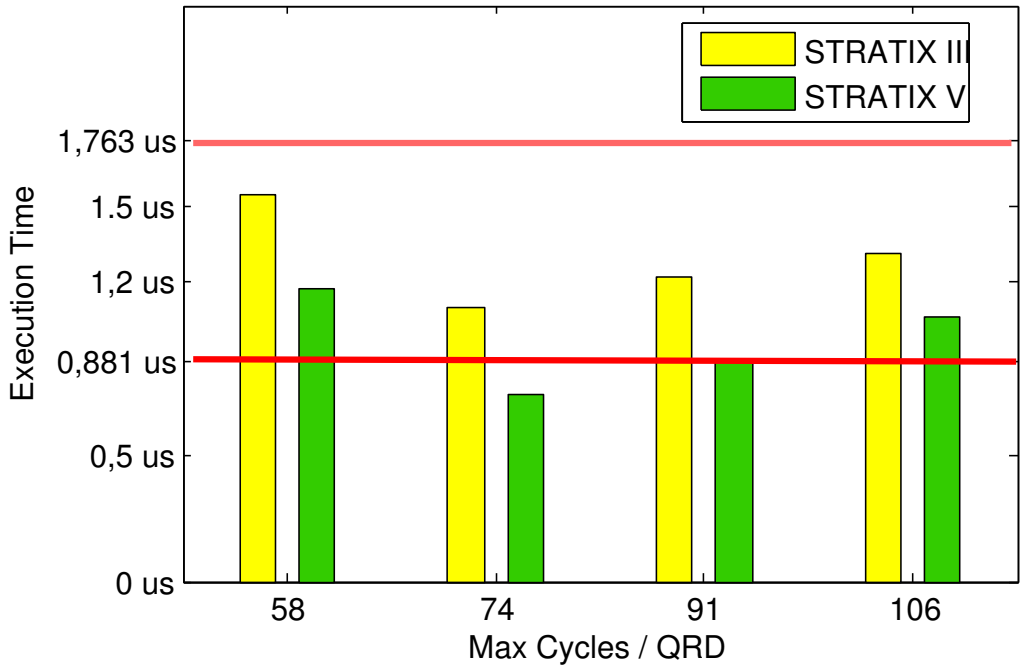
Register stages	Cycles per rotation	Cycles per QRD	STRATIX III		STRATIX V	
			F_{max} [MHz]	Execution time [μs]	F_{max} [MHz]	Execution time [μs]
0	2	55 - 58	37.54	1.54*	49.17	1.17*
2	4	88 - 91	79.6	1.14*	106.44	0.85**
3	5	103 - 106	77.39	1.37*	122.96	0.86**

4.2.1.3 Synthesis results

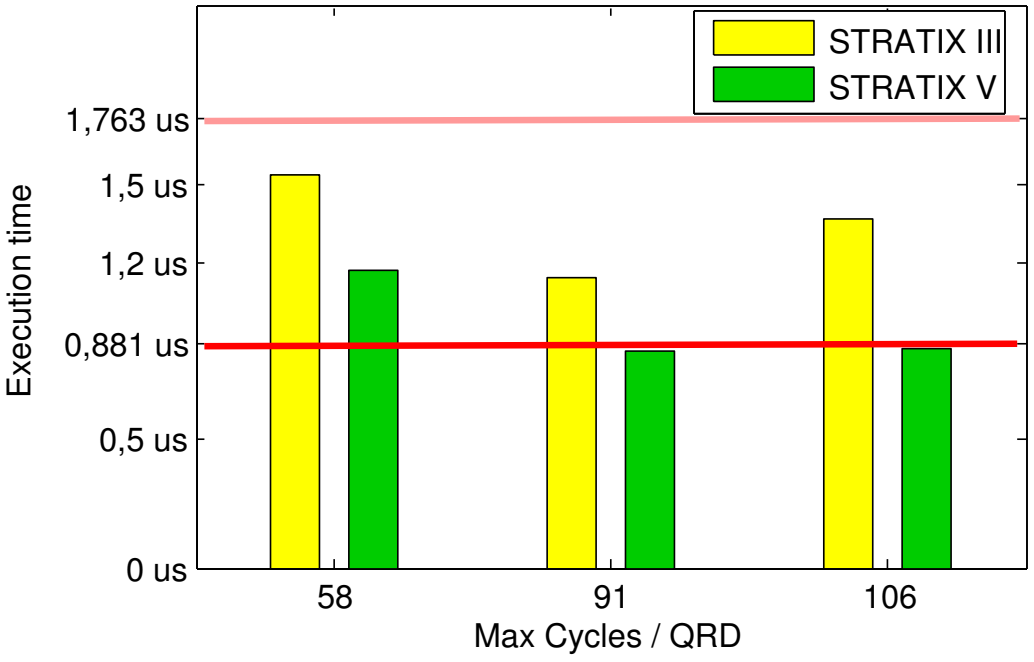
The architecture has been synthesized for Stratix III FPGA (EP3SL70F780C2) and Stratix V FPGA (5SEE9H40C2). These FPGAs are manufactured with 65 nm and 28 nm technology processes respectively. The ultimate goal of the implementation is the achievement of real-time requirements derived from the LTE standard, 1.763 μs or 0.881 μs for the complete QRD, depending on the maximal allowed relative receiver speed. The time required for the QRD is the product of the number of cycles and the duration of one clock cycle. The time-critical operation of QRD is the boundary cell of the systolic array. Two speed-up techniques have been applied to that cell: multi-cycle and register insertion. Both techniques increase the number of cycles per rotation (and therefore for the entire QRD) but significantly decrease the cycle time.

The synthesis results are summarized in Tables 4.1 and 4.2. The reported clock frequency assumes the fast process corner and temperature of 0° C. The execution times that fulfill the real-time requirement of 1.763 μs and of 0.881 μs are marked by one asterisk (*) and two asterisks (**), respectively. Figures 4.8a and 4.8b visualize the execution times; both real-time constraints are shown by horizontal lines.

The basic design without optimizations fulfills the lower real-time requirement, and the stricter requirement can be achieved using multi-cycles or register stages. However, their number should be low, since increasing this number beyond a certain threshold leads to too many clock cycles, an increase that cannot be compensated by clock frequency improvement. The extent of execution-time reduction is similar for both optimization techniques. Note that multi-cycles are inserted automatically by the synthesis tool, whereas registers are placed manually by the designer.



(A) Multicycle constraints



(B) Register stages

FIGURE 4.8: Timing results

TABLE 4.3: Area results

FPGA	Combinational ALUTs(%)	Dedicated registers(%)	Total Pins(%)	DSP blocks(%)
STRATIX III	46	3	84	65
STRATIX V	5	<1	78	42

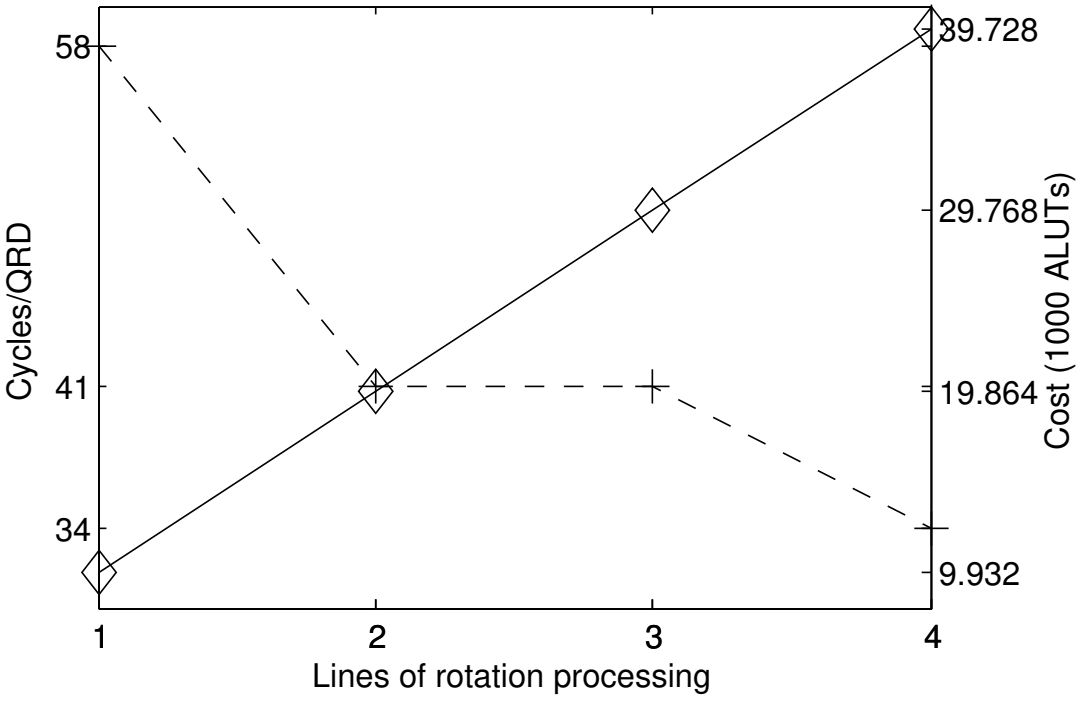


FIGURE 4.9: Cycles/QRD vs cost

To assess the speed gain of the hardware architecture, time measurements of respective software implementation have been performed, measuring roughly 5.4 milliseconds for a single QRD. This is orders of magnitude slower than the hardware implementation which works within a few microseconds. The area results for both FPGA families are presented in Table 4.3. The circuit easily fits onto the respective FPGA.

In case of stricter execution time constraints, the design is easily scalable, by adding processing lines to the architecture. Each rotation, or operation on two rows requires one processing line. One additional processing line allows two nulling operations in one cycle, with linear area cost increase. Figure 4.9 shows the cost and the number of cycles required for the complete QRD depending on number of rotation processing lines.

4.3 Reliability analysis of combinational components

The reliability of individual computational sub-modules of implemented MMSE sorted QRD is assessed by gate level FPGA-based fault injection.

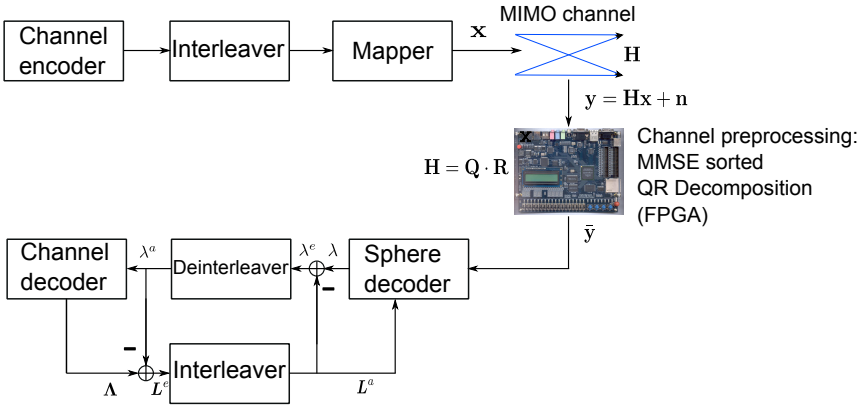


FIGURE 4.10: Simulation chain

4.3.1 Simulation chain

Since virtual massive MIMO hardware is not yet available, the closed loop small-scale MIMO system is used in simulation. The simulation chain of MIMO transmission and reception system is depicted in Fig. 4.10. The channel encoder adds redundant bits to the information word (same rate 1/2 convolutional encoder is used). The interleaver reduces dependencies between adjacent codeword bits. Next, the encoded word is 16-QAM modulated, and finally a vector $\mathbf{x}$ of M_t symbols is formed and transmitted over M_t antennas at the same time and in the same frequency. The MIMO channel is characterized by the channel matrix $\mathbf{H}$ which contains gains associated with individual transmission paths. The size of the channel matrix is $M_r \times M_t$, where M_r is the number of receive antennas. The received symbol vector $\mathbf{y}$ is defined in Eq. 2.61.

The channel preprocessing block performs MMSE sorted QR decomposition on the channel matrix $\mathbf{H}$. The result of the decomposition is the upper triangular matrix $\mathbf{R}$ and unitary matrix $\mathbf{Q}$. Equation 2.61 is then transformed to Eq. 2.114 by unitary matrix $\mathbf{Q}^H$.

The resulting triangular system of equations is then mapped to a tree search performed by the sphere decoder. The soft-input soft-output SD determines the likelihood of the bits demodulated from received vector $\bar{\mathbf{y}}$ using the a priori information $\mathbf{L}^a$ from the channel decoder. Only the extrinsic information $\lambda^e = \lambda - \mathbf{L}^a$ is passed on to the soft Viterbi decoder. The channel decoder uses the a priori information λ^a from the sphere decoder for calculation of the estimated codeword and the a posteriori log-likelihood ratios $\mathbf{\Lambda}$ of the codeword. The extrinsic information $\mathbf{L}^e = \mathbf{\Lambda} - \lambda^a$ is then returned to the SD, closing the loop. As the channel decoder itself is operating with two iterations, the MIMO receiver represents a doubly iterative architecture.

The system level performance is characterized by FER at the the output of the channel decoder. The channel preprocessing block is implemented on an FPGA, allowing access to individual gates of the design for fault injection. The remaining blocks of the simulation chain are implemented in software (IT++ library [165]), using the same fixed point number format and arithmetic as the hardware component.

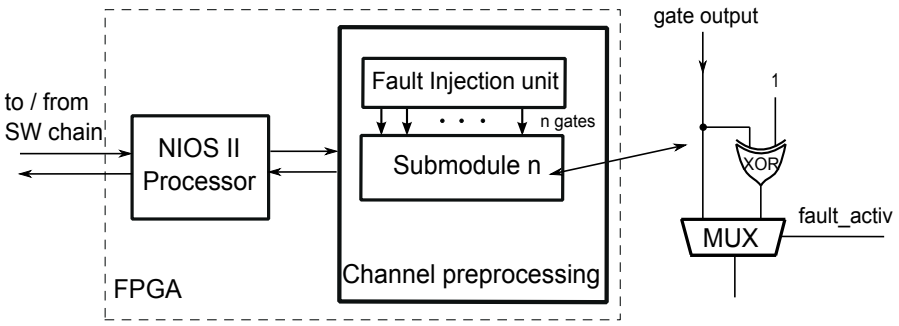


FIGURE 4.11: Fault injection architecture

4.3.2 Fault injection

The hardware errors in various sub-modules of channel preprocessing are assumed to be the effects of transient faults at the outputs of the gates comprising the netlist. The transient faults are simulated by fault injection similar to the work in [166].

First, the injection requires modification of the HDL (hardware description language) design files to include redundant multiplexers and exclusive-or (XOR) gates as shown in Fig. 4.11. The XOR gate flips the value of the gate output, and the multiplexer passes the flipped value further, when the **fault_activ** control signal is high. After the files are automatically modified, the fault injection unit is connected to the respective sub-module of the channel preprocessing block. The NIOS II processor receives the channel matrix input data from the software part of the simulation chain and passes it to the MMSE sorted QRD. After the decomposition is done, it sends the **R** and **Q** matrices back to the software part of the simulation chain. It also sets the value of the fault injection rate.

The architecture of the fault injection unit is depicted in Fig. 4.12. It takes as the input a stream of $m \cdot 64$-bit random numbers and outputs vectors of fault injection stimuli with a configurable rate of ones. First, the random number inputs are fed into m $64:6$ Priority Encoders. As the received random number stream can be assumed to be uniformly distributed, outputs of the Priority Encoders correspond to certain probabilities.

Controlled via the parameter port **rate** outputs of the Priority Encoders are used to generate in parallel m bits of a bit-stream with a certain rate of ones by the Rate Adjustment block. This stream is then fed into an n-bit shift register that receives m bits per clock cycle. The output of the shift register is connected to the **fault_activ** ports of the sub-module under injection. The fault injection unit allows generation of ten distinct transient fault rates ψ, from 10^{-12} to 10^{-4}.

4.3.3 Simulation results

Consider the channel preprocessing architecture depicted in Fig. 4.5. The *INNER PRODUCT*, *NORM VECTOR* and *NORM UPDATE* blocks are required by sorting. Sorting does not improve the FER performance, it reduces the complexity of the tree search performed by the SD. Thus, the hardware errors in these blocks would not affect

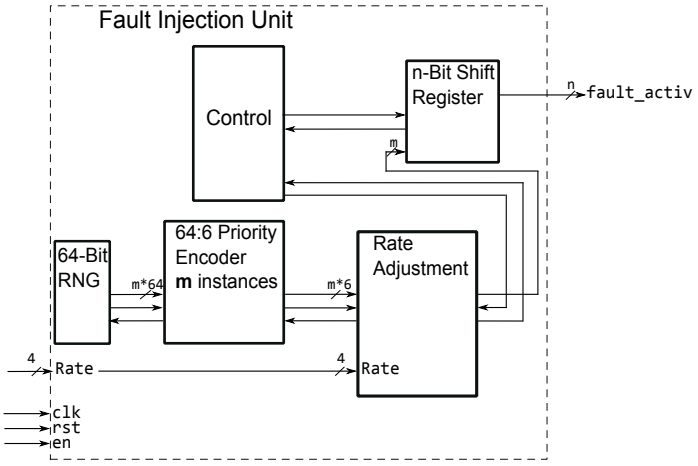


FIGURE 4.12: Fault injection controller

the FER, but would increase the number of tree passes, as the search would be started from the suboptimal starting point. This complexity increase can be quantified, but it is out of scope of this work. The errors in the internal memory can be countered by standard protection techniques [167]. In general, protecting only integer parts of the stored fixed point numbers may be sufficient.

The boundary and internal cells are the components actually performing the computation of the resulting $\mathbf{Q}$ and $\mathbf{R}$ matrices. Hence, the hardware errors within these combinational blocks will definitely affect the FER. The impact of hardware errors within the boundary cell outweighs the impact of hardware errors within internal cells. This is due to the fact that the $\sin\varphi$ and $\cos\varphi$ values computed by the boundary cell are propagated to internal cells. Wrong $\sin\varphi$ and $\cos\varphi$ values will cause error propagation and yield further computation of the rotation completely wrong.

FER results for faults injected at random locations of the boundary cell and random internal cells, are shown in Fig. 4.13. For all fault injection rates the FER performance is shown after the second iteration of the SD.

Starting from the SNR value of 10 dB the hardware errors start to outweigh the errors introduced by the channel. It can be observed that in this high SNR region, for low fault injection rates the FER performance is affected more by the errors within the boundary cell. For fault injection rate of 10^{-9}, for instance, FER for faults injected in the boundary cell is almost one order of magnitude worse than the FER for faults injected into the internal cells for all SNR values. As the fault injection rate increases, it can be observed that the location of hardware errors has less and less impact on the overall FER performance. It can be observed that faults injected into boundary cell introduce the FER floor of 10^{-3} for fault injection rate of 10^{-9}, FER floor of 10^{-2} for fault injection rate of 10^{-6} and FER floor of 10^{-1} for fault injection rate of 10^{-5}.

Consider the architecture of the boundary cell, depicted in Fig. 4.6. The upper four real valued multipliers and three real valued adders are computing the 2-norm squared of the rotated vector according to line 4 of Alg. 9. Next, the inverse square root is computed. The left-side real valued multiplier obtains the main diagonal element of the $\mathbf{R}$ matrix by multiplying the result of the inverse square root module by the 2-norm squared. The

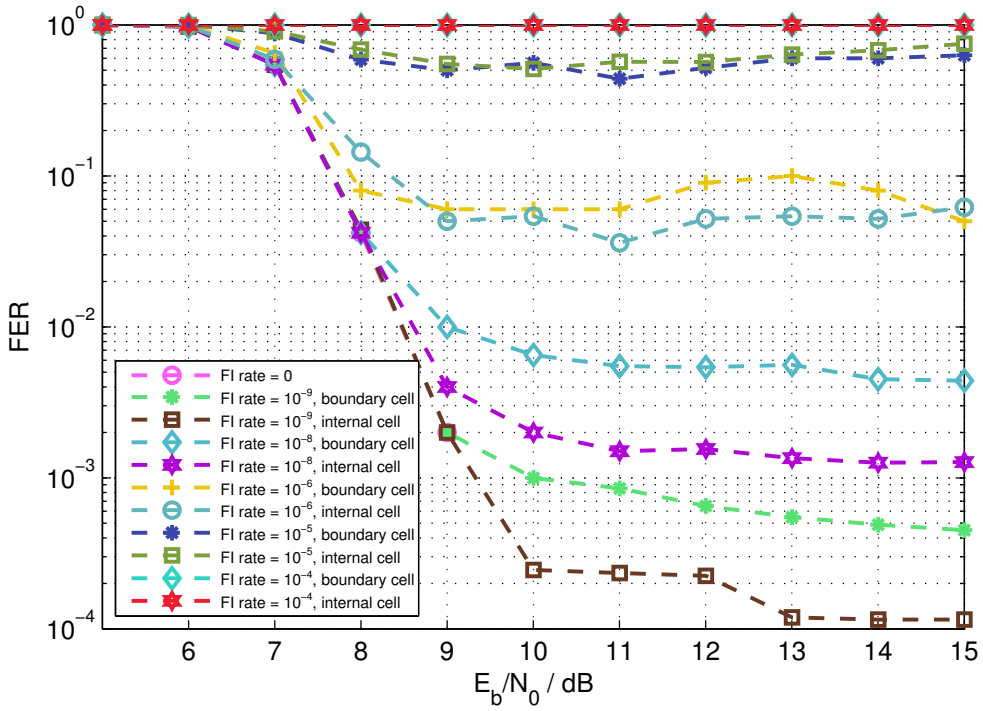


FIGURE 4.13: Boundary cell and internal cell under fault injection

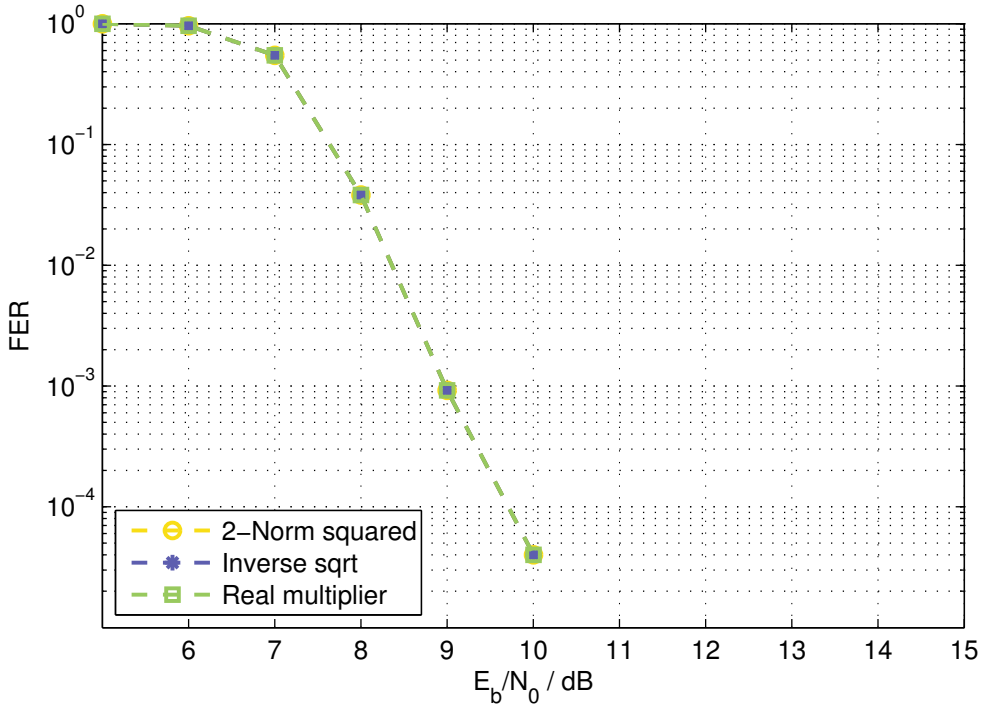


FIGURE 4.14: Fault injection, $\psi = 10^{-10}$

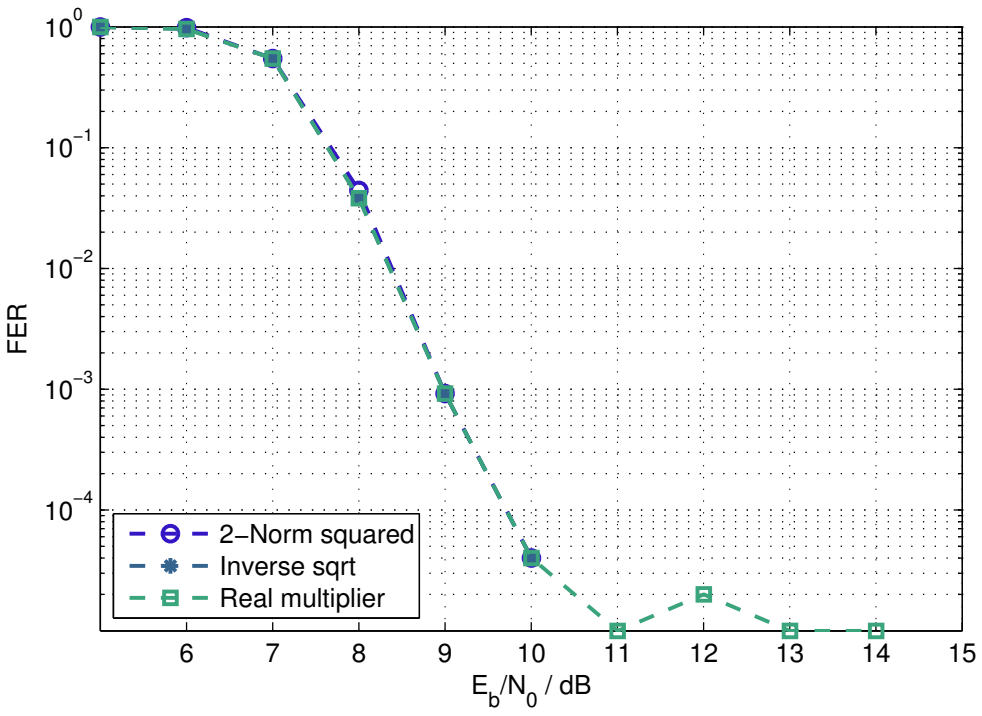


FIGURE 4.15: Fault injection, $\psi = 10^{-9}$

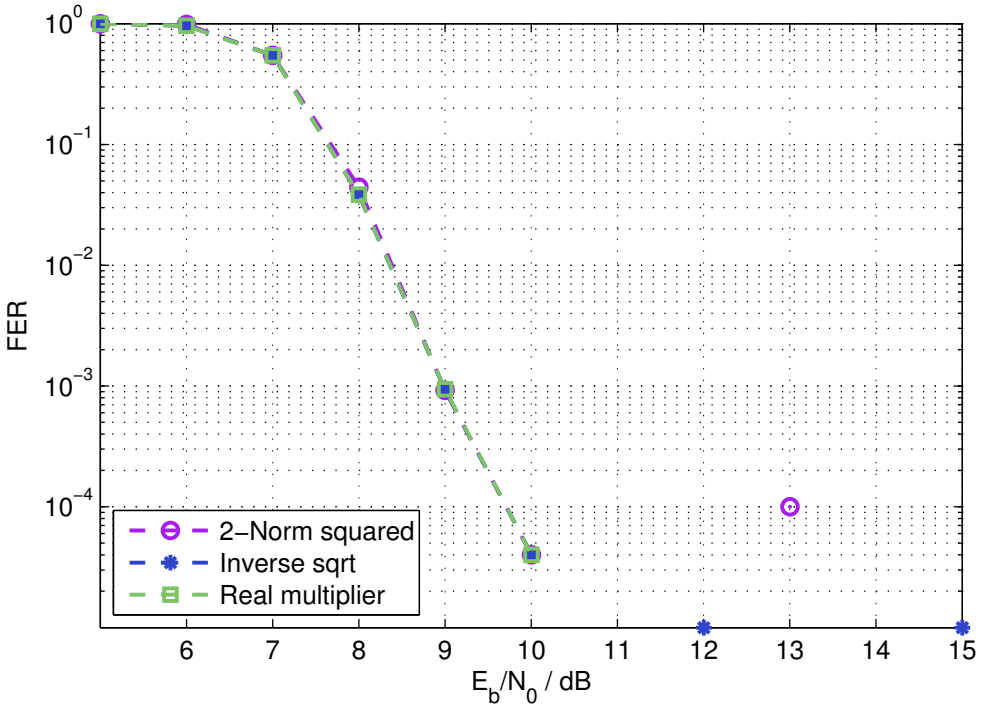


FIGURE 4.16: Fault injection, $\psi = 10^{-8}$

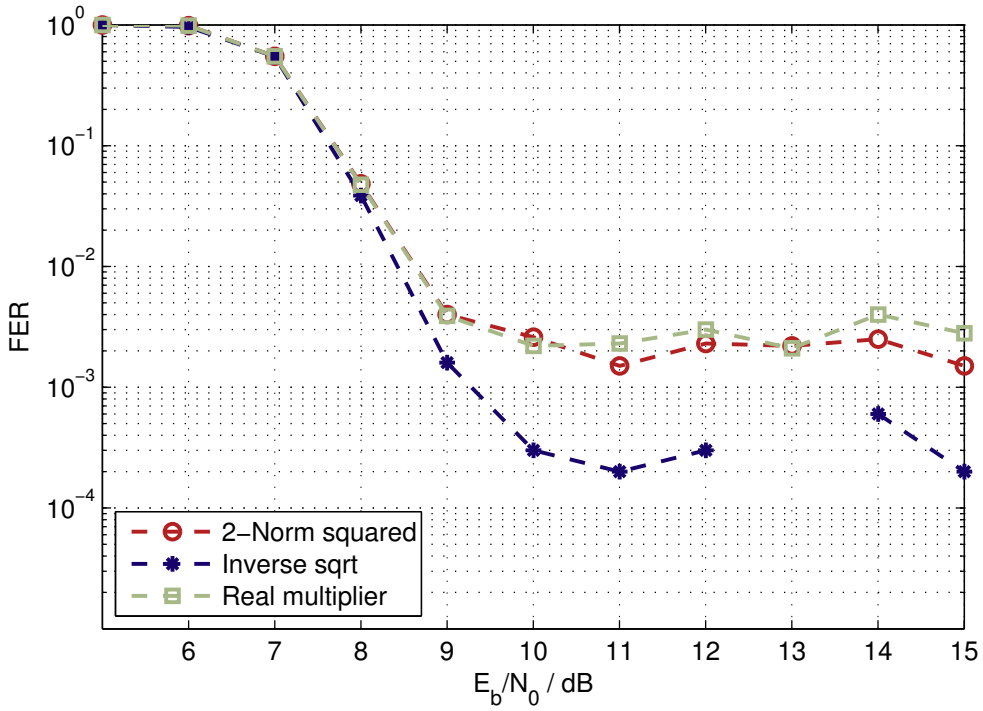


FIGURE 4.17: Fault injection, $\psi = 10^{-7}$

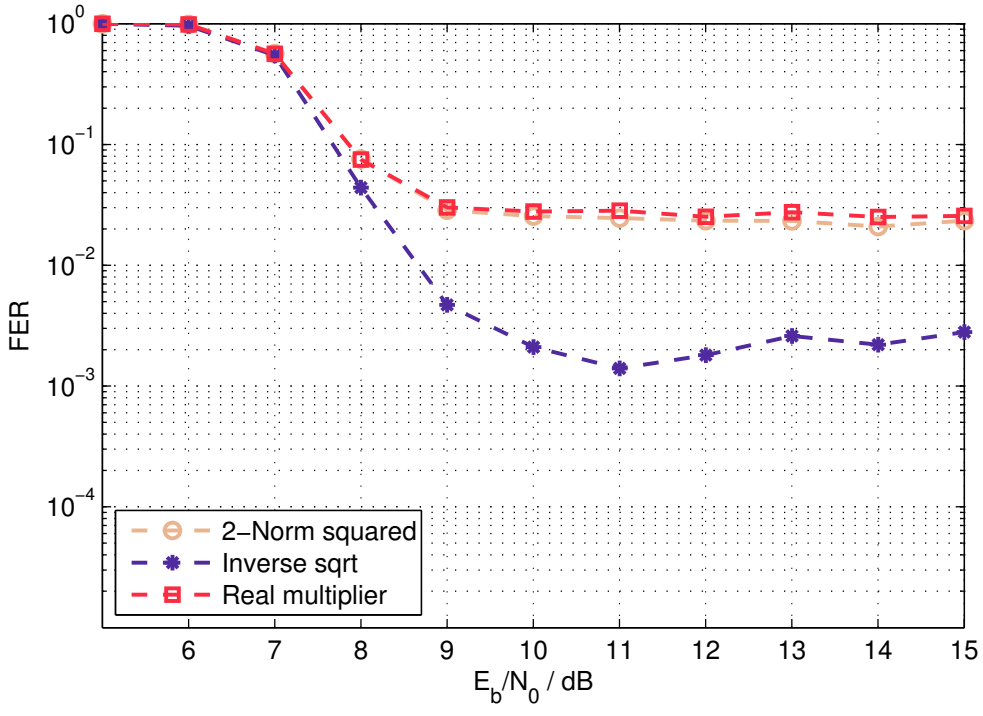


FIGURE 4.18: Fault injection, $\psi = 10^{-6}$

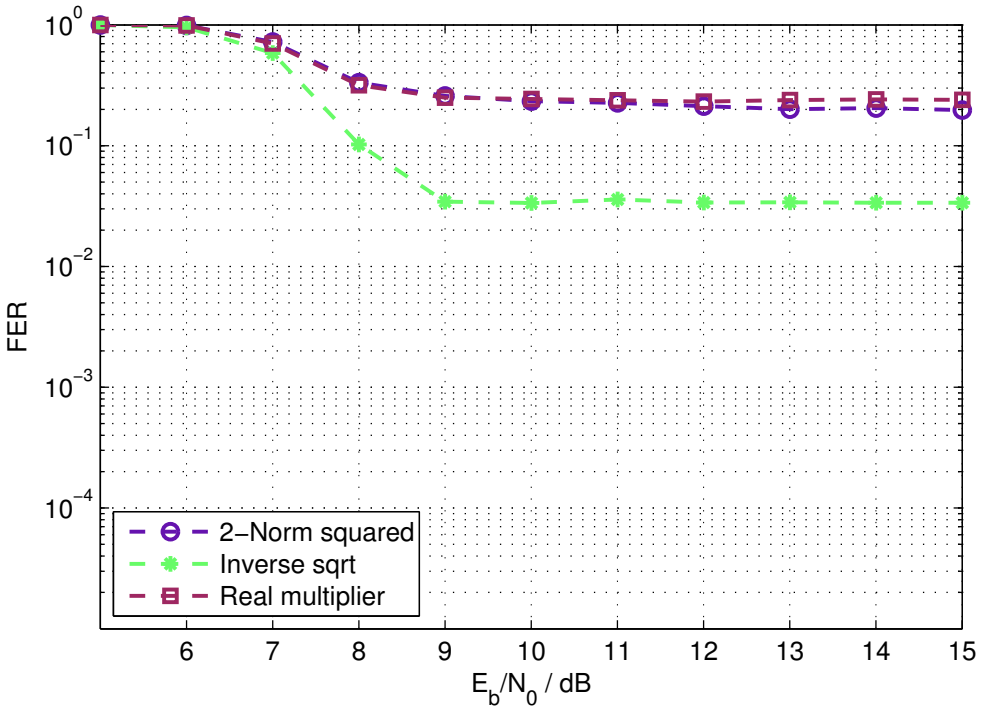


FIGURE 4.19: Fault injection, $\psi = 10^{-5}$

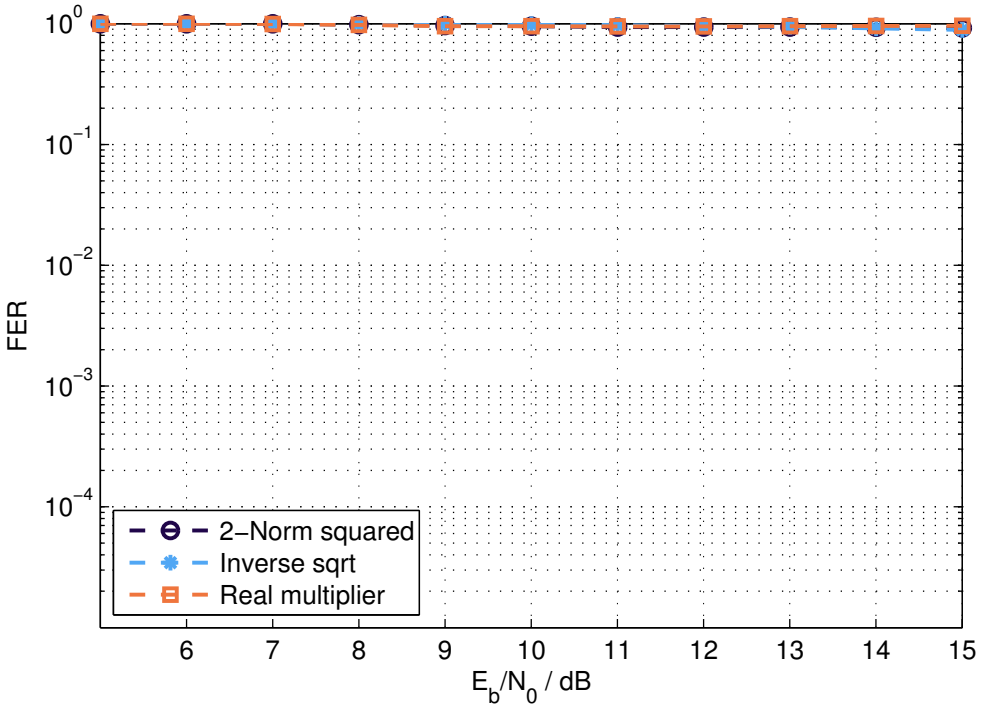


FIGURE 4.20: Fault injection, $\psi = 10^{-4}$

hardware errors in that real valued multiplier would affect only the computation of the main diagonal element. The hardware errors within the inverse square root block and the 2-norm squared calculation will propagate to both the new main diagonal value and the $\sin\varphi$ and $\cos\varphi$ values. The errors that occur in the four right-side real valued multipliers only affect the $\sin\varphi$ and $\cos\varphi$ values. The FER results when faults are injected into 2-norm squared, inverse square root and real multiplier hardware are depicted in Figures 4.14 to 4.20.

It can be observed that FER is less affected by the errors occurring in the inverse square root block. For fault injection rate from 10^{-7} and to 10^{-5}, the FER floor for faults injected in the inverse square root is almost one order of magnitude lower than the FER floors introduced by the errors within real multiplier and 2-norm squared. The computation performed by the boundary cell is iterative. The newly computed diagonal element is fed back to the boundary cell, for computing the next rotation. The 2-norm squared and real multiplier are all taking part in computing the value of the diagonal element, a hardware error within any one of them would render subsequent rotations wrong. The inverse square root is also used when computing new diagonal element value and it also computes the values of $\sin\varphi$ and $\cos\varphi$. The results indicate that wrong new diagonal value, due to errors within the real multiplier, affects the FER more than wrong both new diagonal element and the $\sin\varphi$ and $\cos\varphi$ values, caused by errors in the square root computation. The same holds for wrong both new diagonal element and the $\sin\varphi$ and $\cos\varphi$ values due to errors in the 2-norm squared.

4.4 Summary

This chapter considered a new architecture for MMSE sorted QR decomposition which meets the real-time constraints of the LTE standard. The architecture is based on Givens rotations and incorporates circuitry that implements sorting. The high speed is achieved by applying design optimizations (multicycles or register stage insertion) on the base architecture. Higher speed can be easily achieved at the cost of area.

A fault injection based reliability analysis of computational components of implemented MMSE sorted QRD is performed. The most vulnerable block within the channel pre-processing is identified as the boundary cell. Errors occurring in the boundary cell have more impact on the FER performance than errors occurring within the internal cells' hardware. This FER degradation is quantified as a FER floor in the high SNR region.

Within the boundary cell, the inverse square root turns out to be the least critical. The real multiplier has more impact, even as it only computes the new diagonal value, whereas the 2-norm squared and inverse square root affect both the new diagonal element and $\sin\varphi$ and $\cos\varphi$ values. The experimental results show that due to the iterative function of the boundary cell, the incorrect value of new diagonal is impacting the FER performance stronger than incorrectly computed $\sin\varphi$ and $\cos\varphi$ values.

Therefore, the fault tolerant implementation of the boundary cell must specifically address the protection of the new diagonal element computation. As the errors in the latter rotations are less critical, it can be considered to provide varying amount of protection to first and subsequent iterations of the boundary cell.

Chapter 5

Evaluation of robust codes

Traditional methods to protect circuits against faults, such as triple modular redundancy [5], are impractical in most instances as they incur unacceptable area and, more importantly, power overhead. Error-detecting codes (EDC) [168] are typically associated with lower cost but are effective only against well-specified errors. For example, the parity code applied to a circuit's outputs cannot detect a single fault within the circuit which propagated to an even number of outputs. The interest in effectiveness of EDCs recently lead to the development of robust codes specifically optimized for malicious fault injection (rather than random faults due to radiation and noise) [169, 170]. This chapter starts with the background on robust codes, which have been developed for security applications. Then, the performance of these codes is investigated when applied to protection of general purpose circuits. Since the performance of robust codes has been evaluated based on information-theoretic measures, this chapter first introduces the circuit or fault-based measures that capture the input–fault–error relationship. Next, a cross-level protection scheme is introduced which identifies faults that escape detection and targets them by selective hardening. Finally, robust codes are applied to an actual cryptographic circuit and their performance is quantified in terms of introduced fault-based metrics.

5.1 Robust codes

The general error detecting architecture is depicted in Fig. 5.1 [5, 168]. The k output bits of the original device are denoted by $y(x)$. The input vector x of the device is concurrently applied to the inputs of a combinational predictor, which generates the r redundant bits (RB) $p(x)$. A further combinational block EDN (error-detection network) evaluates the consistency of the original device's outputs $y(x)$ and the check bits $p(x)$. More formally, EDN checks whether the pair $y(x).p(x)$ is a codeword c of a predefined code $\mathcal{C}$.

The design of error detecting codes against fault-based attacks is based on the assumption [169] that injected faults manifest themselves as additive errors at the output, i.e., in a presence of a fault a fault-free output vector c becomes $y = c \oplus e$, where e is an error vector. There are three types of errors that are also illustrated in Fig. 5.2:

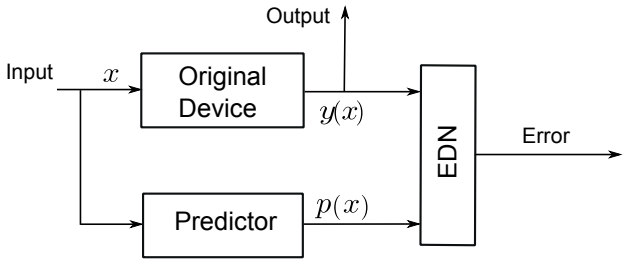


FIGURE 5.1: Error detecting architecture

- Errors that are always detected: $\forall\, c \in \mathcal{C},\ c \oplus e \notin \mathcal{C}$.
- Errors that are never detected: $\forall\, c \in \mathcal{C},\ c \oplus e \in \mathcal{C}$.
- Errors that may be detected or not detected depending on the codeword: $\exists\, c_a, c_b \in \mathcal{C}$ such that $c_a \oplus e \notin \mathcal{C}$ but $c_b \oplus e \in \mathcal{C}$.

The set that contains all the errors that are not detected by any codeword is called the kernel of the code, and is denoted by K_d. The *error-masking probability* $Q(e)$ is defined as the ratio between the number of codewords that are masked in the presence of the error e, and the total number of codewords:

$$Q(e) = \frac{\mid \{c \mid c \in \mathcal{C} \text{ and } c \oplus e \in \mathcal{C}\} \mid}{\mid \mathcal{C} \mid} \tag{5.1}$$

This definition implicitly assumes that the output of a circuit is uniformly distributed. The maximal error masking probability of a code is defined as

$$Q_{\text{mc}} = \max_{e \notin K_d} Q(e) \tag{5.2}$$

Robust codes are EDC that detect all nonzero errors with some nonzero probability. More formally, they satisfy the condition

$$Q(e) < 1 \text{ for all } e \neq 0 \tag{5.3}$$

(This definition is equivalent to $K_d = \{0\}$.) For a given k and r, the maximal error masking probability is lower bounded by $Q_{\text{mc}} \geq \{2^{-k+1}, 2^{-r}\}$ [171]. A code that meets this bound is called an optimal code.

The rate of a code is denoted by $R = \frac{k}{k+r}$. Most robust codes are of relatively small rate [172–174]. There are two codes with rate greater than 0.5, the quadratic sum (QS) code [175] and the punctured cubic (PC) code [176]. Before describing these codes, some notations are introduced.

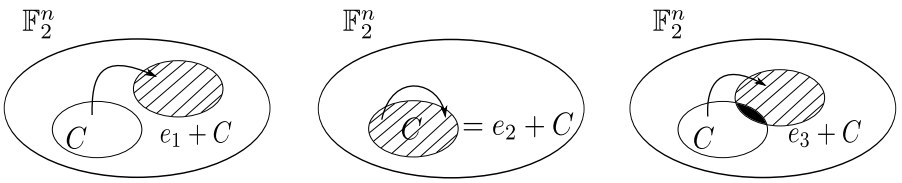


FIGURE 5.2: Possible detection status of errors at the input of the EDN in Fig. 5.1

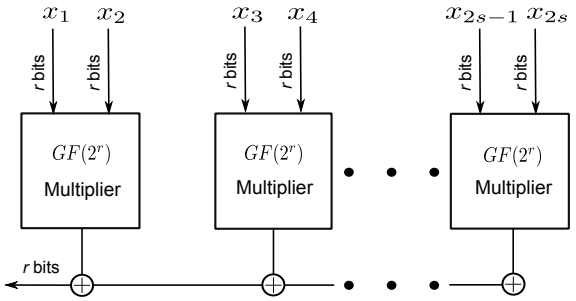


FIGURE 5.3: General architecture for EDN of the QS code

A binary vector can be referred to as an element in a finite field. For example, a k-bit vector can be regarded as a vector in k-dimensional space $\mathbb{F}_2^k$ with ($\mathbb{F}_2 = GF(2)$), but can also be referred to as an element of the finite field $\mathbb{F}_{2^k} = GF(2^k)$. The following constructions use both representations. For example, the expression x^3P where P is a binary $k \times r$ matrix, should be read as: refer to x as an element in $\mathbb{F}_{2^k}$ and compute x^3, then refer to the result as a vector in $\mathbb{F}_2^k$ and multiply it by the matrix P modulo 2; the outcome of this operation is an element in $\mathbb{F}_2^r$. Note that addition of two elements of $\mathbb{F}_2^k$ means coordinate-wise addition modulo 2 (XOR).

5.1.1 Quadratic sum code

The quadratic sum code [175] is defined by $\mathcal{C} = \{(x, w) : x = (x_1, \dots x_{2s}) \in \mathbb{F}_{2^k}, x_i \in \mathbb{F}_{2^r}, w \in \mathbb{F}_{2^r}, w = x_1x_2 \oplus x_3x_4 \oplus \dots \oplus \dots x_{2s-1}x_{2s}\}$. It can be shown that for $k = 2sr$ and $s > 0$, the maximal error-masking probability of the QS code is $Q_{\text{mc}} = 2^{-r}$. As a result, the QS code is an optimal robust code for $k = 2sr$ and $s > 0$. The minimal distance of the QS code is 1, therefore, this code has no error correction capabilities.

Let $0 \neq l \leq 2r$. A code of dimension $k = 2sr - l$ is called a shortened QS (SQS) code. A SQS code is constructed by appending l zeros to the information word (These extra zeros are only used for calculating redundant bits and are not explicitly included into the word). A code designer can choose where to place the l additional zeros. This choice determines the value of Q_{mc}. If x_i and x_{i+1} (where i is odd) are both appended with zeros, then the resulted code is not optimal. If an entire x_i is appended with zeros, the resulted code is not robust. The error masking probability of a SQS code is $Q_{\text{mc}} \geq \max(2^{\lceil \frac{l}{2s} \rceil}2^{-r}, 2^{-\lfloor \frac{k}{2} \rfloor})$ [177]. As a result, there are pairs of k and r for which no optimal SQS code can be constructed.

The architecture of the EDN for the QS code has two parts. The first part consists of s finite filed multipliers in GF(2^r) [178] (in the simplest case of GF(2), these are AND2 gates), and the second part is a XOR network that XORs of the results of the multipliers. This is illustrated in Fig. 5.3.

5.1.2 Punctured cubic code

The construction of the punctured cubic code [176] is based on a binary $k \times r$ matrix P of rank $r \leq k$. The code is then defined by $\mathcal{C} = \{(x, w) : x \in F_{2^k}, w = x^3P \in \mathbb{F}_{2^r}\}$.

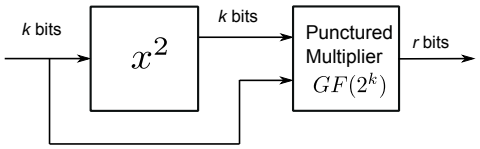


FIGURE 5.4: General architecture for EDN of the PC code

The maximal error masking probability of the PC code is $Q_{\text{mc}} \leq 2^{-r+1}$ [179]. If k is even and r is small enough, then it is possible to find matrix P such that the resulting PC code has $Q_{\text{mc}} = 2^{-r}$ and the resulted code is optimal. In other cases, if r is large enough it is possible to construct PC code with minimal distance $d \geq 2$ [177].

For example, for $k = 10, r = 4$ the SQS code is not optimal but there is an optimal PC codes for these parameters. On the other hand, for $k = 10, r = 5$ the QS code is optimal but there is no optimal PC code for these parameters.

The architecture of the EDN of the PC code consists of two modules. The first module computes $z = x^2$, and the second computes $(x * z)P$. The easiest way to implement the PC code is by choosing $P = [I|0]^T$. In this case, a multiplication by P is done by simply disregarding some of the outputs of $(x * z)$. As a result, in order to calculate $(x * z)P$, it is sufficient to implement only part of the multiplier. Such a multiplier is called a punctured multiplier. This is illustrated in Fig. 5.4. In this work, the PC code is implemented for $k = 64$. A simple implementation is chosen using $P = [0|I]$, for $r = 1, 2, 3, 4$. For example, for $r = 1$, the redundancy is the LSB of x^3.

5.2 Evaluation methodology

A key difficulty in transferring theoretical properties of codes to faults in actual circuits is the difference between additive errors on the information-theoretical level (discussed above) and faults in actual circuit structures. This requires specific metrics [46] which are explained next.

5.2.1 Fault-based metrics

When a fault occurs in a circuit equipped by an error-detecting architecture, four distinct outcomes are possible:

- Masked fault: A fault in the device does not affect $y(x)$ and $Err = 0$. For example, the fault-affected logic gate may drive a gate with a controlling value on its side input.
- Detected fault: A fault affects $y(x)$ and $Err = 1$. This happens when $y(x)$ becomes inconsistent with $p(x)$ due to the fault and the EDN identifies that $y(x).p(x)$ is a non-codeword.
- Silent data corruption (SDC) [180]: A fault affects $y(x)$ and is not detected: $Err = 0$. This may happen if the fault-induced changes to $y(x)$ and/or $p(x)$ still result in $y(x).p(x)$ being a codeword, or when a fault affects the EDN itself.

- False alarm: A fault does not affect $y(x)$ but error detection is reported: $Err = 1$. This scenario is possible for errors in the EDN and for faults in the predictor which influence $p(x)$.

Obviously, the faults which could critically influence the circuit operation are of concern. This is not the case for masked faults or false alarms, even though the latter might trigger unnecessary recovery. Moreover, detected faults are considered uncritical as the system is aware of the fault and can initiate corrective measures. In contrast, SDC is considered critical. The definition of SDC does not distinguish between root causes of a fault. The faults could be induced by random noise phenomena or introduced by a malicious adversary who is interested in the fault-based attack being undetected.

It is to distinguish between the notions of a fault and an error. A fault f affects internal logic gates and lines of the circuit. Single and multiple stuck-at faults on logic gate outputs within the architecture (device, predictor and EDN) are considered. In contrast, error $e(f)$ is the effect of the fault on the device's output y. If the fault-free response of the device to input x is $y(x)$ and the response of the device affected by fault f is $y_f(x)$, then the error $e(f(x))$ is defined as $e(f(x)) := y(x) \oplus y_f(x)$. It is important that the error depends not only on the fault but also on the input vector applied. The same fault could be masked, detected or undetected depending on the input vector. Therefore, the efficiency of error detecting code in the context of fault detection cannot be calculated analytically but rather has to be evaluated by simulations of the actual circuit.

Let the set of possible input vectors be I. Let the set of input vectors for which a fault f is detected be $I_{\mathrm{DET}}(f)$, and let the set of input vectors for which it results in an SDC be $I_{\mathrm{SDC}}(f)$. Let the sizes of the sets (the number of the respective input vectors) be $|I|$, $|I_{\mathrm{DET}}(f)|$ and $|I_{\mathrm{SDC}}(f)|$, respectively. In context of random transient faults, the silent data corruption scenario is considered critical, and detected faults as well as faults with no effect can be excluded from further consideration [181]. Therefore, an appropriate metric with respect to reliability aspects is the *SDC* rate:

$$SDC(f) = \frac{|I_{\mathrm{SDC}}(f)|}{|I|} \tag{5.4}$$

$SDC(f) = 0$ means that fault f never goes undetected. A higher value of $SDC(f)$ means a higher chance that a silent data corruption takes place. Note that this definition implicitly assumes uniformly distributed inputs I.

From the perspective of a malicious attacker, the undetected faults are of interest. If a fault is masked, the attacker can simply repeat the attack. However, if a fault is detected, countermeasures could be invoked. For instance, the device may be deactivated or the secret key exchanged, making the attack useless. Therefore, an attacker is interested in faults that will not be detected assuming they have manifested themselves. This conditional probability is expressed by the *attack success rate*(ASR):

$$ASR(f) = \frac{|I_{\mathrm{SDC}}(f)|}{|I_{\mathrm{SDC}}(f)| + |I_{\mathrm{DET}}(f)|} \tag{5.5}$$

An $ASR(f)$ of 0 means that the attacker has no chance to inject the fault f without being detected. An $ASR(f)$ of 1 identifies particularly critical faults which are never

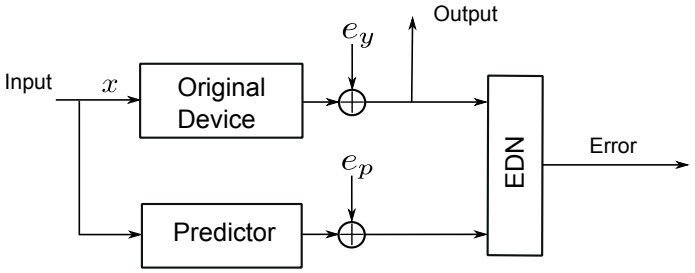
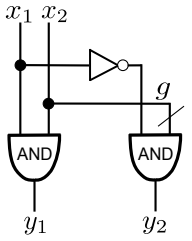


FIGURE 5.5: Additive error assumption



inputs		fault-free			g-stuck-at-0		
x_1	x_2	y_1	y_2	p	y_1	y_2	p
0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0

FIGURE 5.6: Mismatch between structural fault and additive error

detected. $ASR(f)$ is related to the classical definitions of fault-secure and self-testing circuits [182] as follows. If the circuit is fault-secure with respect to fault f, then $SDC(f)$ is empty and $ASR(f) = 0$. If the circuit is self-testing with respect to f, then at least one input vector detects f and $ASR(f) < 1$.

$ASR(f)$ obviously depends both on the fault f and on the employed EDC. Previous work [169, 170] analyzed the detection properties of various codes based on the additive-error model defined on the outputs of the original device and the predictor (Fig. 5.5). An additive error e would transform the combined output $y(x).p(x)$ into the erroneous output $y_e(x).p_e(x) = y(x).p(x) \oplus e$. Note that the additive error e is independent from the input vector x. For linear codes $\mathcal{C}$, there are additive errors $e \neq 0$ which are never detected, as for any possible output $c = y(x).p(x)$ with $c \in \mathcal{C}$, the erroneous output has the property $c \oplus e \in \mathcal{C}$ and is a codeword. For a given code $\mathcal{C}$ and an additive error $e \neq 0$, the error-masking probability Q_e (Eq. 5.1) can be viewed as the probability that error e leads to an SDC, assuming that all the codewords (output vectors) occur with uniform probability.

There are important differences between structural faults considered along with $ASR(f)$ and the additive errors from the previous works which are the foundation for $Q(e)$. For a fault f (e. g., a stuck-at fault), the induced error e_f on the outputs changes the value for different input vectors, and may even disappear ($e_f = 0$) for a considerable number of input vectors. Moreover, fault detection cannot be guaranteed even if robust codes are employed. It follows from the definition of $Q(e)$ that, for $e \neq 0$, there is at least one $c' \in \mathcal{C}$ with $c' + e \notin \mathcal{C}$. The implicit assumption is that such c' has a non-zero probability of occurrence and therefore the error can be detected. When considering actual circuits, some codewords may never show up on the circuit's outputs, and codewords that can be produced on the outputs may fail to detect error e. More generally, uniform distribution of input vectors does not always result in a uniform distribution of output vectors. Therefore, bounds on $Q(e)$ (Eq. 5.2) obtained assuming that all codewords are equally likely, may not be accurate.

For instance, consider the two-input, two-output circuit in Fig. 5.6. Assume it is protected by the QS code $\mathcal{C}$ with one redundant bit $p = y_1 y_2$ (i.e., $\mathcal{C} = \{000, 010, 100, 111\}$). This code is robust, as for each $e \neq 0$ there is a $c \in \mathcal{C}$ such that $c + e \notin \mathcal{C}$ (every additive error is detected by at least one codeword). However, injecting the single-stuck-at-0 fault at line g in the circuit results in a change of its Boolean function (see table in Fig. 5.6), while it is not detected by comparison of redundant bit p for any of the inputs (in fact, p is always 0). Therefore, the fault is never detected. Note that the additive error induced by the g-stuck-at-0 is not constant for different inputs: it is 010 for $x_1 x_2 = 01$ and 000 for all other inputs.

In summary, theoretical results on robust codes cannot be directly applied to analyzing effects of faults in the circuit's structure. Note that this observation holds only when considering faults within a circuit. It does not hold for faults in memories or communication channels for which the additive-error model is accurate.

5.2.2 Fault injection platform

The faults are simulated by fault injection. The fault injection platform is implemented on an Altera Cyclone IV FPGA. The overview of the framework is depicted in Fig. 5.7. It allows injecting stuck-at faults of arbitrary multiplicity to any logic gate of the device, the predictor and the EDN. In order to use the platform, HDL design files are automatically modified to include the redundant multiplexers required for fault injection. Fault injection takes place at a location if the control input **fault_act** of the multiplexer at that location is high. The value of injected fault (stuck-at-0 or stuck-at-1) is carried by the multiplexer input **fault_val**. The inputs to the device under injection are either fed by a pseudo-random source (an LFSR) or provided by the injection software, which runs on the NIOS II processor. The control information (fault locations and stuck-at values) are shifted in through scan chains.

The NIOS II controls the execution of the injection campaign. It computes the number of cycles the scan chain needs to be shifted, such that the fault arrives to the gate to be injected. It then passes the computed number of cycles to the scan chain controller.

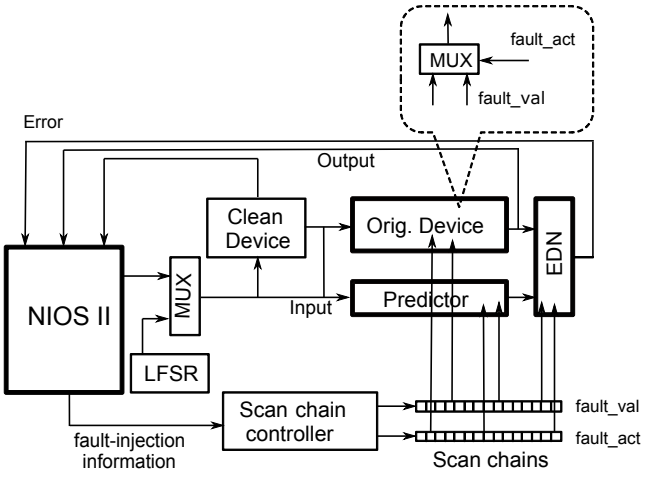


FIGURE 5.7: Fault injection framework

The framework also includes a "clean" copy of the original device without fault-injection logic, in order to identify faults which altered the device's output but did not lead to a detected error. This allows to exclude the event of a false alarm. The output of the fault-affected device, the clean device and the error status calculated by the EDN are fed back to the processor for further analysis. Basic preprocessing of the results is done on the on-chip processor, and the intermediate results are transmitted to the workstation, where desired numbers such as $ASR(f)$ for each fault are calculated.

Note that the fault injection framework is independent of the particular EDC used. Moreover, it does not require any low-level information. If some of the gates have been selected for hardening (introduced in next section), then the multiplexers performing fault injection at these gates are simply deactivated.

5.3 Cross-level protection

The objective of proposed cross-level protection is to achieve a required level of protection of a given circuit by combining error-detecting capabilities of the architecture from Fig. 5.1 with low-level hardening of insufficiently protected logic gates by eliminating the possibility of faults f with large values of $ASR(f)$. To do so, $ASR(f)$ is first computed for all modeled faults using large-scale fault-injection campaigns and order the fault list by these values. Then a number of logic gates involved in faults with large $ASR(f)$ is selected for hardening. Hardening is implemented by resizing the gate such that it becomes immune to physical disturbances such as particle strikes (in case of soft errors) or charge generation by a laser (in case of malicious attacks).

It is possible to use two alternative approaches to select the gates to be hardened:

1. Identify faults whose attack success rate exceeds some threshold and harden gates involved in these faults.
2. Allocate a budget for gates (e.g., 10%) which can be hardened and select the gates involved in faults with the largest $ASR(f)$.

For each selected gate, the resizing factor needed to render it tolerant against noise is determined.

The resulting circuit is synthesized assuming a mix of hardened and unhardened gates and its area and power overhead is estimated. The fault characterization is re-run excluding fault effects on the hardened gates: faults affecting hardened gates are masked, and multiple faults on a mixture of hardened and unhardened gates are transformed to faults of lower multiplicity on unhardened gates only.

Note that the aim is the minimization of $ASR(f)$. It could have been equally possible to minimize the probability of undetected fault occurrence $\left(\frac{|I_{\mathrm{SDC}}(f)|}{|I|}\right)$ or to maximize the fault detection probability $\left(\frac{|I_{\mathrm{DET}}(f)|}{|I|}\right)$ instead . Using a different objective would result in a selection of a different subset of gates for hardening.

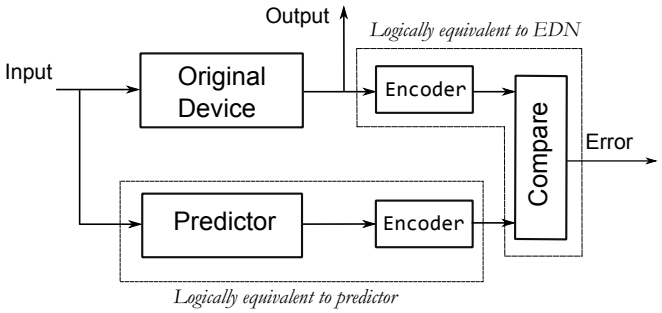


FIGURE 5.8: Implementation of error detecting architecture

5.3.1 Implementation of code-based protection

Here, four codes are investigated: linear parity code and robust QS codes over Galois fields $GF(2)$, $GF(4)$ and $GF(16)$.

For a circuit with 8 outputs $y_1, \ldots, y_8$, the linear parity code has one redundant bit

$$p = y_1 \oplus \cdots \oplus y_8$$

The QS code over $GF(2)$ has one redundant bit

$$p = y_1y_2 \oplus y_3y_4 \oplus y_5y_6 \oplus y_7y_8$$

The nonlinear parity code over $GF(4)$ has two redundant bits

$$\begin{aligned} p_1 &= y_1y_3 \oplus y_2y_4 \oplus y_5y_7 \oplus y_6y_8 \\ p_2 &= y_1y_4 \oplus y_2y_4 \oplus y_2y_3 \oplus y_5y_8 \oplus y_6y_7 \oplus y_6y_8 \end{aligned}$$

The nonlinear parity code over $GF(16)$ has four redundant bits

$$\begin{aligned} p_1 &= y_0y_4 \oplus y_1y_7 \oplus y_2y_6 \oplus y_3y_5 \\ p_2 &= y_0y_5 \oplus y_1y_4 \oplus y_1y_7 \oplus y_2y_6 \oplus y_2y_7 \oplus y_3y_5 \oplus y_3y_6 \\ p_3 &= y_0y_6 \oplus y_1y_5 \oplus y_2y_4 \oplus y_2y_7 \oplus y_3y_6 \oplus y_3y_7 \\ p_4 &= y_0y_7 \oplus y_1y_6 \oplus y_2y_5 \oplus y_3y_4 \oplus y_3y_7 \end{aligned}$$

To generate predictor and EDN from Fig. 5.1, the generic procedure outlined in Fig. 5.8 is used. The predictor is composed of a copy of the original device and the code generator (e.g., an XOR gate for the linear parity code). Similar, the EDN is created from another copy of the code generator and a bit-wise comparator. The original device, the predictor and the EDN are then, in isolation, synthesized by Synopsys Design Compiler, in order to avoid optimizations which compromise the protection. All three blocks are mapped to a sub-set of the Nangate Open Cell Library [183] which consists of INV, and 2- and 3-input NOR, and NAND gates.

It is important that the three blocks are synthesized in isolation, as synthesizing them together would result in identification and elimination of many common sub-expressions, which, in turn, would impair the protection. For instance, if the synthesis tool would process the original device and the predictor, which includes one copy of the original

device, it would simply delete this copy from the predictor and wire the output of the original device into the predictor. While such an architecture would be much more compact, it provides little protection against errors.

No special effort has been put to further optimize the predictor and the EDN beyond the simplifications done by the synthesis tool. This is because the objective is to study errors in the original device and their interaction with errors in the predictor and the EDN. An arbitrary implementation of the predictor and the EDN which is functional but not necessarily optimal is sufficient for these purposes.

5.3.2 Selective hardening of individual gates

Once the gates to be hardened have been specified, appropriate electrical parameters (transistor sizes) are identified for each of the gates such that charge deposition due to particle strike or laser pulse does not lead to a faulty signal (bitflip) at this gate's output.

Note that transistor resizing is not effective against upsets of very large energy. However, resizing can facilitate detection of such excessive-energy upsets. Since the size of the transistors is always increased, there is no delay overhead but power overhead.

The upsets only at gate outputs are considered because disturbances of gate-internal nodes must propagate through one or multiple transistors before having any visible effect on the gate outputs. In general, the possibility of a bit flip depends on the following factors [184]:

1. Total charge deposited at the node at which the particle strike occurs.
2. The drive strength of gate that drives the node.
3. Total capacitance of the node which includes the fanin and the fanout gates from the node

Transistor resizing employed here increases the critical charge Q_{crit} of a selected gate by improving its drive strength and capacitance. (Q_{crit} is the minimal charge that must be generated in order to upset the node [185].) The resizing is performed such that Q_{crit} matches realistic charges. Hence, the charge deposition is modeled as a double exponential current pulse [186]:

$$I(t) = \frac{Q_{dep}}{T_\alpha - T_\beta}\left(e^{-t/T_\alpha} - e^{-t/T_\beta}\right) \tag{5.6}$$

where Q_{dep} denotes the charge deposited, T_α is collection time constant of the junction and T_β is the ion-track establishment time constant. According to [187], T_α is set to 10 ps and T_β is set to 5 ps. The worst-case charge that could be deposited is taken to be 130 fc.

The worst case condition for particle strike occurs, when the site of particle strike is at the output of the gate, since the particle strike could effectively contribute logic changes

TABLE 5.1: Gate counts of individual protection schemes

EDC	Braun (150 gates)			b20 (14159)		
	Predictor	EDN	Total	Predictor	EDN	Total
Linear	165	35	**350**	16025	3139	**33323**
QS $GF(2)$	127	18	**295**	14659	1460	**30278**
QS $GF(4)$	165	45	**360**	17035	3721	**34915**
QS $GF(16)$	228	88	**466**	20896	7658	**42713**

at the fanout gates. A gate selected for hardening is simulated, along with its fanout, assuming that the above-mentioned worst-case charge has been deposited at its output. The size of the gate is increased until the peak transient at its output falls below 0.5 V_{dd}.

5.3.3 Synthesis and evaluation of circuit with hardened gates

As mentioned above, the circuits used for experiments were synthesized and mapped to a library consisting of INV, 2-input NOR, 2-input NAND, 3-input NOR and 3-input NAND gates. Spice libraries for 45nm technology were obtained from Predictive Technology Models [24][1].

After selective hardening, the area of the circuit is derived from the cell library. For computing the power consumption, a large number of random input test cases is generated in the test bench of the design to create switching activity file using Synopsys VCS which is then used by Synopsys Power Compiler to estimate dynamic power.[1]

5.3.4 Experimental results

The combinational 4×4 Braun array multiplier circuit [188] (synthesized using optimizations from [189]) and the ITC-99 benchmark circuit b20 is considered in the experiments. These circuits are representative, as the multiplier is a small and highly structured arithmetic block and b20 is an instance of a larger random-logic circuit. The exhaustive simulation of all 256 possible input vectors was employed for the Braun multiplier and $1,000$ random vectors for b20. Four EDC-based protection architectures were synthesized for these circuits using linear parity and QS codes over $GF(2)$, $GF(4)$ and $GF(16)$. These codes employ 1, 1, 2 and 4 redundant bits, respectively. The composition of the architectures is shown in Table 5.1 (recall that it is not an objective of this work to optimize the predictor and the EDN).

For each of the architectures, an exhaustive fault-injection campaign for all single-stuck-at faults is performed. For each fault f (single-stuck-at-0 and single-stuck-at-1), the sets $|I_{\mathrm{DET}}(f)|$ of input vectors that detect the fault and $|I_{\mathrm{SDC}}(f)|$ of faults which are not detected (silent data corruptions) are obtained. From the cardinalities of these sets, the attack success rate $ASR(f)$ is obtained.

[1]This study has been conducted by research partner Sudarshan Srinivasan

TABLE 5.2: ASR classes (no hardening) for Braun multiplier, single faults

EDC	*ASR* class											
	0	(0, 0.1)	[0.1, 0.2)	[0.2, 0.3)	[0.3, 0.4)	[0.4, 0.5)	[0.5, 0.6)	[0.6, 0.7)	[0.7, 0.8)	[0.8, 0.9)	[0.9, 1)	1
Linear	66	39	65	57	41	18	6	6	1	1	0	0
QS $GF(2)$	5	0	0	1	5	12	112	88	36	20	7	14
QS $GF(4)$	18	4	16	29	40	52	48	59	28	3	0	3
QS $GF(16)$	54	22	64	123	34	3	0	0	0	0	0	0

TABLE 5.3: ASR classes (no hardening) for b20, single faults

EDC	*ASR* class											
	0	(0, 0.1)	[0.1, 0.2)	[0.2, 0.3)	[0.3, 0.4)	[0.4, 0.5)	[0.5, 0.6)	[0.6, 0.7)	[0.7, 0.8)	[0.8, 0.9)	[0.9, 1)	1
Linear	7237	572	714	1010	795	572	529	185	61	22	8	148
QS $GF(2)$	651	3	86	434	934	3997	4941	1120	381	157	3	641
QS $GF(4)$	1385	208	1376	5993	1534	464	476	169	21	6	0	220
QS $GF(16)$	3258	5044	968	257	92	9	62	7	0	0	0	47

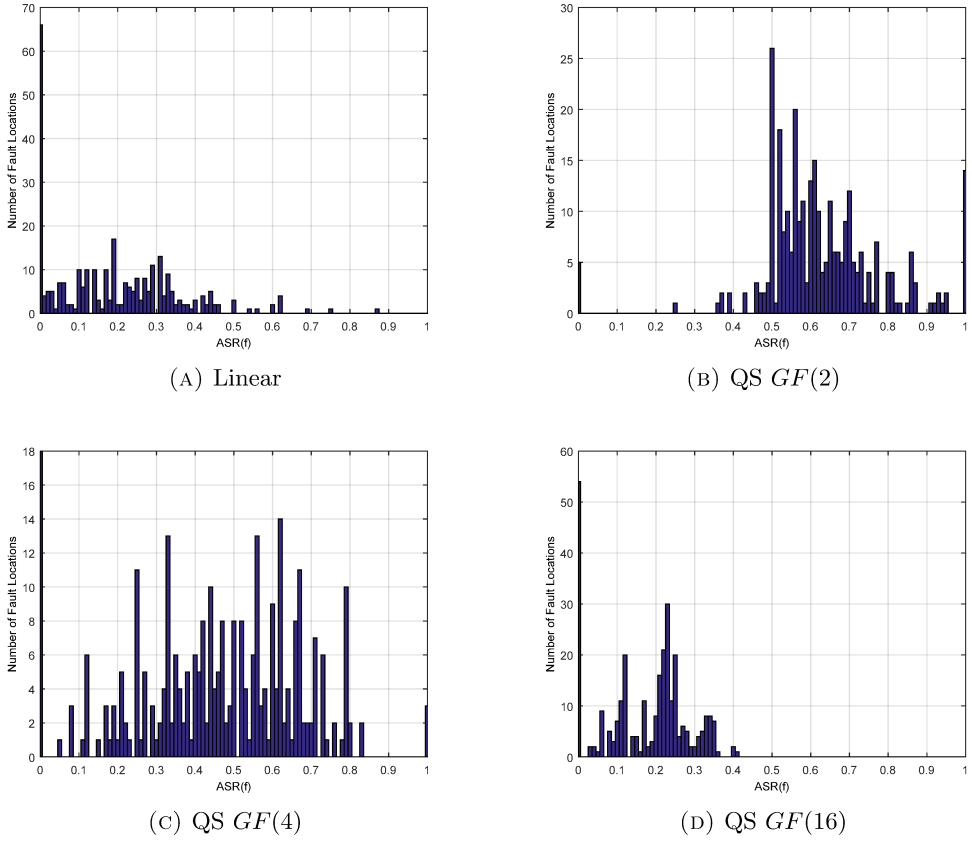
Each fault is assigned into an ASR class, as shown in Tables 5.2 and 5.3. Class 0 includes faults f with $ASR(f) = 0$, i. e., faults which are always detected. Class (0, 0.1) includes faults f with $0 < ASR(f) < 0.1$; class [0.1, 0.2) includes faults f with $0,1 \leq ASR(f) < 0.2$; and so forth. Note that a square bracket indicates that the value is included into the class while a round bracket denotes that the value is excluded. Faults from a class with larger boundaries are more critical than faults from a class with smaller boundaries, as the malicious attacker has a higher probability that the advertent fault injection is undetected; his chances to escape detection are perfect for faults from class 1. Note that faults that are always masked (redundant) have no valid ASR value are not included in the table.

Tables 5.2 and 5.3 allow some interesting insights. As expected, QS codes using 2 and 4 redundant bits perform better than the QS code over $GF(2)$ with only one redundant bit. However, predictions made using the additive error model at the outputs are not confirmed. It has been shown in [169] that $Q(e) \leq Q_{mc}$ where $Q_{mc} = 1/2$ if the QS code over $GF(2)$ is employed; for its counterparts over $GF(4)$ and $GF(16)$, the value of Q_{mc} equals 1/4 and 1/16, respectively. When considering the distribution of codes to ASR classes for these codes, $ASR(f)$ is by no means bounded by these values. Moreover, some of the faults are never detected, creating a large vulnerability. As pointed out above, the mismatch between $Q(e)$ and $ASR(f)$ is due to two modeling deviations. $Q(e)$ is defined for an additive error e on the circuit's outputs, while $ASR(f)$ assumes stuck-at faults within the circuit's combinational logic. Furthermore, $Q(e)$ assumes that the circuit's output values are distributed uniformly and therefore all codewords have a nonzero probability of showing up. In contrast, $ASR(f)$ is based on the presumption that the circuit's input vectors are uniformly distributed, meaning that some of the output vectors may never be produced.

Figures 5.9 and 5.10 depict the exact histograms of ASR classes for both Braun multiplier and b20 circuits. These histograms visualize the data of Tables 5.2 and 5.3

Since $ASR(f)$ is a random variable itself, its mean can be obtained by averaging over all possible faults

$$\overline{ASR} = \frac{1}{|F|} \sum_{f \in F} ASR(f) = \frac{1}{|F|} \sum_{f \in F} \frac{|I_{\mathrm{SDC}}(f)|}{|I_{\mathrm{SDC}}(f)| + |I_{\mathrm{DET}}(f)|} \tag{5.7}$$



(A) Linear

(B) QS $GF(2)$

(C) QS $GF(4)$

(D) QS $GF(16)$

FIGURE 5.9: Histograms of ASR values of single faults, Braun multiplier

where F is the list of all modeled faults (here: single-stuck-at faults) and I is the set of all possible input vectors. The mean $SDC(f)$ is obtained the same way as

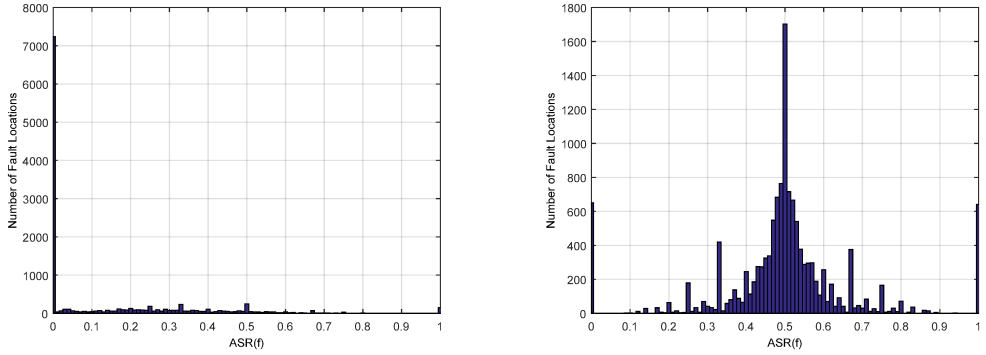
$$\overline{SDC} = \frac{1}{|F|} \sum_{f \in F} \frac{|I_{\text{SDC}}(f)|}{|I|} \tag{5.8}$$

Interestingly, for Braun multiplier, $\overline{ASR}$ does not correlate with Q_{mc} of the respective robust EDC. It is however the case for b20. In case of a large circuit, the ASR values are centered around the Q_{mc}. However, as mentioned previously, there is a significant amount of faults in the circuit which are never detected. This problem is tackled by hardening of these locations.

5.3.4.1 Evaluation of hardening for single faults

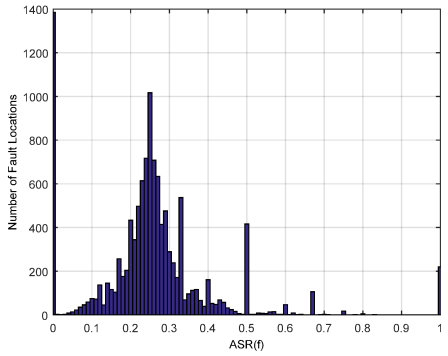
Hardening targets the faults from the most critical classes as indicated in Tables 5.2 and 5.3. Hence, following four hardening scenarios are defined:

- **ASR0.9**: Harden all gates with $ASR(f) > 0.9$ for either stuck-at-1 or stuck-at-0 fault at that gate.

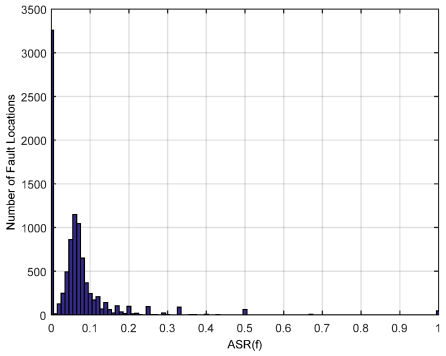


(A) Linear

(B) QS $GF(2)$



(C) QS $GF(4)$



(D) QS $GF(16)$

FIGURE 5.10: Histograms of ASR values of single faults, b20

- **ASR0.5**: Harden all gates with $ASR(f) > 0.5$ for either stuck-at-1 or stuck-at-0 fault at that gate.
- **10**%: Harden 10% of gates from the most critical ASR classes.
- **50**%: Harden 50% of gates from the most critical ASR classes.

Scenario **ASR0.9** aims at eliminating most critical faults. Scenario **ASR0.5** allows only faults with attack success rate of 1/2 or less. The remaining two scenarios consider a budget of logic gates available for hardening.

Table 5.4 summarizes the effects and the cost of the various hardening scenarios. The resizing technique is applied to the selected gates, and the area of the re-synthesized circuit and its power consumption are calculated. Only the original device itself, not the predictor or the EDN, is hardened. The area and power consumption reported are for the original device only. Incorporating the predictor and the EDN into the calculation would essentially add a large offset and underestimate the cost of selective hardening. The number of gates, the area requirement and the power consumption of the hardened circuit are reported in columns 2 through 4 and and 7 through 9 of Table 5.4.

The mean ASR and the mean SDC obtained by Eq. 5.7 and 5.8 are also reported. Since faults are assumed not to occur on hardened gates, they do not count towards mean ASR. $\overline{ASR}$ can be seen as a measure how difficult it is to impose an undetected attack

TABLE 5.4: Different hardening scenarios, single faults

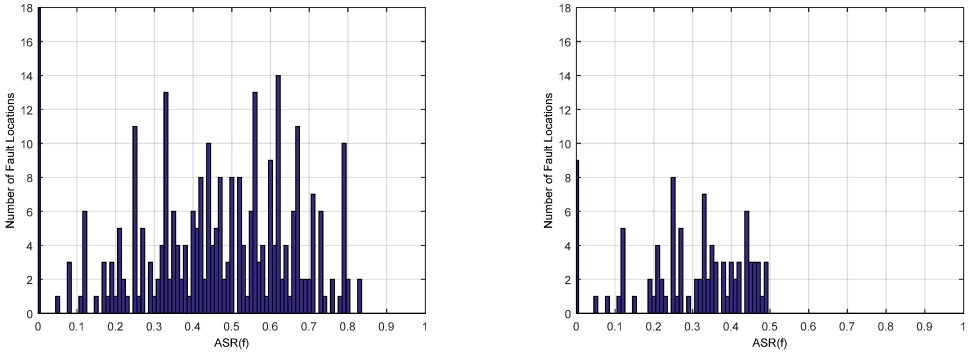
	Braun multiplier					b20				
Scenario	Gates hardened	Area [μm^2]	Power [μW]	$\overline{ASR}$	$\overline{SDC}$	Gates hardened	Area [μm^2]	Power [μW]	$\overline{ASR}$	$\overline{SDC}$
Linear code										
None	–	217.9	24.84	0.188	0.028	–	21243	2.4631	0.129	0.0021
ASR0.9	No hardening necessary					56	21265	2.6049	0.126	0.0020
ASR0.5	9	225.9	30.58	0.165	0.025	648	21488	2.7493	0.006	0.0001
10%	15	227.5	31.79	0.155	0.023	1415	21775	2.9399	0.075	0.0008
50%	75	245.6	42.16	0.066	0.006	7080	24893	4.6543	0.004	0.0001
QS $GF(2)$										
None	–	217.9	24.84	0.627	0.129	–	21243	2.4631	0.5	0.0152
ASR0.9	18	223.2	28.74	0.601	0.109	90	21283	2.6170	0.500	0.0153
ASR0.5	148	262.6	54.45	0.470	0.001	7200	24331	4.5108	0.500	0.2158
10%	15	223.2	28.74	0.606	0.113	1415	21838	2.8712	0.500	0.2157
50%	75	242.4	40.91	0.549	0.058	7080	24278	4.4807	0.501	0.2157
QS $GF(4)$										
None	–	217.9	24.84	0.457	0.088	–	21243	2.4631	0.256	0.0058
ASR0.9	3	217.9	25.16	0.451	0.086	66	21273	2.6102	0.254	0.0054
ASR0.5	103	253.5	48.52	0.291	0.016	153	21311	2.6345	0.250	0.0054
10%	15	226.4	31.51	0.434	0.076	1415	21847	2.9743	0.252	0.0941
50%	75	244.5	41.11	0.339	0.032	7080	24289	4.4795	0.250	0.0467
QS $GF(16)$										
None	–	217.9	24.84	0.172	0.028	–	21243	2.4631	0.063	0.0019
10%	15	225.4	31.03	0.161	0.024	1415	21656	2.8552	0.062	0.0011
50%	75	240.8	41.62	0.129	0.011	7080	24660	4.8600	0.062	0.0003

when the attacker can accurately control the location of fault injection while $\overline{SDC}$ is a measure of the attacker's chances of success in absence of such capability.

It can be seen that relatively few gates are associated with $ASR(f) > 0.9$, and therefore scenario **ASR0.9** has rather low additional cost. Scenario **ASR0.5** requires the hardening of a large number of gates (almost all gates in some cases) Even in such cases the area increase is limited. This is because the hardened cells with resized transistors are often not much larger than their unhardened counterparts (the sizes are sometimes even identical). In contrast, power consumption grows almost linearly with the number of the hardened gates and is largely independent from the actual selection of gates. It can be concluded that the main cost of selective hardening is power overhead, which tends to scale with the number of selected gates.

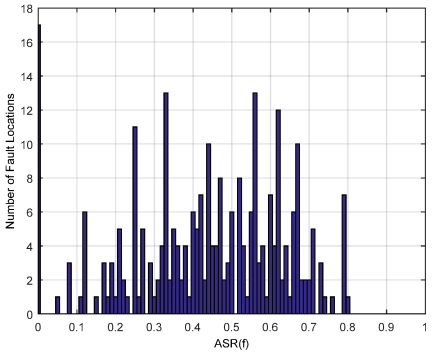
The mean ASR reported for the larger circuit b20 for nonlinear codes matches well with theoretical predictions from [190] ($Q(e)$ is bounded by 0.5, 0.25 and 0.00625 for the nonlinear code over $GF(2)$, $GF(4)$ and $GF(16)$, respectively). The deviations of the measured mean ASR from these values are much larger for the Braun multiplier. This is probably due to the fact that a uniform distribution of vectors at the multiplier's inputs results in a very non-uniform distribution of output vectors. Hardening improves mean ASR for the linear code and for the Braun multiplier but has almost no effect for the nonlinear code in b20. This is because almost all faults have very similar $ASR(f)$ and eliminating some of them by hardening does not improve the mean ASR. However, mean SDC is significantly improved by hardening, even though the selection of the gates was driven by a different metric (ASR).

Figure 5.11 depicts the effect of hardening for various scenarios for Braun multiplier protected by QS over GF(4).

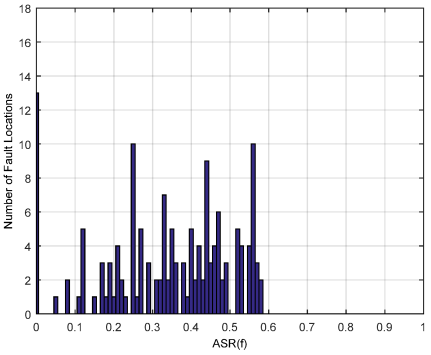


(A) ASR0.9

(B) ASR0.5



(C) 10%



(D) 50%

FIGURE 5.11: ASR distribution for various hardening scenarios, QS $GF(4)$, Braun multiplier

It can be observed that, compared to the original histograms, the critical faults are removed. In case of Braun multiplier this has an effect on the mean ASR, which tends to Q_{mc} if the number of hardened gates increases. For a larger circuit, b20, $\overline{ASR}$ corresponds to the Q_{mc} of the respective robust code and it does not change significantly when different hardening scenarios are applied (Table 5.4). The hardening solely removes the outliers.

5.3.4.2 Multiple faults

The experiment is repeated for random selections of double and higher-order multiple faults (5000 faults of multiplicities 3 to 6). The results of the two experiments are reported in Tables 5.5 and 5.6 (columns 3 and 4 repeat the results for single faults for reference, and column 2 quotes the power overhead in per cent compared with power consumed when no hardening is performed).

For both circuits, $\overline{SDC}$ improves with increasing fault multiplicity for nonlinear code and shows no clear trend for linear code. This is consistent with the intuition that a fault with a higher multiplicity is more severe and easier to detect. $\overline{ASR}$ is much worse for multiple faults if a linear code is employed and tends to slightly improve for nonlinear codes. Interestingly, hardening has little effect on mean ASR (but significantly improves

Table 5.5: Faults of different multiplicity, Braun multiplier

Scenario	Power overh.[%]	Single faults		Double faults		Multiple faults	
		$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$
Linear code							
None	–	0.1878	0.0280	0.3674	0.0275	0.4720	0.0257
ASR0.5	23.1	0.1654	0.0246	0.3556	0.0169	0.4703	0.0247
10%	28.0	0.1553	0.0229	0.3492	0.0163	0.4682	0.0241
50%	69.7	0.0661	0.0058	0.2773	0.0241	0.4466	0.0150
QS $GF(2)$							
None	–	0.6269	0.1290	0.6015	0.0508	0.5632	0.0343
ASR0.9	15.7	0.6006	0.1093	0.5879	0.0443	0.5588	0.0311
ASR0.5	119.2	0.4702	0.0005	0.4778	0.0003	0.5103	0.0003
10%	15.7	0.6061	0.1132	0.5912	0.0458	0.5598	0.0317
50%	64.7	0.5492	0.0575	0.5555	0.1245	0.5449	0.0203
QS $GF(4)$							
None	–	0.4574	0.0875	0.4304	0.0221	0.3639	0.0202
ASR0.9	1.3	0.4507	0.0859	0.4273	0.0217	0.3631	0.0199
ASR0.5	95.3	0.2907	0.0158	0.3017	0.0047	0.2980	0.0059
10%	26.9	0.4341	0.0757	0.4150	0.00195	0.3571	0.0185
50%	65.5	0.3389	0.0318	0.3226	0.0102	0.3236	0.0102
QS $GF(16)$							
None	–	0.1718	0.0283	0.1653	0.0043	0.1257	0.0060
10%	24.9	0.1609	0.0242	0.1579	0.00035	0.1223	0.0054
50%	67.6	0.1289	0.0110	0.1138	0.00012	0.1138	0.0031

mean SDC). Therefore, if the attacker is capable to target specific gates, hardening some of them is not likely to prevent an undetected attack. The protection must be based on a sufficient performance of the code employed.

One basic assumption on the hardening scheme is that the gate resizing is sufficient to protect against all possible errors and consequently no error effects on the hardened gates are possible. The good values of $\overline{SDC}$ achieved by most codes provide support for an additional level of protection if the attacker employs laser pulses with energy larger than the maximal values used during transistor resizing. In this case, excessive charge carriers may flip the output of the hardened gate. However, they are highly likely to simultaneously flip the outputs of neighboring gates, resulting in (random) multiple errors, which are relatively easy to detect as suggested by low values of $\overline{SDC}$. Moreover, excessive charge carriers will take time to dissipate, likely imposing (single or multiple) errors in subsequent clock cycle. It has been reported in [190] that such multi-cycle faults have a much higher probability of detection by robust codes than single-cycle faults.

The QS code over $GF(2)$ has the lowest hardware overhead of all protection schemes

Table 5.6: Faults of different multiplicity, b20

Scenario	Power overh.[%]	Single faults		Double faults		Multiple faults	
		$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$
Linear code							
None	–	0.1289	0.0021	0.1350	0.0031	0.3565	0.0072
ASR0.5	5.8	0.1257	0.0020	0.1324	0.0031	0.3478	0.0069
10%	11.6	0.0056	0.0001	0.1205	0.0025	0.3399	0.0063
50%	19.4	0.0752	0.0008	0.0975	0.0020	0.3168	0.0055
QS $GF(2)$							
None	–	0.4997	0.0152	0.4996	0.0118	0.4991	0.0120
ASR0.9	6.2	0.4999	0.0153	0.4997	0.0116	0.4994	0.0120
ASR0.5	83.1	0.5000	0.2158	0.4999	0.0464	0.5003	0.0500
10%	16.6	0.4998	0.2157	0.4996	0.0477	0.5003	0.0500
50%	81.9	0.5008	0.2157	0.5001	0.0477	0.5007	0.0500
QS $GF(4)$							
None	–	0.2555	0.0058	0.2549	0.0059	0.2557	0.0052
ASR0.9	5.9	0.2542	0.0054	0.2542	0.0054	0.2552	0.0052
ASR0.5	6.9	0.2501	0.0054	0.2537	0.0054	0.2511	0.0052
10%	20.8	0.2521	0.0941	0.2520	0.0233	0.2511	0.0251
50%	81.9	0.2501	0.0467	0.2502	0.0232	0.2503	0.0250
QS $GF(16)$							
None	–	0.0629	0.0019	0.0621	0.000622	0.0634	0.0013
10%	15.9	0.0615	0.0011	0.0621	0.0006	0.0631	0.0013
50%	86.1	0.0615	0.0003	0.0620	0.0006	0.0626	0.0012

considered. It turns out to have the worst detection capabilities of single faults for most scenarios, in terms of both $\overline{ASR}$ and $\overline{SDC}$. This is unexpected, as QS codes are robust and should outperform linear codes in terms of ASR. The two-redundant bit QS code over $GF(4)$ and the linear parity code are, by and large, comparable. The four-redundant bit QS code over $GF(16)$ has much better detection properties but requires around 30% additional gates than other codes.

Selective hardening appears to be a useful alternative to exclude large vulnerabilities (faults which can be injected with no or little chance of detection), as evidenced by relatively small number of gates to be hardened in scenario **ASR0.9**. On the other hand, ASR-guided selection of gates is not suitable to compensate for systematic deficiencies of the basic protection scheme used. If the code misses faults with a high probability, the hardening technique is only effective if almost all gates in the circuit are hardened.

It is a legitimate question whether using a better code with a higher area overhead or employing a weaker code in combination with hardening selected gates is a better design alternative. When comparing the unhardened design using the QS code over $GF(16)$ and the hardened versions based on other codes, there is no clear conclusion.

5.4 Application to cryptographic circuits

In previous section robust codes have been applied to protection of general purpose circuits. This section addresses application of robust codes to protection of the combinational circuit implementation of the lightweight block cipher PRINCE [191]. The PRINCE algorithm is organized in 10 rounds, and each round is composed of four operations. The own circuit implementation was designed which is fully unrolled and performs all 10 rounds of encryption in one cycle. It is purely combinational and consists of 8,320 gates (the reference combinational implementation from [191] requires 8,260 gates).

In order to investigate the granularity of protection, the flow introduced in Sec. 5.2 is applied to the complete 10-round circuit; the portion of the unrolled circuit corresponding to one round; and one SBox, which is the only nonlinear operation of the cipher. The complete circuit and one round have 64 inputs and 64 outputs, whereas the SBox has four inputs and four outputs.

Following linear codes are employed, along with the single parity code from previous section: interlaced parity code with two through four parity bits responsible for non-overlapping groups of information bits. The extended (8, 4) Hamming code is employed solely for protection of the SBox of PRINCE.

Along with QS codes from previous section, QS code over $GF(8)$ is employed. This code has three redundant bits (RB) and its maximal error masking probability is $Q_{mc} = 0.125$. Additionally, PC codes with same number of RBs and same Q_{mc} as the QS codes are employed. It is to note that the robust codes with multiple RBs are not applied to the SBox of PRINCE due to the fact that the latter has only four outputs.

Three different fault-injection campaigns for each considered code and circuits were performed: one assuming single stuck-at-1 and stuck-at-0 faults of one clock cycle duration at all lines of the circuit, the predictor and the EDN, one assuming 10,000 randomly selected double faults, and one assuming 10,000 faults of higher multiplicity. All campaigns were run using random input sequences of 10,000 vectors.

5.4.1 Results

The mean ASR and SDC results are shown in Table 5.7. It can be seen that the linear parity codes outperform the robust codes for single faults in SBox and one round of PRINCE and a low number of redundant bits. This observation holds for both $\overline{ASR}$ and $\overline{SDC}$.

The good performance of linear codes might indicate that many single *faults* within the circuit structure tend to manifest themselves as single *errors* at the circuit outputs. This is plausible for cryptographic hardware composed of mostly reversible (bijective) modules. Since linear codes have a minimal Hamming distance of 2, they are effective in detecting such faults. For example, the 4-bit parity code $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ will always detect a single bit-flip at the first output in x_1. On the other hand, the QS code over $GF(2)$ $x_1x_2 \oplus x_3x_4$ will only detect the same bit-flip if $x_2 = 1$.

Table 5.7: Mean ASR and SDC, PRINCE circuit

	Code	RB	Single faults		Double faults		Multiple faults	
			$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$
SBox	Parity	1	0.0677	0.0434	0.2754	0.1145	0.4210	0.2788
	QS $GF(2)$	1	0.4422	0.2598	0.5063	0.2255	0.5140	0.3452
	Punctured Cubic	1	0.4453	0.2254	0.5134	0.2565	0.5230	0.3012
	QS $GF(4)$	2	0.2565	0.0155	0.2476	0.0456	0.2402	0.0174
	Parity interlaced	2	0.0348	0.0184	0.1238	0.0502	0.1962	0.0119
	Punctured Cubic	2	0.2587	0.0245	0.2357	0.0395	0.2335	0.0234
	QS $GF(8)$	3	0.1245	0.0245	0.1235	0.0424	0.1221	0.0301
	Punctured Cubic	3	0.1256	0.0245	0.1251	0.0234	0.1254	0.0326
	Hamming	4	0	0	0.0137	0.0039	0.0283	0.0014
	QS $GF(16)$	4	0.0634	0.0155	0.0627	0.0532	0.0633	0.0134
	Punctured Cubic	4	0.0678	0.0201	0.0626	0.0133	0.0625	0.0223
One round	Parity	1	0.0980	0.0467	0.3424	0.0232	0.4727	0.0568
	QS $GF(2)$	1	0.4970	0.1708	0.5007	0.0321	0.5007	0.0568
	Punctured Cubic	1	0.4988	0.0264	0.5004	0.0267	0.5001	0.0588
	Parity interlaced	2	0.0667	0.0057	0.1979	0.0025	0.2500	0.0068
	QS $GF(4)$	2	0.2481	0.0743	0.2506	0.0140	0.2509	0.0256
	Punctured Cubic	2	0.2503	0.0102	0.2501	0.0022	0.2502	0.0047
	QS $GF(8)$	3	0.1344	0.0408	0.1330	0.0370	0.1291	0.0659
	Punctured Cubic	3	0.1239	0.0047	0.1257	0.0011	0.1257	0.0024
	Parity interlaced	4	0.0439	0.0042	0.1051	0.0015	0.1121	0.0031
	QS $GF(16)$	4	0.0623	0.0025	0.0626	0.0005	0.0630	0.0011
	Punctured Cubic	4	0.0624	0.0023	0.0625	0.0013	0.0623	0.0030
Full circuit	Parity	1	0.4614	0.2219	0.4755	0.1829	0.4912	0.3223
	QS $GF(2)$	1	0.5008	0.1157	0.5001	0.0639	0.5003	0.1094
	Punctured Cubic	1	0.4996	0.1311	0.4999	0.1352	0.4999	0.2487
	Parity interlaced	2	0.2305	0.0908	0.2392	0.0308	0.2435	0.0012
	QS $GF(4)$	2	0.2501	0.0505	0.2502	0.0319	0.2501	0.0546
	Punctured cubic	2	0.2503	0.0587	0.2507	0.0676	0.2501	0.0123
	QS $GF(8)$	3	0.1325	0.0501	0.1298	0.0056	0.1267	0.0035
	Punctured cubic	3	0.1250	0.0014	0.1251	0.0111	0.1248	0.0004
	Parity interlaced	4	0.0568	0.0150	0.0600	0.0061	0.0609	0.0003
	QS $GF(16)$	4	0.0624	0.0103	0.0625	0.0080	0.0626	0.0101
	Punctured cubic	4	0.0628	0.0014	0.0630	0.0056	0.0625	0.0002

The situation changes when faults of higher multiplicity, more elaborate codes or larger circuits are considered. The performance of the QS code over $GF(2)$ is quite close to the parity code for the SBox and is practically indistinguishable from the parity code for the larger circuits. Robust codes with four RBs tend to outperform their linear

TABLE 5.8: Mean ASR and SDC, faults in original device only

	Code	RB	Single faults		Double faults		Multiple faults	
			$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$
Full circuit	Parity	1	0.4611	0.0182	0.4694	0.0059	0.4859	0.0085
	QS $GF(2)$	1	0.4998	0.0194	0.5002	0.0063	0.5002	0.0087
	Punctured Cubic	1	0.4999	0.0131	0.5002	0.0063	0.4999	0.0087
	Parity interlaced	2	0.2312	0.0091	0.2361	0.0029	0.2426	0.0042
	QS $GF(4)$	2	0.2494	0.0096	0.2502	0.0031	0.2501	0.0043
	Punctured cubic	2	0.2502	0.0059	0.2500	0.0031	0.2499	0.0043
	QS $GF(8)$	3	0.1321	0.0034	0.1245	0.0056	0.1239	0.0028
	Punctured cubic	3	0.1253	0.0026	0.1251	0.0015	0.1248	0.0021
	Parity interlaced	4	0.0571	0.0022	0.0588	0.0007	0.0607	0.0010
	QS $GF(16)$	4	0.0626	0.0023	0.0625	0.0008	0.0625	0.0010
	Punctured cubic	4	0.0624	0.0014	0.0625	0.0008	0.0624	0.0010

TABLE 5.9: Mean ASR and SDC, exploitable fault locations

	Code	RB	Single faults		Double faults		Triple faults	
			$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$	$\overline{ASR}$	$\overline{SDC}$
PRINCE round 8/9	Parity	1	0.5001	0.2501	0.5001	0.5633	0.4996	0.4371
	QS $GF(2)$	1	0.5412	0.2707	0.5065	0.5729	0.5042	0.4424
	Parity interlaced	2	0.3119	0.1560	0.2603	0.2969	0.2564	0.2262
	QS $GF(4)$	2	0.2505	0.1253	0.2501	0.2817	0.2501	0.2188
	Punctured cubic	3	0.1981	0.0990	0.1375	0.1590	0.1329	0.1184
	Parity interlaced	4	0.1404	0.0702	0.0756	0.0901	0.0708	0.0644
	QS $GF(16)$	4	0.0622	0.0311	0.0624	0.0703	0.0625	0.0547
	Punctured cubic	4	0.1407	0.0703	0.0756	0.0898	0.0708	0.0643

counterparts. It is interesting that $\overline{ASR}$ almost perfectly matches Q_{mc} for all fault multiplicities. This is different compared with the significant deviations reported in Sec. 5.3 for Braun multiplier design.

The reason is the high degree of reversibility of cryptographic circuits: the output codewords are uniformly distributed, whereas a multiplier produces only a small subset of output bit combinations. Comparing both classes of robust codes with each other, the $\overline{SDC}$ of the punctured cubic codes tends to track with $\overline{SDC}$ of linear codes, while their $\overline{ASR}$ closely matches the $\overline{ASR}$ of quadratic sum codes. They appear to combine the best of the worlds, even though their construction is more complex.

Table 5.8 shows the results for faults injected only in the original device, whereas predictor and EDN are fault-free. The numbers do not deviate from the results when all parts of the architecture were considered during fault injection, indicating the high stability of the predictions.

TABLE 5.10: Distribution of ASR values over individual faults, one round of PRINCE

	Code	RB	Number of faults with ASR in indicated range											
			0.0	0.0–0.1	0.1–0.2	0.2–0.3	0.3–0.4	0.4–0.5	0.5–0.6	0.6–0.7	0.7–0.8	0.8–0.9	0.9–1.0	1.0
single faults	Parity	1	855	0	23	33	11	7	9	14	3	1	0	64
	QS $GF(2)$	1	0	0	0	1	5	526	483	5	0	0	0	0
	Punctured Cubic	1	0	0	0	0	5	514	496	5	0	0	0	0
	Parity interlaced	2	58	0	4	8	4	0	0	0	0	0	0	8
	QS $GF(4)$	2	0	0	46	941	33	0	0	0	0	0	0	0
	Punctured Cubic	2	0	0	33	945	41	1	0	0	0	0	0	0
	QS $GF(8)$	3	0	91	883	33	0	3	5	0	0	0	0	4
	Punctured Cubic	3	0	102	916	2	0	0	0	0	0	0	0	0
	Parity interlaced	4	174	0	0	0	0	0	0	0	0	0	0	8
	QS $GF(16)$	4	0	182	0	0	0	0	0	0	0	0	0	0
	Punctured Cubic	4	0	1012	8	0	0	0	0	0	0	0	0	0
double faults	Parity	1	164	936	2858	3128	2081	2842	2844	248	235	140	32	43
	QS $GF(2)$	1	16	0	4	9	54	7130	7348	80	20	5	2	26
	Punctured Cubic	1	0	0	0	0	8	1413	1486	11	0	0	0	1
	Parity interlaced	2	1341	288	539	411	222	340	396	15	19	39	6	9
	QS $GF(4)$	2	17	4	390	12409	392	16	10	6	1	0	0	9
	Punctured Cubic	2	0	0	74	2256	64	0	0	0	0	0	0	1
	QS $GF(8)$	3	4	969	11653	306	58	51	24	12	5	5	3	13
	Punctured Cubic	3	0	207	2209	6	0	0	0	0	1	0	0	1
	Parity interlaced	4	2361	469	319	92	87	190	189	25	19	37	5	18
	QS $GF(16)$	4	0	2322	0	2	0	0	0	0	0	0	0	0
	Punctured Cubic	4	2	2394	27	0	0	0	0	0	0	0	0	0
multiple faults	Parity	1	0	9	105	520	1600	9564	7116	82	13	5	1	1
	QS $GF(2)$	1	0	0	4	10	50	9014	9510	79	11	3	0	0
	Punctured Cubic	1	0	0	0	1	12	2882	3012	22	1	0	0	0
	Parity interlaced	2	424	466	1030	3634	868	663	249	10	5	1	1	1
	QS $GF(4)$	2	3	3	276	17206	357	16	11	5	1	2	1	0
	Punctured Cubic	2	0	0	120	4701	136	1	1	0	0	0	0	0
	QS $GF(8)$	3	3	860	16765	238	30	20	5	1	2	1	0	0
	Punctured Cubic	3	0	354	4579	17	0	0	0	1	0	0	0	0
	Parity interlaced	4	1602	2582	1814	940	258	177	49	11	7	0	0	0
	QS $GF(16)$	4	0	4859	17	4	0	1	0	0	0	0	0	0
	Punctured Cubic	4	2	4977	61	1	0	0	0	0	0	0	0	0

In order to study the behavior of the codes in an actual attack scenario, faults were injected according to the requirements of the multi-stage algebraic attack on the PRINCE cipher [39]. The 64-bit state of PRINCE is organized into 4-bit *nibbles*, and the attacker must flip an arbitrary number of bits in a single nibble without affecting other bits for successful cryptanalysis. The attack is mounted in two stages: in the first stage the fault is injected in a nibble in the state during round 9 (out of 10), and in the second stage the injection is done into a state nibble in round 8. Three locations from the same state nibble in round 9 were selected and a fault-injection campaign on all single, double and triple faults on these locations was performed. This models an attacker who targets these three locations by a probabilistic injection technique (such as laser illumination) in order to obtain some modification of the nibble with a high probability. The same procedure is repeated for other three suitable locations in round 8. The results are shown in Table 5.9. They are quite similar to the result for larger fault lists from Table 5.7, and the match with the theoretical Q_{mc} is even closer.

Table 5.10 shows the distribution of ASR values over individual faults for one round of PRINCE. The findings discussed above are supported: linear codes are quite effective for single and, to some extent, also double faults, but are inferior for multiple faults. Robust codes always exhibit a clear profile: the vast majority of faults is collected in

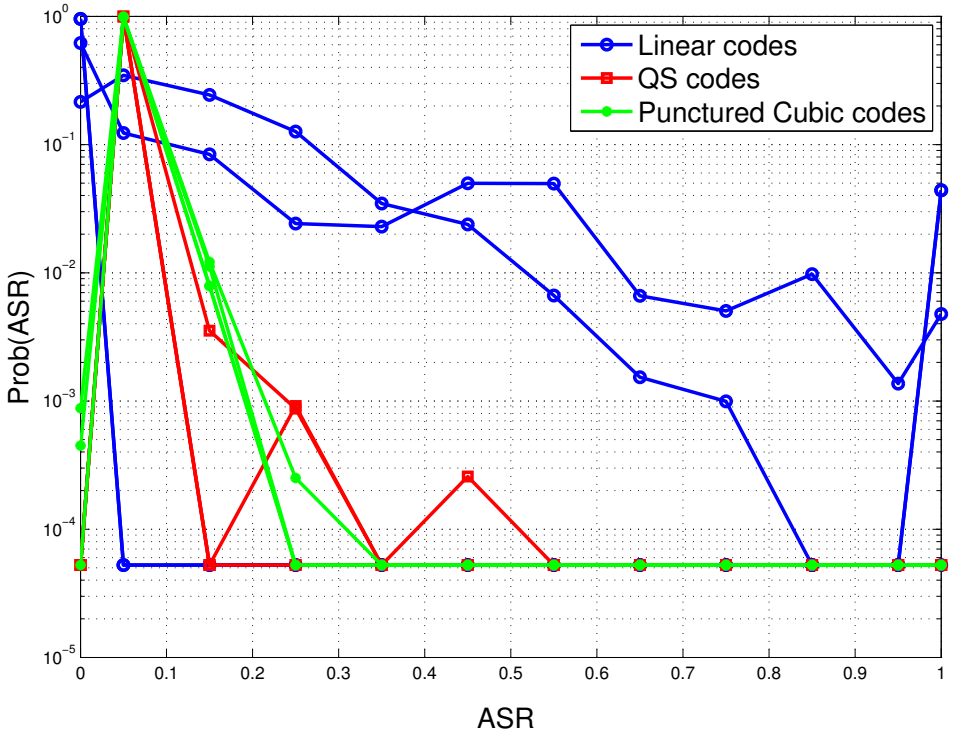


FIGURE 5.12: ASR classes for different codes with four RB (percentages)

the class(es) which contains Q_{mc} of the code. For example, the quadratic sum code over $GF(4)$ has two redundant bits and $Q_{mc} = 0.25$, which belongs to class 0.2–0.3. This class contains by far most faults.

An important observation about robust codes is that the number of faults with high $ASR(f)$ (1.0 or somewhat below) is much lower compared with linear codes with the same number of RBs. In particular, robust codes with four RBs have no high-ASR faults. This is important in security context, where an attacker is free to pick any desired fault. A fault with a high $ASR(f)$ is likely (and a fault with an $ASR(f)$ of 1 is certain) to escape detection, thus allowing the adversary to complete the attack. This finding is consistent with the purpose of the robust codes: to avoid the worst-case scenario which the attacker could make use of. Figure 5.12 visualizes the $ASR(f)$ distributions for different codes with four RB.

5.5 Summary

A cross-level protection solution for digital circuits which combines information redundancy (error-detecting codes) and low-level hardening was introduced. The results show that spots vulnerable to undetected faults exist in the circuit for any code considered. This is also true if advanced robust QS codes specifically designed for eliminating undetected faults or bounding their probability are employed. From a malicious attacker's point of view, such spots provide an opportunity to manipulate circuit operation without being noticed. This capability of the attacker is modeled by the new formalism of attack

success rate, and the relation between ASR and the value Q_{mc} previously used in formal analysis of robust codes is investigated. It is demonstrated that pinpointedly hardening vulnerable spots is feasible and associated with limited area and power overhead. On the other hand, ASR-guided selection of gates is not suitable to compensate for systematic deficiencies of the basic protection scheme used. If the code misses faults with a high probability, the hardening technique is only effective if almost all gates in the circuit are hardened. An efficient solution for a given circuit must combine an effective code with a good selection strategy for gate hardening.

The application of robust codes to protection of a cryptographic block against malicious attacks was also investigated. The error-detection capabilities of quadratic-sum and punctured cubic codes were determined and compared with the respective numbers for linear codes. Robust codes were demonstrated to provide better security against rare worst-case scenarios and also exhibited a good match to theoretical information-level predictions. Punctured cubic codes showed detection properties comparable to quadratic-sum codes with respect to malicious attacks and outperformed them when random transient faults were considered.

Chapter 6

Conclusion

Current CMOS technology is affected by process variations and is susceptible to physical disturbances such as ionizing radiation. This thesis aimed at improving fault tolerance of MIMO detection algorithms and underlying hardware. The trend in wireless communications is the usage of large scale antenna arrays which leads to integration of very large memories. This work addressed UW-OFDM in its interpretation as a virtual massive MIMO system. Consequently, detection algorithms initially introduced for generic large scale MIMO have been adopted to MIMO UW-OFDM. In particular, the modification of likelihood ascent search, which uses the soft information of the LMMSE initial solution, denoted MAP LAS was proposed. In coded scenario, the proposed MAP LAS algorithm outperforms the standard LAS algorithm. As far as the ML detection is concerned, MIMO UW-OFDM with LAS detection is able to outperform MIMO CP-OFDM with sphere decoding in uncoded scenario. However, associated computational complexity is overwhelming and not suited for real-time implementation. In coded scenario MIMO CP-OFDM with sphere decoding still performs better. It is to conclude that MIMO UW-OFDM represents performance/complexity tradeoff between MIMO CP-OFDM with LMMSE and MIMO CP-OFDM with sphere decoding (comparable to successive interference cancellation). The large sizes of required memories (as in MIMO UW-OFDM) renders their protection by standard means like error detecting codes costly. The errors in the receive buffer memory are therefore treated as additional non-Gaussian noise. This work proposed detection algorithms, derived from nonlinear MMSE estimator, that take memory errors into account when deciding on the detected symbols. The proposed log-NMMSE algorithm is able to significantly lower the BER floor introduced by the memory errors even for very high bit-flip probability of 10^{-3}. When LAS is applied on top of the proposed algorithm, the BER performance is shown to be very close to the memory error free case. With bit-flip probability of 10^{-4}, the effect of memory errors is canceled out. It has been shown, that allowing memory errors with bit-flip probability of 10^{-4} by reducing the supply voltage of the memory circuit would reduce the power consumption by 40%. Therefore, the proposed algorithm with LAS run on top would allow significant power savings, without sacrificing the performance. Future work could address other quasi-ML algorithms implemented recently (reactive tabu search, belief propagation on factor graphs, Markov chain Monte Carlo methods). Another interesting direction is the hardware implementation/optimization of proposed memory error resilient algorithms.

Actual hardware implementation of a MIMO receiver contains not only the detection algorithm but also some preprocessing (QR decomposition) or post-processing (channel decoding) blocks. Faults in different components of the receiver have different impact on the detection quality. QR decomposition represents the critical point for the correct operation of the whole reception chain. In order to quantify this impact, the QRD based on MMSE sorted Givens rotation was implemented fulfilling the stringent real-time requirements of the LTE standard. The implementation was then targeted by fault injection. The transient faults were injected into diverse computational components of the implementation. The novelty of this experiment is stressed out by the fact that previous works considered faults in internal buffering memories only. The implemented QRD was integrated into the precise bit-accurate system-level simulation that allowed to quantify the effect of faults injected into distinct computational sub-components on the system performance in terms of FER. The results indicated that the affected computation of the new diagonal element has the major impact on FER. Future research should therefore consider fault tolerant implementation of the boundary cell, specifically addressing the protection of the new diagonal element computation. As the errors in the latter rotations are less critical, it can be considered to provide varying amount of protection to first and subsequent iterations of the boundary cell.

Inherent vulnerability of contemporary hardware to physical disturbances makes it accessible to fault-based attacks. These attacks have been proven to be effective against cryptographic components, present in by far most current systems (communications systems for example). This work addressed countermeasures against fault based attacks. Here, robust codes were applied to actual circuits. In order to perform analysis of these codes, new fault-based metrics have been introduced opposed to standard error-centric metrics used previously. The introduced formalisms of attacks success rate and silent data corruption allowed to identify critical location of stuck-at-faults within the circuits which could escape detection. In order to remove these options for the possible attacker, hardening of these locations has been proposed. The proposed combined solution can be considered as a cross-level protection scheme that combines information and hardware redundancy. Also, the classes of circuits where the robust codes are not able to guarantee their properties have been identified. Finally, robust codes have been applied to protection of actual low-power PRINCE cipher implementation. The performance of robust codes and linear codes has been compared in terms of the introduced metrics. It has been shown that robust codes are indeed superior over linear codes in terms of ASR. Future work could concentrate on construction of codes that cover both malicious attacks and random transient faults and empirical investigation of their properties.

Appendix A

Convolutional codes

Convolutional codes are linear codes. The difference from the block linear codes lies in the encoding procedure. The encoding of K information bits to N code is performed in one time step. The redundant bit depends on the current value of the information bit and on the values of the information bits fed into the encoder at past time steps. The encoding operation is not memoryless [56]. Usually, K and N are small numbers (1,2,3), whereas for block codes K and N are large ($N = 2048$ for Reed-Solomon code). Due to the memory of the encoder, convolutional codes are able to perform well for small K and N. The memoryless encoder of a block code would produce poor codes for small K, N. Convolutional codes and block codes are closely related. In fact, a convolutional code can be regarded as a block code, with the generator matrix having special structure, such that the encoding operation is expressed as convolution [55].

For an information bit sequence $\mathbf{b}$, partitioned into blocks of length K:

$$\begin{aligned} \mathbf{b} &= (\mathbf{b}_0, \mathbf{b}_1, \dots) \\ \mathbf{b}_i &= (b_i^{(1)}, b_i^{(2)}, \dots, b_i^{(K)}) \end{aligned} \tag{A.1}$$

the encoder outputs the encoded bit sequence $\mathbf{c}$, partitioned into blocks of length N

$$\begin{aligned} \mathbf{c} &= (\mathbf{c}_0, \mathbf{c}_1, \dots) \\ \mathbf{c}_i &= (c_i^{(1)}, c_i^{(2)}, \dots, c_i^{(N)}) \end{aligned} \tag{A.2}$$

The index i denotes the time step or the time index. In general, $N > K$ code bits are generated at each time step from K information bits, yielding the code rate:

$$R = \frac{K}{N} \tag{A.3}$$

Consider an example rate 1/2 encoder depicted in Fig. A.1 The encoder contains a memory element and an exclusive-or element. Since the first code bit equals the information bit - $c_i^{(1)} = b_i$, it is a systematic encoder. In a non-systematic encoder, it is not possible to separate the information bits and the code bits within the codeword. The second code bit is obtained as $c_i^{(2)} = b_i + b_{i-1}$. Initially, the memory element is set to zero. The

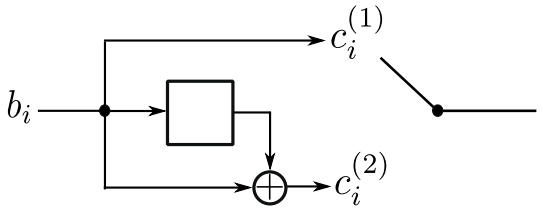


FIGURE A.1: Rate 1/2 convolutional encoder

delay, until the code values are produced is defined by the number of time steps (cycles) required to feed all memory elements of the encoder.

Looking at the example encoding circuit it is easy to identify the parameters of a convolutional code [54, 55]:

- maximal delay M is the number of time steps until valid code bits are produced by the encoder circuit. For the example encoder in Fig. A.1 the maximal delay is $M = 1$.
- code memory is the minimal number of registers required to construct the encoder circuit. It is at most $M \times K$.
- constraint length l_c is the overall number of information bits affecting code bits generated at time step i, including the information bits fed into the encoder up to M time steps back in time.

$$l_c = MK + K = (M + 1)K \tag{A.4}$$

A convolutional code is called systematic if the N code bits generated at time step i contain K information bits. For the example code it is the case and it is obviously systematic.

The codes with larger constraint length tend to provide better error correcting performance, hence one of the main design goals of the convolutional codes is to obtain a code with the maximum constraint length, while trying to keep the delay (number of required memory elements) and thus the complexity of the encoder as low as possible [56].

Convolutional codes are linear codes, therefore the encoding can be described by multiplication with a generator matrix $\mathbf{G}$ [55]

$$\mathbf{c} = \mathbf{G}\mathbf{b}^T \tag{A.5}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_M & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_M & \\ & & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_M \\ & & & \ddots & \ddots & & \ddots \end{bmatrix} \tag{A.6}$$

The $K \times N$ sub-matrices $\mathbf{G}_m$, $m = 0, 1, \ldots, M$, with elements from $GF(2)$ specify how an information bit block $\mathbf{b}_{i-m}$, delayed m steps in time, affects the code bit block $\mathbf{x}_i$:

$$\mathbf{x}_i = \sum_{m=0}^{M} \mathbf{G}_m \mathbf{b}_{i-m}^T, \forall i \tag{A.7}$$

For the code in Fig. A.1 two sub-matrices of size 1×2 have to be specified:

The sub-matrix $\mathbf{G}_0$ defines how u_i affects the generation of code bits $x_i^{(1)}, x_i^{(2)}$. Hence this matrix is $\mathbf{G}_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}$. The sub-matrix $\mathbf{G}_1$ identifies the code bits affected by the u_{i-1}, obviously, it equals $\mathbf{G}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$.

Alternatively, a convolutional code is defined by $K \times N$ polynomial generator matrix [56], [54]

$$\mathbf{G}(D) = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \cdots & g_1^{(N)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \cdots & g_2^{(N)}(D) \\ \vdots & \vdots & & \vdots \\ g_K^{(1)}(D) & g_K^{(2)}(D) & \cdots & g_K^{(N)}(D) \end{bmatrix} \tag{A.8}$$

where the matrix elements are polynomials over $GF(2)$

$$g_i^{(j)}(D) = g_{i,0}^{(j)} + g_{i,0}^{(j)} D + g_{i,2}^{(j)} D^2 + \cdots + g_{i,M}^{(j)} D^M \tag{A.9}$$

The j-th row of $\mathbf{G}(D)$ specifies how the j-th entry u_i^j of each information bit block $\mathbf{u}_i$ affects the code bit block $\mathbf{x}$.

The information bit sequence and code bit sequence can be described by vectors of polynomials

$$\mathbf{b} = (b^{(1)}(D), b^{(2)}(D), \cdots b^{(K)}(D)) \tag{A.10}$$

where $b^{(j)}(D) = b_0^{(j)} + b_1^{(j)} D + \cdots + b_i^{(j)} D^i + \cdots$, $j = 1, 2, \cdots, K$,

$$\mathbf{c} = (c^{(1)}(D), c^{(2)}(D), \cdots c^{(N)}(D)) \tag{A.11}$$

where $c^{(j)}(D) = c_0^{(j)} + c_1^{(j)} D + \cdots + c_i^{(j)} D^i + \cdots$, $j = 1, 2, \cdots, N$,

The encoding equation for the polynomial representation is therefore

$$\mathbf{c}(D) = \mathbf{G}(D)\mathbf{b}^T(D) \tag{A.12}$$

The polynomial matrix $\mathbf{G}(D)$ is obtained from $\mathbf{G}$ as follows:

$$g_{i,m}^{(j)} = g_m(i, j) \tag{A.13}$$

$i = 1, \cdots K$, $j = 1, \cdots, N$, m $= 0, \cdots, M$, where $g_m(i, j)$ is the (i, j)-th entry of $\mathbf{G}_m$. The code parameter M defines the largest degree of all polynomials $g_i^{(j)}(D)$.

For the example code in Fig. A.1, two generator polynomials $g_1^{(1)}(D)$ and $g_1^{(2)}(D)$ need to be constructed to specify $\mathbf{G}(D) = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) \end{bmatrix}$. The first polynomial specifies

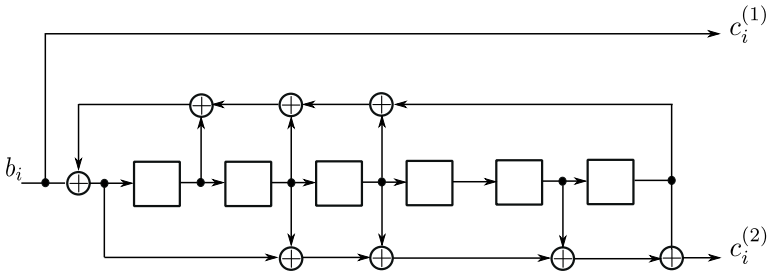


FIGURE A.2: Rate 1/2 (171, 133) recursive systematic encoder

how the information bit, shifted in time up to M time steps, affects the first code bit. As the first code bit is simply b_i and does not depend on b_{i-1}, the first generator polynomial is given as $g_1^{(1)}(D) = 1 + 0 \cdot D = 1$.

The second code bit is affected by both the current information bit b_i and the information bit, residing in the memory element, namely b_{i-1}. Therefore the generator polynomial is $g_1^{(2)}(D) = 1 + 1 \cdot D = 1 + D$. The generator polynomial matrix is commonly given in octal notation. In that format the polynomial generator matrix for the example code is

$$\mathbf{G}(d) = \begin{bmatrix} 1 & 1 + D \end{bmatrix} = \begin{bmatrix} 100_2 & 110_2 \end{bmatrix} = \begin{bmatrix} 4_8 & 6_8 \end{bmatrix} \tag{A.14}$$

The octal notation is practical as it specifies the information about the generator polynomial of the code and consequently the corresponding encoding circuit in compact form. For example, the industry standard rate 1/2 convolutional code is defined in octal notation as $(171, 133)$ [54]. Disregarding the leading zeros in the octal notation the generator polynomials are read as:

$$\begin{aligned} g_1^{(1)}(D) &= 171 = (00)1111001 = 1 + D + D^2 + D^3 + D^6 \\ g_1^{(2)}(D) &= 133 = (00)1011011 = 1 + D^2 + D^3 + D^5 + D^6 \end{aligned} \tag{A.15}$$

The resulting code is not systematic, as the first code bit is not just the current information bit itself. In order to obtain a systematic code, the non-systematic generator matrix has to be divided by the the first generator polynomial.

The resulting systematic polynomial generator matrix is

$$\mathbf{G}_{sys} = \begin{bmatrix} 1 & \frac{1+D^2+D^3+D^5+D^6}{1+D+D^2+D^3+D^6} \end{bmatrix} \tag{A.16}$$

The first code bit is just the current information bit itself, rendering the code systematic. The second polynomial is now a fraction of the two initial polynomials. The polynomial in the nominator specifies the feed-forward connections and the polynomial in the denominator specifies the feedback connection in the encoding circuit. Such systematic convolutional codes are denoted as *recursive systematic*.

The resulting recursive systematic circuit of the rate 1/2 (171, 133) convolutional code is depicted in Fig. A.2.

The decoding of convolutional codes is based on their trellis representation [55]. A trellis is a directed graph, whose nodes are labeled with the variables $s_i^{(j)} \in GF(2)$,

$j = 0, 1, \ldots, MK-1$. The vector $\mathbf{s}_i = \begin{bmatrix} s_i^{(0)} & s_i^{(1)} & \cdots & s_i^{(MK-1)} \end{bmatrix}$ combining all memory element contents at time step i is called the *state* of the encoder at time step i. Therefore, at each time step i there are $S = 2^{MK}$ nodes in a trellis. Each state node has 2^K incoming and outgoing branches. The branch labels contain the values of the information bit input to the encoder at time step i and the resulting code bit values.

In order to obtain the trellis of a given convolutional code it is first necessary to find out how state at each time step i depends on the information bit b_{i-m} and possibly on the state $s_{i-m}, m > 0$. In the example code depicted in Fig. A.1 the state is defined as $s_i = b_{i-1}$. Secondly, it must be defined how the code bits are generated from the information bits and the state values. In the example code (Fig. A.1), the first code bit is given by the information bit itself and does not depend on the state - $c_i^{(1)} = b_i$. The second code bit is computed from the current information bit and the value s_i stored in the memory element - $c_i^{(2)} = b_i + s_i = b_i + b_{i-1}$. Based on these two relationships the trellis section of the example code is depicted in Fig. A.1.

At the receiver the encoded data possibly containing errors is denoted $\mathbf{r}$. The convolutional codes are decoded by the Viterbi algorithm [56]. At each time step i corresponding to a section of the trellis the node and the branch metrics are computed. The branch metric is given as

$$\lambda_i = \sum_{j=1}^{N} \lambda_i^j \tag{A.17}$$

where the bit metric λ_i^j is the Hamming distance between the receive and the code bit

$$\lambda_i^j = \begin{cases} 0 & \text{when } c_i^{(j)} = r_i^{(j)} \\ 1 & \text{when } c_j^{(i)} \neq r_i^{(j)} \end{cases} \tag{A.18}$$

For each node the metrics for all 2^K incoming branches Λ_i are computed as follows:

$$\Lambda_i = \begin{cases} \lambda_i & , i = 0 \\ \Lambda_{i-1} + \lambda_i & , i > 0 \end{cases} \tag{A.19}$$

Out of 2^K branches merging into a node, choose the one with the smallest metric Λ_i. The other branches are discarded. The path through the trellis obtained so far is referred to as the survivor path. If some metrics are equal, the survivor path is chosen by some

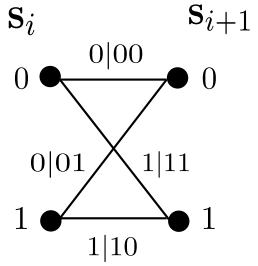


FIGURE A.3: Trellis section

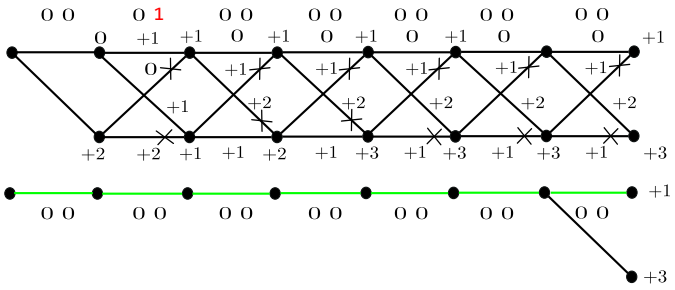


FIGURE A.4: Hard-decision decoding, one error

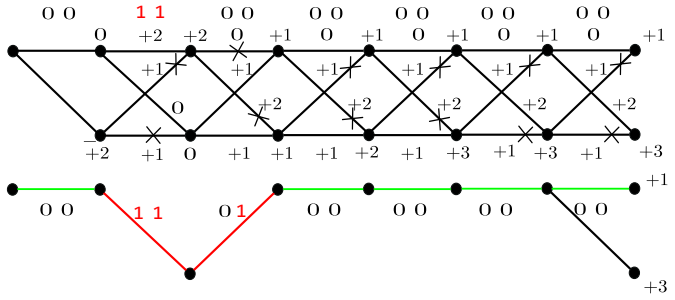


FIGURE A.5: Hard-decision decoding, two errors

arbitrary rule. For example, the upper/lower branch can be preferred or the branch can be picked randomly.

After the algorithm processed l trellis sections or more, the node with the smallest overall metric is chosen. The path which lead to this node through the trellis identifies the code bit sequence which has the minimum Hamming distance to the receive bit sequence $\mathbf{r}$. The branches of the final survivor path identify the output information bit sequence. The parameter l is the decision delay of the algorithm and specifies how many received symbols have to be processed until the first block of decoded bits is available. As a rule of thumb, the decision delay is often set to $5M$.

Consider an example code sequence $\mathbf{c} = \begin{bmatrix} 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$ output from the example 1/2 rate convolutional encoder in Fig. A.1. Assume that during the transmission an error has occurred as indicated in red in Fig. A.4 and the received code bit sequence is $\mathbf{r} = \begin{bmatrix} 00 & 10 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$. The branches are labeled with the branch metrics λ_i and the nodes are labeled with the Λ_i metric of the survivor path. The discarding of paths is shown by crossing. The algorithm moves through the trellis, and finally, there are two survivor paths with $\Lambda_7 = +1$ and $\Lambda_7 = +3$. The path with the smallest metric is picked as the final survivor and is traced back to the beginning of the trellis. The corresponding error corrected code sequence is obtained.

In case of multiple errors in the same transmitted code sequence, the received code sequence is $\mathbf{r} = \begin{bmatrix} 00 & 11 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$ as illustrated in Fig. A.5. It can be observed that again the final survivor path is the one with $\Lambda_7 = +1$, however, the errors caused three decoding errors due to the changes in the path. Finally, there is one information bit error.

In order to improve the performance, soft-decision decoding is employed [55]. The soft decision decoding uses the confidence information expressed as log-likelihood ratio (LLR) [192]. The LLR of a coded bit c_i is expressed as the ratio of probabilities of c_i being zero and one, respectively:

$$L(c_i) = \frac{P(c_i = 0)}{P(c_i = 1)} \tag{A.20}$$

Assume the following mapping of binary bit values to real values: $0 \rightarrow +1, 1 \rightarrow -1$. Using Bayes rule, the probability to guess the transmitted code bit c_i, given the received code bit r_i is expressed as:

$$P(c_i|r_i) = p(c_i, r_i)/p(r_i) = p(r_i|c_i)P(c_i)/p(r_i) \tag{A.21}$$

Therefore, the LLR of the transmitted code bit, given the received code bit is

$$\begin{aligned} L(c_i|r_i) &= \log \frac{P(c_i = +1|r_i)}{P(c_i = -1|r_i)} \\ &= \log \frac{p(r_i|c_i = +1)P(c_i = +1)/p(r_i)}{p(r_i|c_i = -1)P(c_i = -1)/p(r_i)} \\ &= \log \frac{p(r_i|c_i = +1)P(c_i = +1)}{p(r_i|c_i = -1)P(c_i = -1)} \end{aligned} \tag{A.22}$$

Assuming equal probability of transmitted code bits being $+1$ or -1, the LLR is further simplified

$$L(c_i|r_i) = \log \frac{p(r_i|c_i = +1)}{p(r_i|c_i = -1)} \tag{A.23}$$

In case of binary input additive white Gaussian noise (AWGN) channel, the receive code bit is given as [192]

$$r_i = c_i + n_i \tag{A.24}$$

where n_i is the AWGN noise sample, with zero mean and variance σ_n^2. The conditional probabilities in Eq. A.23 are therefore Gaussian

$$p(r_i|c_i = 1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(r_i - 1)^2}{2\sigma_n^2}\right) \tag{A.25}$$

$$p(r_i|c_i = -1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(\frac{(r_i + 1)^2}{2\sigma_n^2}\right) \tag{A.26}$$

The closed-form expression for LLR is then obtained as

$$\begin{aligned} L(c_i|r_i) &= \log \frac{\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(r_i-1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(r_i+1)^2}{2\sigma_n^2}\right)} \\ &= \log \exp\left(-\frac{(r_i - 1)^2}{2\sigma_n^2} + \frac{(r_i + 1)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{2\sigma_n^2}(-(r_i^2 - 2r_i + 1) + (r_i^2 + 2r_i + 1)) \\ &= \frac{2}{\sigma_n^2} r_i \end{aligned} \tag{A.27}$$

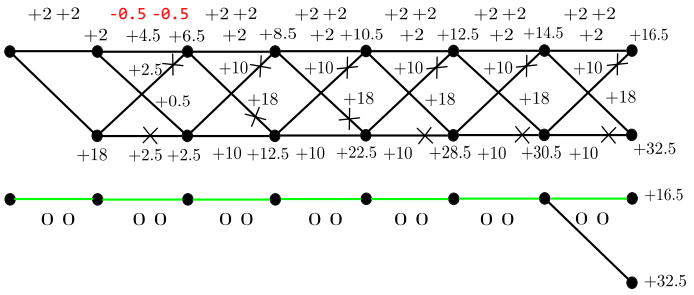


FIGURE A.6: Soft-decision decoding, two errors

The sign of the LLR gives the value of the of the transmitted code bit c_i, and the magnitude of LLR identifies the confidence in the taken decision. The positive sign of LLR indicates $p(c_i = 0|r_i)$ being larger than $p(c_i = 1|r_i)$, resulting in decision for $c_i = 0$. The magnitude of LLR of $+0.0043$ would give less confidence in the decision compared to the LLR of $+2.5$, as it indicates that the $p(c_i = 0|r_i)$ is just slightly larger than $p(c_i = 1|r_i)$. As Eq. A.27 indicates that LLR magnitude depends on the AWGN noise variance, in case of bad channel condition (large σ_n^2), the decision will be less certain that in case of good channel condition (small σ_n^2).

Consider the same transmitted code sequence mapped to real values

$$\mathbf{c} = \Big[+1+1 \quad +1+1 \quad +1+1 \quad +1+1 \quad +1+1 \quad +1+1 \quad +1+1 \quad +1+1\Big].$$

Assume that the errors happened due to bad channel conditions and the LLR values of the erroneous bits are -0.5. The correct bits are received while the channel was reliable and their LLR values equal $+2$. The receive code sequence is therefore $\mathbf{r} = \Big[+2+2 \quad -0.5-0.5 \quad +2+2 \quad +2+2 \quad +2+2 \quad +2+2 \quad +2+2 \quad +2+2\Big]$. The branch metric is now computed as Euclidean distance between the LLR of the received bit and the code bit.

$$\lambda_i^j = (L(r_i) - c_i)^2 \tag{A.28}$$

The soft decision decoding is illustrated in Fig. A.6. It can be observed that, with Euclidean distance metric computation, the correct path is taken through the trellis and the errors are corrected.

Bibliography

[1] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson. Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays. *IEEE Signal Processing Magazine*, 30(1):40–60, Jan 2013.

[2] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer. Large-Scale MIMO Detection for 3GPP LTE: Algorithms and FPGA Implementations. *IEEE Journal of Selected Topics in Signal Processing*, 8(5):916–929, Oct 2014.

[3] M. Stanisavljevic, A. Schmid, and Y. Leblebici. *Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures*. Springer New York, 2010.

[4] G. Karakonstantis, A. Chatterjee, and K. Roy. Containing the Nanometer Pandora-Box: Cross-Layer Design Techniques for Variation Aware Low Power Systems. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 1(1):19–29, March 2011.

[5] Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2007.

[6] M.S. Khairy, Chung-An Shen, A.M. Eltawil, and F. Kurdahi. Error resilient MIMO detector for memory-dominated wireless communication systems. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 3566–3571, Dec 2012.

[7] M.S. Khairy, Chung-An Shen, A.M. Eltawil, and F.J. Kurdahi. Algorithms and Architectures of Energy-Efficient Error-Resilient MIMO Detectors for Memory-Dominated Wireless Communication Systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 61(7):2159–2171, July 2014.

[8] M. Karpovsky and A. Taubin. New class of nonlinear systematic error detecting codes. *IEEE Transactions on Information Theory*, 50(8):1818–1819, Aug 2004.

[9] M. Huemer, C. Hofbauer, and J. B. Huber. The Potential of Unique Words in OFDM. In *Proceedings of the 15th International OFDM-Workshop 2010 (InOWo' 10)*, pages 140 – 144, September 2010.

[10] C. Hofbauer and M. Huemer. A study of data rate equivalent UW-OFDM and CP-OFDM concepts. In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pages 173–177, Nov 2012.

[11] M. Huemer, C. Hofbauer, and J.B. Huber. Non-Systematic Complex Number RS Coded OFDM by Unique Word Prefix. *Signal Processing, IEEE Transactions on*, 60(1):285–299, Jan 2012.

[12] A. Onic and M. Huemer. Sphere Decoding for Unique Word OFDM. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5, Dec 2011.

[13] Alexander Onic. *Receiver Concepts for Unique Word OFDM.* PhD thesis, Johannes Keppler Universitaet Linz, 2013.

[14] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 7–18, Dec 2003.

[15] S. Ghosh, S. Bhunia, and K. Roy. CRISTA: A New Paradigm for Low-Power, Variation-Tolerant, and Adaptive Circuit Synthesis Using Critical Path Isolation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(11):1947–1956, Nov 2007.

[16] G. Karakonstantis, N. Banerjee, and K. Roy. Process-Variation Resilient and Voltage-Scalable DCT Architecture for Robust Low-Power Computing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(10):1461–1470, Oct 2010.

[17] Jung Hwan Choi, N. Banerjee, and K. Roy. Variation-Aware Low-Power Synthesis Methodology for Fixed-Point FIR Filters. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(1):87–97, Jan 2009.

[18] V. Gupta, G. Karakonstantis, D. Mohapatra, and K. Roy. VEDA: Variation-aware energy-efficient Discrete Wavelet Transform architecture. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 260–265, Oct 2010.

[19] N.R. Shanbhag, R.A. Abdallah, R. Kumar, and D.L. Jones. Stochastic computation. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 859–864, June 2010.

[20] R. Hegde and N.R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*, pages 30–35, Aug 1999.

[21] R. Hegde and N.R. Shanbhag. Soft digital signal processing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(6):813–823, Dec 2001.

[22] E.P. Kim and N.R. Shanbhag. Soft N-Modular Redundancy. *Computers, IEEE Transactions on*, 61(3):323–336, March 2012.

[23] Byonghyo Shim, S.R. Sridhara, and N.R. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(5):497–510, May 2004.

[24] S. Narayanan. *Estimation-theoretic Framework for Robust and Energy-efficient System Design*. PhD thesis, University of Illinois, 2010.

[25] M.S. Khairy, A. Khajeh, A.M. Eltawil, and F.J. Kurdahi. FFT processing through faulty memories in OFDM based systems. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1946–1951, Dec 2010.

[26] M. May, M. Alles, and N. Wehn. A Case Study in Reliability-Aware Design: A Resilient LDPC Code Decoder. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 456–461, March 2008.

[27] C. Novak, C. Studer, A. Burg, and G. Matz. The effect of unreliable LLR storage on the performance of MIMO-BICM. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 736–740, Nov 2010.

[28] Christina Gimmler-Dumont and Norbert Wehn. A Cross-Layer Reliability Design Methodology for Efficient, Dependable Wireless Receivers. *ACM Trans. Embed. Comput. Syst.*, 13(4s):137:1–137:29, April 2014.

[29] Christina Gimmler-Dumont, Christian Brehm, and Norbert Wehn. Reliability study on system memories of an iterative MIMO-BICM system. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 255 –258, oct. 2012.

[30] H. Bar-El et al. The Sorcerer's Apprentice Guide to Fault Attacks. *Proc. IEEE*, 94:370–382, 2006.

[31] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault Injection Attacks on Cryptographic Devices: Theory, Practice and Countermeasures. *Proc. IEEE*, 99, 2012.

[32] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Annual Int'l Cryptology Conf. (LNCS 1294)*, pages 513–525, 1997.

[33] O. Derouet. Secure smartcard design against laser fault injection attacks. In *Works. Fault Diagnosis and Tolerance in Cryptography*, 2007.

[34] A. Dehbaoui et al. Electromagnetic transient faults in-jection on a hardware and a software implementations of AES. In *Works. Fault Diagnosis and Tolerance in Cryptography*, pages 7–15, 2007.

[35] P. Jovanovic, M. Kreuzer, and I. Polian. A Fault Attack on the LED Block Cipher. In *Int'l Workshop on Constructive Side-channel Analysis and Secure Design (LNCS 7275)*, pages 120–134, 2012.

[36] Chong Hee Kim and Jean-Jacques Quisquater. *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems: First IFIP TC6 / WG 8.8 / WG 11.2 International Workshop, WISTP 2007, Heraklion, Crete, Greece, May 9-11, 2007. Proceedings*, chapter Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures, pages 215–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[37] M. Tunstall, D. Mukhopadhyay, and S. Ali. Differential fault analysis of the Advanced Encryption Standard using a single fault. In *Workshop in Information Security Theory and Practice (LNCS 6633)*, pages 224–233, 2011.

[38] M. Tunstall, D. Mukhopadhyay, and S. Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. *LNCS*, 6633:pp. 224–233, 2011.

[39] Philipp Jovanovic, Martin Kreuzer, and Ilia Polian. Multi-Stage Fault Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2013/778, 2013.

[40] Nasour Bagheri, Reza Ebrahimpour, and Navid Ghaedi. New Differential Fault Analysis of PRESENT. *EURASIP Journal on Advances in Signal Processing*, 2013 (1):1–10, 2013.

[41] Ling Song and Lei Hu. Differential Fault Attack on the PRINCE Block Cipher. Cryptology ePrint Archive, Report 2013/043, 2013.

[42] Victor Tomashevich and Ilia Polian. Detection Performance of MIMO Unique Word OFDM. In *20th ITG Workshop on Smart Antennas*, 2016.

[43] Victor Tomashevich and Ilia Polian. Memory Error Resilient Detection for Massive MIMO Systems. Submitted to the 2016 European Signal Processing Conference (EUSIPCO), 2016.

[44] Victor Tomashevich, Christina Gimmler-Dumont, Christian Fesl, Norbert Wehn, and Ilia Polian. A New Architecture for Minimum Mean Square Error Sorted QR Decomposition for MIMO Wireless Communication Systems. In *Design and Diagnostics of Electronic Circuits and Systems, 2014 IEEE 17th Symposium on*, pages 246 –250, april 2014.

[45] V. Tomashevich, C. Gimmler-Dumont, N. Wehn, and I. Polian. Reliability analysis of MIMO channel preprocessing by fault injection. In *Wireless for Space and Extreme Environments (WiSEE), 2014 IEEE International Conference on*, pages 1–6, Oct 2014.

[46] V. Tomashevich, S. Srinivasan, F. Foerg, , and I. Polian. Cross-level protection of circuits against faults and malicious attacks. In *IEEE Int'l On-Line Test Symp.*, 2012.

[47] V. Tomashevich, Y. Neumeier, R. Kumar, O. Keren, and I. Polian. Protecting cryptographic hardware against malicious attacks by nonlinear robust codes. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*, pages 40–45, Oct 2014.

[48] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, New York, NY, USA, 2005.

[49] Proakis. *Digital Communications 5th Edition*. McGraw Hill, 2007.

[50] Gordon L. Stüber. *Principles of Mobile Communication (2Nd Ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[51] IEEE. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, Oct 2009.

[52] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals and Systems (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[53] Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan, editors. *Simulation of Communication Systems: Modeling, Methodology and Techniques*. Kluwer Academic Publishers, Norwell, MA, USA, 2nd edition, 2000.

[54] Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

[55] Shu Lin and Daniel J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[56] Martin Bossert. *Channel Coding for Telecommunications*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[57] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[58] C. Hofbauer, M. Huemer, and J.B. Huber. Coded OFDM by unique word prefix. In *Communication Systems (ICCS), 2010 IEEE International Conference on*, pages 426–430, Nov 2010.

[59] A. Alexiou and M. Haardt. Smart antenna technologies for future wireless systems: trends and challenges. *Communications Magazine, IEEE*, 42(9):90–97, Sept 2004.

[60] P.F. Driessen and G.J. Foschini. On the capacity formula for multiple input-multiple output wireless channels: a geometric interpretation. *Communications, IEEE Transactions on*, 47(2):173–176, Feb 1999.

[61] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, New York, NY, USA, 2005.

[62] Jerry R. Hampton. *Introduction to MIMO Communications*. Cambridge University Press, 2013. Cambridge Books Online.

[63] Juho Lee, Jin-Kyu Han, and Jianzhong Zhang. MIMO Technologies in 3GPP LTE and LTE-advanced. *EURASIP J. Wirel. Commun. Netw.*, 2009:3:1–3:10, March 2009.

[64] R. Adams. Simplified Baseband Diversity Combiner. *Communications Systems, IRE Transactions on*, 8(4):247–249, December 1960.

[65] M. Schwartz, W. Bennett, and S. Stein. *Linear Diversity Combining Techniques*, pages 416–489. Wiley-IEEE Press, 1996.

[66] T. Eng, Ning Kong, and L.B. Milstein. Comparison of diversity combining techniques for Rayleigh-fading channels. *Communications, IEEE Transactions on*, 44 (9):1117–1129, Sep 1996.

[67] B. Solaiman, A. Glavieux, and A. Hillion. Equal gain diversity improvement in fast frequency hopping spread spectrum multiple-access (FFH-SSMA) communications over Rayleigh fading channels. *Selected Areas in Communications, IEEE Journal on*, 7(1):140–147, Jan 1989.

[68] T.K.Y. Lo. Maximum ratio transmission. *Communications, IEEE Transactions on*, 47(10):1458–1461, Oct 1999.

[69] S. Alamouti. A simple transmit diversity technique for wireless communications. *Selected Areas in Communications, IEEE Journal on*, 16(8):1451–1458, Oct 1998.

[70] Vahid Tarokh, N. Seshadri, and A.R. Calderbank. Space-time codes for high data rate wireless communication: performance criterion and code construction. *Information Theory, IEEE Transactions on*, 44(2):744–765, Mar 1998.

[71] Lizhong Zheng and D.N.C. Tse. Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels. *Information Theory, IEEE Transactions on*, 49 (5):1073–1096, May 2003.

[72] P.W. Wolniansky, G.J. Foschini, G.D. Golden, and R. Valenzuela. V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel. In *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*, pages 295–300, Sep 1998.

[73] Tim Brown, Elisabeth De Carvalho, and Persefoni Kyritsi. *Practical Guide to the MIMO Radio Channel with MATLAB Examples*. Wiley, 1 edition, 2012.

[74] P. Almers, E. Bonek, A. Burr, N. Czink, M. Debbah, V. Degli-Esposti, H. Hofstetter, P. Kyosti, D. Laurenson, G. Matz, A. Molisch, C. Oestges, and H. Ozcelik. Survey of Channel and Radio Propagation Models for Wireless MIMO Systems. *EURASIP Journal on Wireless Communications and Networking*, 2007(1):019070, 2007.

[75] Vinko Erceg, Laurent Schumacher, and Persefoni Kyritsi. IEEE P802.11 Wireless LANs*, TGn Channel Models*. IEEE 802.11-03/940r4, May 2004.

[76] M. Debbah and R. Muller. MIMO channel modeling and the principle of maximum entropy. *Information Theory, IEEE Transactions on*, 51(5):1667–1690, May 2005.

[77] L. Schumacher, K.I. Pedersen, and P.E. Mogensen. From antenna spacings to theoretical capacities - guidelines for simulating MIMO systems. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, volume 2, pages 587–592 vol.2, Sept 2002.

[78] K.I. Pedersen, P.E. Mogensen, and B.H. Fleury. A stochastic model of the temporal and azimuthal dispersion seen at the base station in outdoor propagation environments. *Vehicular Technology, IEEE Transactions on*, 49(2):437–447, Mar 2000.

[79] J.P. Kermoal, L. Schumacher, K.I. Pedersen, P.E. Mogensen, and F. Frederiksen. A stochastic MIMO radio channel model with experimental validation. *Selected Areas in Communications, IEEE Journal on*, 20(6):1211–1226, Aug 2002.

[80] C. Michalke, E. Zimmermann, and G. Fettweis. Linear Mimo Receivers vs. Tree Search Detection: A Performance Comparison Overview. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pages 1–7, Sept 2006.

[81] D. Seethaler and F. Hlawatsch. Detection techniques for MIMO spatial multiplexing systems. *Elektrotechnik und Informationstechnik*, 122(3):91–96, 2005.

[82] Sergio Verdú. Computational Complexity of Optimum Multiuser Detection. *ALGORITHMICA*, 4(3):303–312, 1989.

[83] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[84] David G. Luenberger. *Optimization by Vector Space Methods.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[85] W. Koch and A. Baier. Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference. In *Global Telecommunications Conference, 1990, and Exhibition. 'Communications: Connecting the Future', GLOBECOM '90., IEEE*, pages 1679–1684 vol.3, Dec 1990.

[86] I.B. Collings, M.R.G. Butler, and M.R. McKay. Low complexity receiver design for MIMO bit-interleaved coded modulation. In *Spread Spectrum Techniques and Applications, 2004 IEEE Eighth International Symposium on*, pages 12–16, Aug 2004.

[87] U. Fincke and M. Pohst. Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis. *Mathematics of Computation*, 44(170):463–471, 1985.

[88] Wai Ho Mow. Maximum likelihood sequence estimation from the lattice viewpoint. *Information Theory, IEEE Transactions on*, 40(5):1591–1600, Sep 1994.

[89] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *Information Theory, IEEE Transactions on*, 48(8):2201–2214, Aug 2002.

[90] O. Damen, A. Chkeif, and J.-C. Belfiore. Lattice code decoder for space-time codes. *Communications Letters, IEEE*, 4(5):161–163, May 2000.

[91] M.O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *Information Theory, IEEE Transactions on*, 49(10):2389–2402, Oct 2003.

[92] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *Signal Processing, IEEE Transactions on*, 53(8):2806–2818, Aug 2005.

[93] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei. VLSI implementation of MIMO detection using the sphere decoding algorithm. *Solid-State Circuits, IEEE Journal of*, 40(7):1566–1577, July 2005.

[94] Steven Skiena. *Implementing discrete mathematics - combinatorics and graph theory with Mathematica*. Addison-Wesley, 1990.

[95] C. P. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In *Math. Programming*, pages 181–191, 1993.

[96] R. Wang and G.B. Giannakis. Approaching MIMO channel capacity with soft detection based on hard sphere decoding. *Communications, IEEE Transactions on*, 54(4):587–590, April 2006.

[97] C. Studer and H. Bolcskei. Soft Input Soft Output Single Tree-Search Sphere Decoding. *Information Theory, IEEE Transactions on*, 56(10):4827–4842, Oct 2010.

[98] A. Chockalingam. Low-complexity algorithms for large-MIMO detection. In *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, pages 1–6, March 2010.

[99] S.K. Mohammed, A. Chockalingam, and B. Sundar Rajan. A Low-complexity near-ML performance achieving algorithm for large MIMO detection. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 2012–2016, July 2008.

[100] Peng Li and R.D. Murch. Multiple output selection-LAS algorithm in large MIMO systems. *Communications Letters, IEEE*, 14(5):399–401, May 2010.

[101] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[102] R. Hegde and N.R. Shanbhag. A voltage overscaled low-power digital filter IC. *Solid-State Circuits, IEEE Journal of*, 39(2):388–391, Feb 2004.

[103] G. Karakonstantis and K. Roy. Voltage over-scaling: A cross-layer design perspective for energy efficient systems. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 548–551, Aug 2011.

[104] R. Hegde and N.R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. In *Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on*, pages 30–35, Aug 1999.

[105] John F. Wakerly. *Digital Design: Principles and Practices*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 3rd edition, 2000.

[106] P.E. Dodd and L.W. Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *Nuclear Science, IEEE Transactions on*, 50(3): 583–602, June 2003.

[107] E. Normand. Single event upset at ground level. *Nuclear Science, IEEE Transactions on*, 43(6):2742–2750, Dec 1996.

[108] R.C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *Device and Materials Reliability, IEEE Transactions on*, 5(3):305–316, Sept 2005.

[109] R. Baumann. Soft errors in advanced computer systems. *Design Test of Computers, IEEE*, 22(3):258–266, May 2005.

[110] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 389–398, 2002.

[111] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *Nuclear Science, IEEE Transactions on*, 47(6): 2586–2594, Dec 2000.

[112] Kuang-Hua Huang and J.A. Abraham. Algorithm-Based Fault Tolerance for Matrix Operations. *Computers, IEEE Transactions on*, C-33(6):518–528, June 1984.

[113] V.S.S. Nair and J.A. Abraham. General linear codes for fault-tolerant matrix operations on processor arrays. In *Fault-Tolerant Computing, 1988. FTCS-18, Digest of Papers., Eighteenth International Symposium on*, pages 180–185, June 1988.

[114] C.J. Anfinson and F.T. Luk. A linear algebraic model of algorithm-based fault tolerance. *Computers, IEEE Transactions on*, 37(12):1599–1604, Dec 1988.

[115] V.S.S. Nair and J.A. Abraham. Real-number codes for fault-tolerant matrix operations on processor arrays. *Computers, IEEE Transactions on*, 39(4):426–435, Apr 1990.

[116] A.A. Al-Yamani, N. Oh, and E.J. McCluskey. Performance evaluation of checksum-based ABFT. In *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*, pages 461–466, 2001.

[117] B. R. Gaines. Stochastic Computing. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 149–156, New York, NY, USA, 1967. ACM.

[118] Weikang Qian, Xin Li, M.D. Riedel, K. Bazargan, and D.J. Lilja. An Architecture for Fault-Tolerant Computation with Stochastic Logic. *Computers, IEEE Transactions on*, 60(1):93–105, Jan 2011.

[119] S.S. Tehrani, S. Mannor, and W.J. Gross. Fully Parallel Stochastic LDPC Decoders. *Signal Processing, IEEE Transactions on*, 56(11):5692–5703, Nov 2008.

[120] A. Alaghi, Cheng Li, and J.P. Hayes. Stochastic circuits for real-time image-processing applications. In *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pages 1–6, May 2013.

[121] E.P. Kim and N.R. Shanbhag. Statistical analysis of algorithmic noise tolerance. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2731–2735, May 2013.

[122] R. Hegde and N.R. Shanbhag. A low-power digital filter IC via soft DSP. In *Custom Integrated Circuits, 2001, IEEE Conference on.*, pages 309–312, 2001.

[123] S.R. Sridhara and N.R. Shanbhag. Low-power FFT via reduced precision redundancy. In *Signal Processing Systems, 2001 IEEE Workshop on*, pages 117–124, 2001.

[124] R.A. Abdallah and N.R. Shanbhag. Error-Resilient Low-Power Viterbi Decoder Architectures. *Signal Processing, IEEE Transactions on*, 57(12):4906–4917, Dec 2009.

[125] Byonghyo Shim and N.R. Shanbhag. Energy-efficient soft error-tolerant digital signal processing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(4):336 –348, april 2006.

[126] A. Khajeh, K. Amiri, M.S. Khairy, A.M. Eltawil, and F.J. Kurdahi. A Unified Hardware and Channel Noise Model for Communication Systems. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Dec 2010.

[127] A.M.A. Hussien, M.S. Khairy, A. Khajeh, K. Amiri, A.M. Eltawil, and F.J. Kurdahi. A combined channel and hardware noise resilient Viterbi decoder. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 395–399, Nov 2010.

[128] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(12):1859–1880, Dec 2005.

[129] A.K. Djahromi, A.M. Eltawil, F.J. Kurdahi, and R. Kanj. Cross Layer Error Exploitation for Aggressive Voltage Scaling. In *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, pages 192–197, March 2007.

[130] K. Agarwal and S. Nassif. The Impact of Random Device Variation on SRAM Cell Stability in Sub-90-nm CMOS Technologies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(1):86–97, Jan 2008.

[131] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[132] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM side-channel(s). In *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 2523)*, pages 29–45, 2003.

[133] Kahraman D. Akdemir, Zhen Wang, Mark Karpovsky, and Berk Sunar. *Fault Analysis in Cryptography*, chapter Design of Cryptographic Devices Resilient to Fault Injection Attacks Using Nonlinear Robust Codes, pages 171–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[134] D. Boneh, R.A. DeMillo, and R.J. Lipton. On the importance of elimination errors in cryptographic computations. *Jour. Cryptology*, 14:101–119, 2001.

[135] D. Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. *LNCS*, 5580:pp. 421–434, 2009.

[136] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The LED Block Cipher. *LNCS*, 6917:pp. 326–341, 2011.

[137] A. Bogdanov et al. PRESENT: An ultra-lightweight block cipher. In *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 4727)*, pages 450–466, 2007.

[138] A. Barenghi, G. M. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi. Low voltage fault attacks to AES. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 7–12, June 2010.

[139] Frederic Amiel, Christophe Clavier, and Michael Tunstall. *Fault Diagnosis and Tolerance in Cryptography: Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006. Proceedings*, chapter Fault Analysis of DPA-Resistant Algorithms, pages 223–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[140] Sudhakar Govindavajhala and Andrew W. Appel. Using Memory Errors to Attack a Virtual Machine. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, pages 154–, Washington, DC, USA, 2003. IEEE Computer Society.

[141] L. Anghel and M. Nicolaidis. Cost reduction and evaluation of a temporary faults detecting technique. In *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, pages 591–598, 2000.

[142] J. H. Patel and L. Y. Fung. Concurrent Error Detection in ALU's by Recomputing with Shifted Operands. *IEEE Transactions on Computers*, C-31(7):589–595, July 1982.

[143] A. Chockalingam and B. Sundar Rajan. *Large MIMO Systems*. Cambridge University Press, 2014. Cambridge Books Online.

[144] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn. A 477mW NoC-based digital baseband for MIMO 4G SDR. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 278–279, Feb 2010.

[145] William W. Hager. Updating the Inverse of a Matrix. *SIAM Review*, 31(2):221–239, 1989.

[146] B.M. Hochwald and S. ten Brink. Achieving near-capacity on a multiple-antenna channel. *Communications, IEEE Transactions on*, 51(3):389–399, March 2003.

[147] *MATLAB version 8.5.0.197613 (R2015a) Documentation*. The Mathworks, Inc., Natick, Massachusetts, 2015.

[148] H. W. Sorenson and D. L. Alspach. Recursive Bayesian Estimation Using Gaussian Sums. *Automatica*, 7(4):465–479, July 1971.

[149] Jonathan Q. Li and Andrew R. Barron. Mixture Density Estimation. In *IN ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 12*, pages 279–285. MIT Press, 1999.

[150] A. Kundu, S. Chatterjee, A. Sreenivasa Murthy, and T.V. Sreenivas. GMM based Bayesian approach to speech enhancement in signal / transform domain. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4893–4896, March 2008.

[151] J.T. Flam, S. Chatterjee, K. Kansanen, and T. Ekman. On MMSE Estimation: A Linear Model Under Gaussian Mixture Statistics. *Signal Processing, IEEE Transactions on*, 60(7):3840–3845, July 2012.

[152] J.T. Flam, D. Zachariah, M. Vehkapera, and S. Chatterjee. The Linear Model Under Mixed Gaussian Inputs: Designing the Transfer Matrix. *Signal Processing, IEEE Transactions on*, 61(21):5247–5259, Nov 2013.

[153] M. Karkooti, J. R. Cavallaro, and C. Dick. FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm. In *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, pages 1625–1629, October 2005.

[154] Zheng-Yu Huang and Pei-Yun Tsai. Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(10):2531 –2542, oct. 2011.

[155] Gabriel Luca Nazar, Christina Gimmler, and Norbert Wehn. Implementation comparisons of the QR decomposition for MIMO detection. In *Proceedings of the 23rd symposium on Integrated circuits and system design*, SBCCI '10, pages 210–214, New York, NY, USA, 2010. ACM.

[156] D. Wubben, R. Bohnke, V. Kuhn, and K. D. Kammeyer. MMSE extension of V-BLAST based on sorted QR decomposition. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 1, pages 508–512 Vol.1, Oct 2003.

[157] W. Givens. Computation of Plain Unitary Rotations Transforming a General Matrix to Triangular Form. *Journal of the Society for Industrial and Applied Mathematics*, 6(1):26–50, 1958.

[158] Richard Lewis Walke. *High sample-rate Givens rotations for recursive least squares*. PhD thesis, University of Warwick, 1997.

[159] R. Doehler. Squared Givens Rotation. *IMA Journal of Numerical Analysis*, 11(1): 1–5, 1991.

[160] Lei Ma, K. Dickson, J. McAllister, and J. Mccanny. QR Decomposition-Based Matrix Inversion for High Performance Embedded MIMO Receivers. *Signal Processing, IEEE Transactions on*, 59(4):1858 –1867, april 2011.

[161] J. Goetze and U. Schwiegelshohn. A Square Root and Division Free Givens Rotation for Solving Least Squares Problems on Systolic Arrays. *SIAM J. Sci. Stat. Comput.*, 12(4):800–807, May 1991.

[162] P. Luethi, A. Burg, S. Haene, D. Perels, N. Felber, and W. Fichtner. VLSI Implementation of a High-Speed Iterative Sorted MMSE QR Decomposition. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 1421 –1424, may 2007.

[163] D. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K. D. Kammeyer. Efficient algorithm for decoding layered space-time codes. *Electronics Letters*, 37(22):1348–1350, Oct 2001.

[164] P. Salmela, A. Burian, H. Sorokin, and J. Takala. Complex-valued QR decomposition implementation for MIMO receivers. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 1433–1436, 2008.

[165] *IT++ 4.3.1 user guide*, 2014.

[166] D. May and W. Stechele. An FPGA-based probability-aware fault simulator. In *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pages 302–309, July 2012.

[167] Veit B. Kleeberger, Christina Gimmler-Dumont, Christian Weis, Andreas Herkersdorf, Daniel Mueller-Gritschneder, Sani R. Nassif, Ulf Schlichtmann, and Norbert Wehn. A Cross-Layer Technology-Based Study of How Memory Errors Impact System Resilience. *IEEE Micro*, 33(4):46–55, 2013.

[168] M. Gössel, V. Ocheretny, E. Sogomonyan, and D. Marienfeld. *New Methods of Concurrent Checking*. Springer, 2008.

[169] M. Karpovsky, K. J. Kulikowski, and A. Taubin. Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In *Dependable Systems and Networks, 2004 International Conference on*, pages 93–101, June 2004.

[170] Konrad J. Kulikowski, Mark G. Karpovsky, and Er Taubin. Robust Codes for Fault Attack Resistant Cryptographic Hardware. In *Fault Diagnosis and Tolerance in Cryptography, 2nd International Workshop*, pages 1–12, 2005.

[171] Zhen Wang, Mark Karpovsky, and Konrad J Kulikowski. Design of memories with concurrent error detection and correction by nonlinear SEC-DED codes. *Journal of Electronic Testing*, 26(5):559–580, 2010.

[172] Konrad J Kulikowski, Zhen Wang, and Mark G Karpovsky. Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems. In *Works. Fault Diagnosis and Tolerance in Cryptography*, pages 41–50. IEEE, 2008.

[173] Mark Karpovsky and Alexander Taubin. New class of nonlinear systematic error detecting codes. *Information Theory, IEEE Transactions on*, 50(8):1818–1819, 2004.

[174] Konrad J Kulikowski, Mark G Karpovsky, and Alexander Taubin. Fault attack resistant cryptographic hardware with uniform error detection. In *Works. Fault Diagnosis and Tolerance in Cryptography*, pages 185–195. Springer, 2006.

[175] Mark G Karpovsky, Konrad J Kulikowski, and Zhen Wang. Robust Error Detection in Communication and Computational Channels. *Int'l Workshop on Spectral Techniques*, 2007.

[176] Yaara Neumeier and Osnat Keren. Punctured Karpovsky-Taubin binary robust error detecting codes for cryptographic devices. In *IEEE Int'l On-Line Testing Symp.*, pages 156–161. IEEE, 2012.

[177] Yaara Neumeier and Osnat Keren. Robust Generalized Punctured Cubic Codes. *IEEE Transactions on Information Theory*, 2013.

[178] Jorge Guajardo, Tim Güneysu, Sandeep S Kumar, Christof Paar, and Jan Pelzl. Efficient hardware implementation of finite fields with applications to cryptography. *Acta Appl. Math.*, 93(1-3):75–118, 2006.

[179] Nir Admaty, Simon Litsyn, and Osnat Keren. Puncturing, expurgating and expanding the q-ary BCH based robust codes. In *IEEE Convention of Electrical & Electronics Engineers in Israel*, pages 1–5. IEEE, 2012.

[180] Shubu Mukherjee. *Architecture Design for Soft Errors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[181] S. Frehse, G. Fey, and R. Drechselr. A Better-than-worst-case Robustness Measure. In *DDECS*, 2010.

[182] J. E. Smith and P. Lam. A Theory of Totally Self-Checking System Design. *IEEE Transactions on Computers*, C-32(9):831–844, Sept 1983.

[183] *Nangate Open cell library, www.nangate.com*, 2012.

[184] Quming Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(1):155–166, Jan 2006.

[185] B. S. Gill, C. Papachristou, F. G. Wolff, and N. Seifert. Node sensitivity analysis for soft errors in CMOS logic. In *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, pages 9 pp.–972, Nov 2005.

[186] M. P. Baze and S. P. Buchner. Attenuation of single event induced pulses in CMOS combinational logic. *IEEE Transactions on Nuclear Science*, 44(6):2217–2223, Dec 1997.

[187] P. Bhattacharya. *Architecture and algorithm for mitigating soft errors in nanoscale VLSI circuits*. PhD thesis, University of south Florida, 2009.

[188] Stephen D. Brown, Zvonko G. Vranesic, and Zvonko Vranesic. *Fundamentals of Digital Logic with VHDL Design*. McGraw-Hill Higher Education, 1st edition, 1999.

[189] Gregory C. Ahlquist, Brent Nelson, and Michael Rice. *Field Programmable Logic and Applications: 9th International Workshop, FPL'99, Glasgow, UK, August 30 - September 1, 1999. Proceedings*, chapter Optimal Finite Field Multipliers for FPGAs, pages 51–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[190] M. G. Karpovsky, K. J. Kulikowski, and Z. Wang. Robust error detection in communication and computation channels. In *Int'l Workshop on Spectral Techniques*, 2007.

[191] J. Borghoff et al. PRINCE: A low-latency block cipher for pervasive computing applications. In *ASIACRYPT*, 2012.

[192] Sarah J. Johnson. *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat- Accumulate Codes*. Cambridge University Press, jan 2010.