Gregor Hauseder, BSc

# Design and RTL Implementation of a Channel Equalizer for NFC High Data Rate Applications

**MASTER'S THESIS**

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

**Graz University of Technology**

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dipl.-Ing. Dr. techn. Ulrich Mühlmann (NXP Semiconductors Austria GmbH)

Institute for Technical Informatics

Graz, Mai 2020

# Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present thesis.

| | |
|---|---|
| Date | Signature |

# Kurzfassung

Im Kontext der Kommunikationstechnologie werden Signale während der Übertragung verzerrt. Verzerrung bedeutet hierbei ein Absinken der Signalqualität. Bei der Übertragungstechnologie *Near Field Communication* (NFC) führt vor allem die Intersymbolinterferenz (ISI) zu Verzerrungen. Außerdem wird die Phase des Signals bei der Übertragung verschoben. Bei Datenraten höher als 848kbps wird das Signal so stark verzerrt, dass keine robuste Übertragung mehr möglich ist. In dieser Arbeit wurde untersucht, inwieweit ein Equalizer basierend auf dem *well-behaved Normalized Constant Modulus Algorithm* (wNCMA) der ISI, als auch der Phasenverschiebung, entgegenwirken kann. Zusätzlich wurde der Equalizer in Verilog implementiert.

# Abstract

In the context of communication technology, signals always experience some form of degradation during transmission. Degradation is the loss of quality of an electronic signal. Regarding Near Field Communication, the main cause of degradation is Intersymbol Interference (ISI). The transmitted signal also experiences a phase shift. For data rate applications greater than 848kbps, ISI becomes severe, preventing robust communication. For the mitigation of ISI and phase shifts, the use of the well-behaved Normalized Constant Modulus Algorithm (wNCMA) was investigated and an equalizer based on this algorithm implemented in Verilog.

# Acknowledgement

I want to thank Prof. Christian Steger from the Institute of Technical Informations at TU Graz, as well as my mentors Dr. Ulrich Mühlmann and Dipl.Ing. Wolfgang Hrauda, and the rest of my colleagues from NXP Semiconductors in Gratkorn. A special thanks goes to Wolfgang for his continual support over the course of this thesis. He always took time for me in case I faced issues and for that I am deeply grateful. Also, there was always a healthy amount of fun included in our discussions, which I enjoyed a lot. Most importantly, I want to thank my girlfriend Elisabetta for her love, understanding and support.

Graz, im Monat Jahr Name des Diplomanden

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Motivation and Introduction

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.
>
> *Claude E. Shannon*
> *A Mathematical Theory of Communication*

## 1.1 Motivation

It was 1948 when Claude E. Shannon's groundbreaking work *A Mathematical Theory of Communication* was published in *Bell Systems Technical Journal*. In that article, Shannon described the basic elements of a general communication system, depicted in Figure 1.1. First, the information source forwards a message to the transmitter. The transmitter then operates on the message to produce a signal suitable for transmission. The signal is then traveling through the communication channel, where it gets corrupted by noise. Lastly, the receiver is restoring the information from the received signal and forwards it to the destination.

A specific kind of communication system is the NFC system. Such systems are explained in detail in Section 2.1. In short, a device called *reader* is building up a magnetic field, which has two purposes. First, it powers passive devices that are in close proximity to the reader. Such a passive device is called *card*. Secondly, the field also acts as the communication channel through which both reader and card can exchange information. Note that also two active devices can communicate with each other, whereas each active device builds up its own field for sending data.

An important parameter of every communication system is the maximum data rate. The very nature of a communication system puts constraints on the data rate,

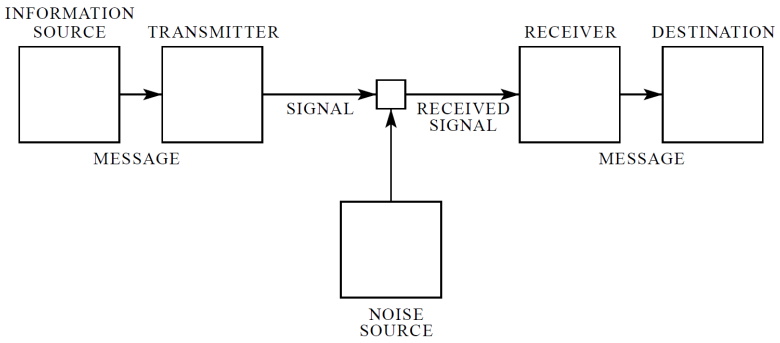



Figure 1.1: Schematic diagram of a general communication system [1].

in that it allows certain rates to pass through the channel more easily than others. For NFC, the ISO14443 standard states that the maximum data rate from card to reader is 6.78Mbit/s [7]. However, already signals carrying data with rates $\geq$ 848kbit/s get heavily degraded. Degradation means that while the signal is passing through the communication channel, it's shape is altered, and the information it carries is more difficult to reconstruct by the receiver. A component trying to reverse the effects of the communication channel is called *equalizer*.

In previous work [5], it was suggested that a certain kind of equalization algorithm, called well-behaved Normalized Constant Modulus Algorithm (wNCMA), shows a significant decrease in the Bit Error Rate (BER) for NFC communication channels. The number of bit errors is the number of received bits of a data stream over a communication channel that has been altered due to degradation. The goal of this thesis is to investigate how wNCMA could be implemented in an NFC receiver and determine it's usefulness for real-world applications.

## 1.2 Channel Equalization

Figure 1.2 illustrates the purpose of an equalizer as part of a communication system. The basic idea is that some data $a[n]$ is transmitted from one point or device to another. Thereby, the data needs to travel through a communication channel, which is modeled by a system with impulse response $h[n]$, as well as additive disturbances, such as noise or interference. The output of the channel is the received signal $x[n]$. The equalizer tries to reverse disturbances by filtering $x[n]$. Finally, a Decision Device (DD) is converting the equalized signal $y[n]$ to a symbol stream $\hat{a}[n]$. Ideally, the equalizer adjusts its filter coefficients such that

$$\hat{a}[n] = a[n - \Delta], \tag{1.1}$$

where $\Delta$ is a variable delay. Note that the channel exhibits deterministic and random disturbances. Typically, the deterministic disturbance is ISI, caused by the channel being dispersive (temporal spreading of the signal, resulting in an overlap of the individual symbol pulses).

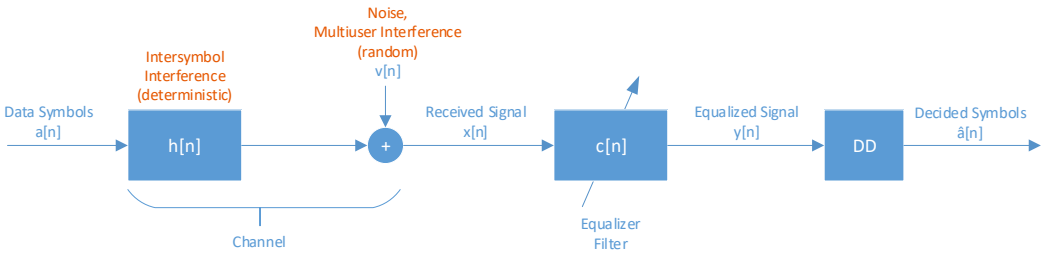



Figure 1.2: Block diagram of a baseband communication system, employing an equalizer to mitigate deterministic and random impairments, caused by the transmission channel.

Based on Equation 1.1, different objectives for equalizer design can be defined.

Minimum Bit Error Rate (MBER): Measure the output of the DD in comparison to a stored and standardized training sequence.

Minimum Mean-Square Error (MMSE): Measure the output of the equalizer in comparison to a training sequence or the DD output. In general, the error has the form

$$e[n] = (y[n] - a[n - \Delta])^2 , \tag{1.2}$$

where $a[n - \Delta]$ is either a training sample or the DD output.

Zero Forcing (ZF): All random impairments are neglected. Instead, the deterministic part of the channel is inverted such that

$$H(z)\, C(z) = z^{-\Delta}. \tag{1.3}$$

An equalizer that works without any training sequence is also called a blind equalizer.

> In the case of a highly nonstationary communications environment (e.g. digital mobile communications), it is impractical to consider the use of a training sequence. In such a situation, the adaptive filter has to equalize the communication channel in a self-organized (unsupervised) manner, and the resulting operation is referred to as *blind equalization*. Clearly, the design of a blind equalizer is a more challenging task than a conventional adaptive equalizer, because it has to make up for the absence of a training sequence by some practical means. Whereas a conventional adaptive equalizer relies on second-order statistics of the input data, a blind equalizer relies on additional information about the environment [10].

The wNCMA itself is classified as a blind deconvolution algorithm. In general, blind deconvolution is used to reverse the effects of convolution performed by a linear time-invariant system, operating on an input signal, without explicit knowledge of the impulse response function used in the convolution. Therefore, such algorithms are "designed such that they do not require the external supply of a desired response to generate the error signal in the output of the adaptive equalization filter." [2]. In other words, no DD is needed to form an error signal, resulting into a general form of a blind deconvolution algorithm, as shown in Figure 1.3, where the error is calculated by $e[n] = \mathrm{T}(y[n]) - y[n]$. Refer to Section 3.5 for finding out what the function T is, in case of the wNCMA.

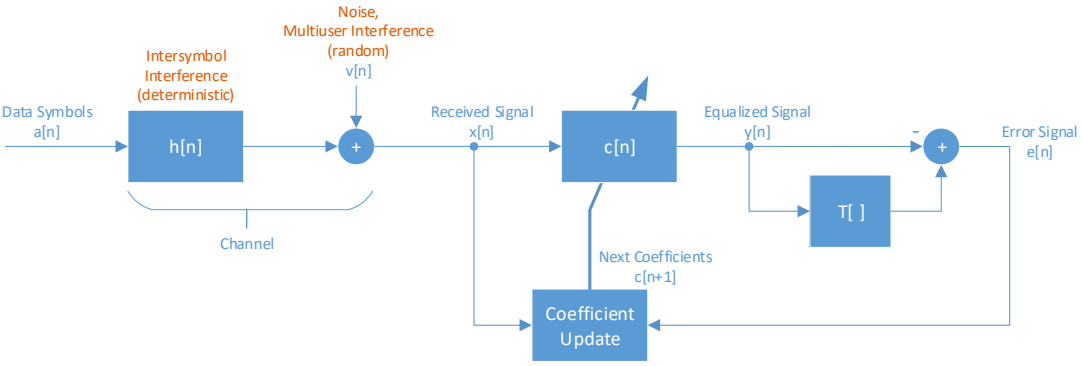



Figure 1.3: Block diagram of a baseband communication system, employing a general blind deconvolution algorithm for equalization purposes [2].

## 1.3 Intersymbol Interference

A channel with frequency response $C(f)$, defined as

$$C(f) = |C(f)|e^{j\theta(f)}, \tag{1.4}$$

is non-distorting if the magnitude response $|C(f)|$ is constant and $\theta(f)$ is a linear function of frequency. On the other hand, if $|C(f)|$ is not constant, the channel distorts the signal in amplitude, while if $\theta(f)$ is not a linear function of frequency, the channel distorts the signal in phase [3]. A channel with a frequency-dependent response is also called a dispersive channel.

> When transmissions are sent across a dispersive channel, it is possible for the output of that channel intercepted by the receiver to be distorted via the temporal spreading and resulting overlap of the individual symbol pulses. One consequence of such temporal spreading is the resulting inability of the receiver to accurately distinguish between different

received pulse shapes. We refer to this phenomenon as intersymbol interference [11].

To get a better understanding of the effect of ISI, consider a transmission channel with impulse response $\mathbf{h} = [1\ \frac{2}{3}\ \frac{1}{3}]$. Furthermore, assume that the sample rate equals the symbol rate. The following calculation determines if a transmission channel with impulse response $\mathbf{h}$ causes ISI, by checking if the result is $\neq 0$.

$$\text{ISI} = \frac{\sum_n |\mathbf{h}[n]| - \max|\mathbf{h}[n]|}{\max|\mathbf{h}[n]|} = \frac{\frac{2}{3} + \frac{1}{3} + 1 - 1}{1} = 1. \tag{1.5}$$

Equation 1.5 demonstrates that a system with impulse response $\mathbf{h}$ causes ISI, and that ISI can only be completely eliminated if $\sum_n |\mathbf{h}[n]| = \max|\mathbf{h}[n]|$. However, this goal is unnecessarily ambitious. In practice, ISI only needs to be reasonably bounded for a receiver to be able to decode the data correctly. A visual tool for determining the severity of ISI is the eye diagram. Hereby, similar to an oscilloscope display, a signal is applied to the vertical input, with a horizontal sweep rate of $\frac{1}{M \times T}$, $M$ being an arbitrary integer number, and $T$ the signal period [3]. As sketched in Figure 1.4, a signal with only low ISI will show a shape similar to an open eye. Similarly, a signal with severe ISI will show the shape of a closed eye. Several quantities, such as the optimum sampling time, peak distortion, and noise margin, can be deduced from the eye diagram.

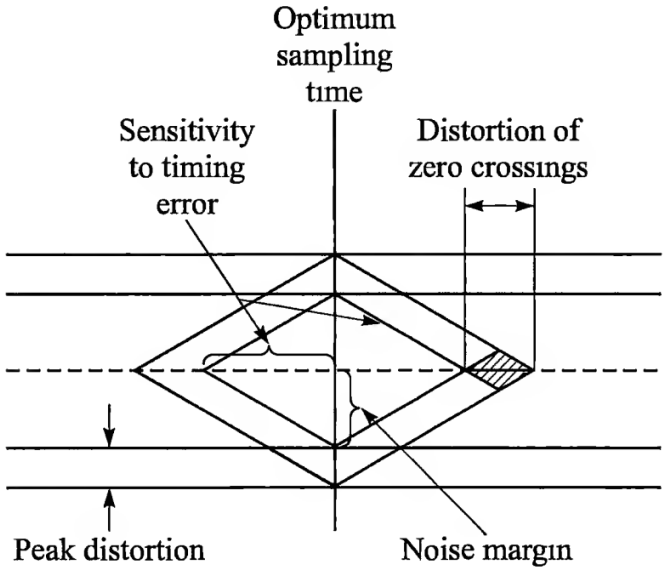



Figure 1.4: Effect of ISI on the eye opening [3].

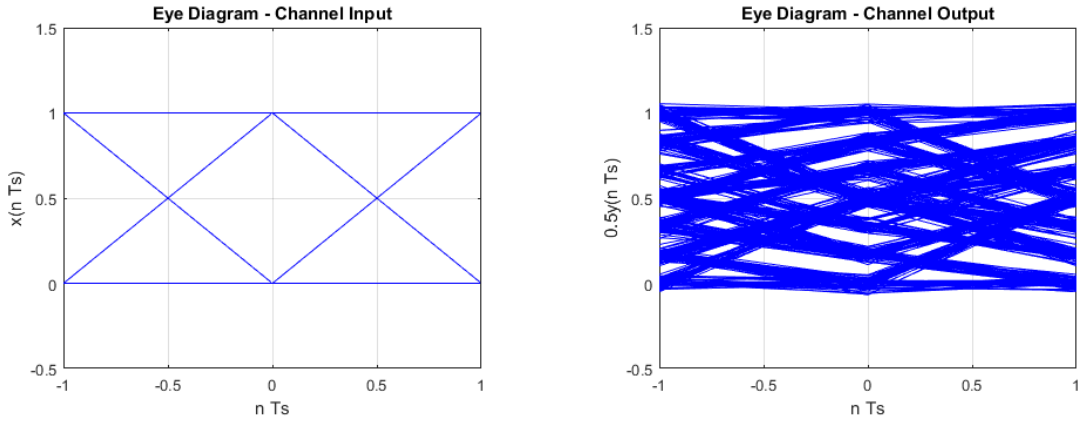
The state of the channel eye (open or closed) can not only be observed visually from the eye diagram, but also calculated. For the eye to be open, the following

condition needs to hold.

$$\begin{aligned} \sum_n |\mathbf{h}[n]| - \max|\mathbf{h}[n]| &< \max|\mathbf{h}[n]| \\ \frac{2}{3} + \frac{1}{3} + 1 - 1 &< 1 \\ 1 &< 1 \ldots \lightning \end{aligned} \tag{1.6}$$

As shown in Equation 1.6, the condition for an open channel eye is not fulfilled, since the result is contradicting. Thus, the eye for a transmission channel with impulse response $\mathbf{h} = [1\ \frac{2}{3}\ \frac{1}{3}]$ is closed. Figure 1.5 shows the eye diagram for the input, as well as the output signal of a system with impulse response $\mathbf{h}$.

Typically, a transmission channel causes ISI due to being band-limited. In [3], Proakis and others discuss how to shape a pulse such that during transmission, the channel introduces less ISI. In a way, such an approach can be seen as pre-equalization at the transmitter, as opposed to post-equalization after the channel at the receiver side. Pulse shaping is not further considered, as it goes beyond the scope of this thesis. Nevertheless, for the sake of completeness, the reader shall be informed about the existence of this approach, as it resembles the counterpart to equalization at the receiver side. Refer to [3] for further information regarding pulse shaping.




(a) Eye diagram of an undistorted binary signal.

(b) Eye diagram of a binary signal, distorted by a transmission channel.

Figure 1.5: Eye diagrams of a binary signal before and after a transmission channel with impulse response $\mathbf{h} = [1\ \frac{2}{3}\ \frac{1}{3}]$. The output signal is also affected by white noise.

## 1.4 Structure

Chapter 2 explores the communication technology NFC, while focusing on areas important for this thesis. Those areas are Load Modulation, as well as the frequency-

and time response of a typical NFC communication channel. Then, research projects regarding channel equalization and the implementation of different equalizers are presented.

Chapter 3 discusses the equalizer design. To begin with, a block-level description of the equalizer is presented, showing all main components and their interconnections. After that, the wNCMA equations are stated and explained. Furthermore, the explanations are supported by visuals, showing the equalization process in the complex domain, as well as that the equalizer can, when given enough time, reconstruct the transmitted signal with great accuracy. As it turns out, time is the real issue here, since the equalizer needs to converge during the preamble of a data frame.

Chapter 4 discusses the biggest topic of this thesis, namely the equalizer implementation, starting with an analysis of which and how many arithmetic components are necessary to implement the equalizer. This analysis is important since a complex algorithm needs to be broken down into simple components to make a successful design possible. Subsequently, a method for approximating multiplications and divisions based on [8] is presented. As it turns out in Section 5.3, such simplifications do not pay off in area in the particular case of the wNCMA equalizer.

The next topic is about numerical aspects of the equalizer. In other words, when the wNCMA is converted from a floating-point to a fixed-point format, how many integer and fractional bits should be assigned to each computational result? Also, what are additional effects that appear when moving to fixed-point format? Finally, after a short introduction into High-Level Synthesis (HLS), it is discussed how the correct functionality of the Register Transfer Level (RTL) code was verified.

Chapter 5 presents important results of the floating-point and fixed-point equalizer, focusing on parameter values and bitwidths Also, synthesis results are discussed. Last but not least, Chapter 6 concludes the thesis and gives an outlook on future improvements and applications.

# Chapter 2

# State of the Art

## 2.1 Theory of NFC Systems

### 2.1.1 Operating Modes and Load Modulation

NFC is a wireless data transfer technology. Powering a high-frequency current through an antenna induces an alternating magnetic field, which spreads around the antenna loop. If a substantial part of the magnetic field moves through the antenna of another NFC device, a detectable voltage is induced in its antenna loop. Now, the NFC initiator (the device which built up the field) can transfer data to the NFC target (the device receiving the field) by modulating the amplitude of the emitted magnetic field. As soon as the NFC target wants to send data back to the NFC initiator, they can switch roles. It is said that those two NFC devices communicate in Active Mode [4, 57].

However, there are also two other modes, which make NFC especially interesting in conjunction with Radio Frequency Identification (RFID). Those two modes are called Passive Mode Reader Emulation and Passive Mode Card Emulation. As the naming suggests, an NFC device can emulate a reader, such that it communicates with a card, and it can emulate a card, such that it communicates with a reader, see Figure 2.1.

Data transfer in Passive Mode works differently than in Active Mode since the card does not produce a field for sending data. In fact, the card is supplied by the reader field for the whole duration of the communication. An important point to realize about NFC and RFID is the fact that reader and card form a coupled system, as long as the card is located in the near-field of the reader [4, 43]. From a system perspective, studying a single NFC or RFID device alone is meaningless. Rather, one needs to always consider a complete system of reader and card.

The fact that reader and card form a coupled system opens up a new possibility for data transfer, namely Passive Load Modulation (PLM). A resonant card in the near-field of the reader draws energy from the magnetic field, causing a transformed impedance on the reader side [4, 43]. In PLM, a load resistance at the card is

switched on and off, which causes a change in impedance and thus a change in voltage at the reader's antenna. If the timing with which the load resistance is switched on and off is controlled by data, this data can be transferred to the reader.

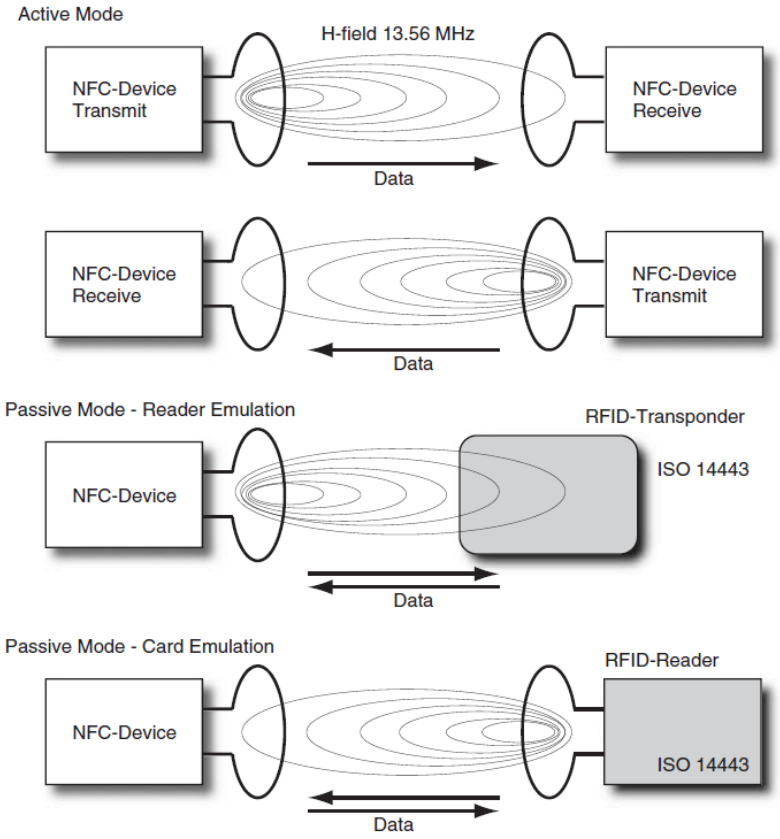



Figure 2.1: NFC distinguishes between three different modes: Active Mode, Passive Reader Emulation Mode and Passive Card Emulation Mode [4, 58].

### 2.1.2 Frequency- and Time Response of an NFC Communication Channel

Figure 2.2 shows a typical NFC circuit, which is also called coupling system, for data transfer via PLM. The reader in Figure 2.2 consists of four main parts. From left to right, these are the voltage source (which generates the field, also called carrier, at $f_c = 13.56\,\text{MHz}$), voltage divider, matching circuit, and resonant circuit. The card in Figure 2.2 is represented by a resonant circuit, in parallel with two resistances $R_{IC}$ and $R_{mod}$. $R_{IC}$ represents the constant resistive load of the card, while $R_{mod}$ is used for modulation.

By switching $R_{mod}$ on and off, the reader can detect a change in voltage $V_{out}$, since reader and card are inductively coupled. This mechanism of transmitting

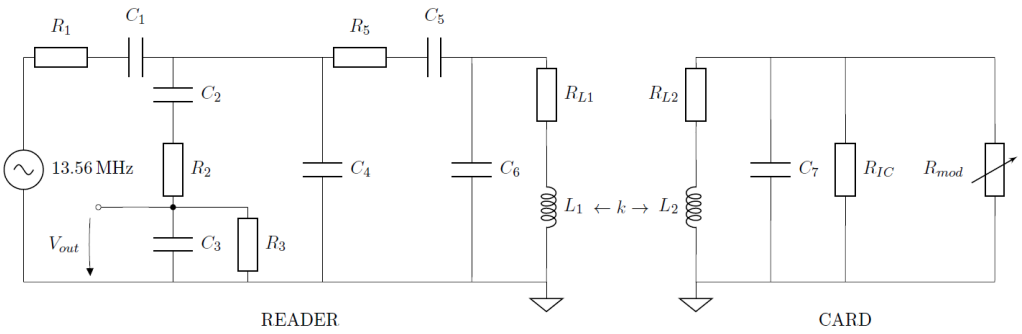



Figure 2.2: Typical NFC circuit schematic. It is also referred to as coupling system [5].

information is called load modulation. To improve detection of the change in voltage $V_{out}$ on the reader side, $R_{mod}$ modulates a subcarrier onto the field, instead of the data stream [Finkenzeller, page 43]. Subsequently, the subcarrier phase is varied over time to transmit information. More specifically, the digital modulation technique BPSK is used.

Another important parameter of the coupling system shown in Figure 2.2 is the coupling factor k. It is calculated by

$$\mathrm{k} = \frac{\mathrm{M}}{\sqrt{\mathrm{L_1 L_2}}}, \tag{2.1}$$

where M is the mutual inductance, $\mathrm{L_1}$ is the inductance of the reader loop and $\mathrm{L_2}$ is the inductance of the card loop. The value of k lies in the range of $0 \leq \mathrm{k} \leq 1$. If $\mathrm{k} = 0$, reader and card are fully decoupled due to great distance or magnetic shielding. If $\mathrm{k} = 1$, reader and card are fully coupled, meaning that both coils are subject to the same magnetic flux. Typical values for k range from 0.6 down to 0.01 or lower.

The coupling factor k is a dynamic factor of the coupling system since it depends on the distance and orientation of reader and card. Therefore, it deserves special attention. Figure 2.3 shows the frequency response of the Radio Frequency (RF) circuit from Figure 2.2 in baseband. The frequency response changes significantly with the coupling factor. For $k < 0.15$, the magnitude response is narrowband, causing ISI. As k increases further, the magnitude response begins to change shape and becomes broadband. Furthermore, the phase response starts to lose its odd symmetry, going hand in hand with a phase shift of the signal passing through. Recall that the Fourier Transform of a real signal $x(t)$ has Hermitian symmetry, meaning that $X(-f) = X^*(f)$, from which we can conclude that $|X(-f)| = |X(f)|$ and $\angle X(-f) = -\angle X(f)$ [3]. In other words, for a real signal $x(t)$, the magnitude of $X(f)$ is even, and the phase is odd. For a complex signal, this symmetry does not hold.

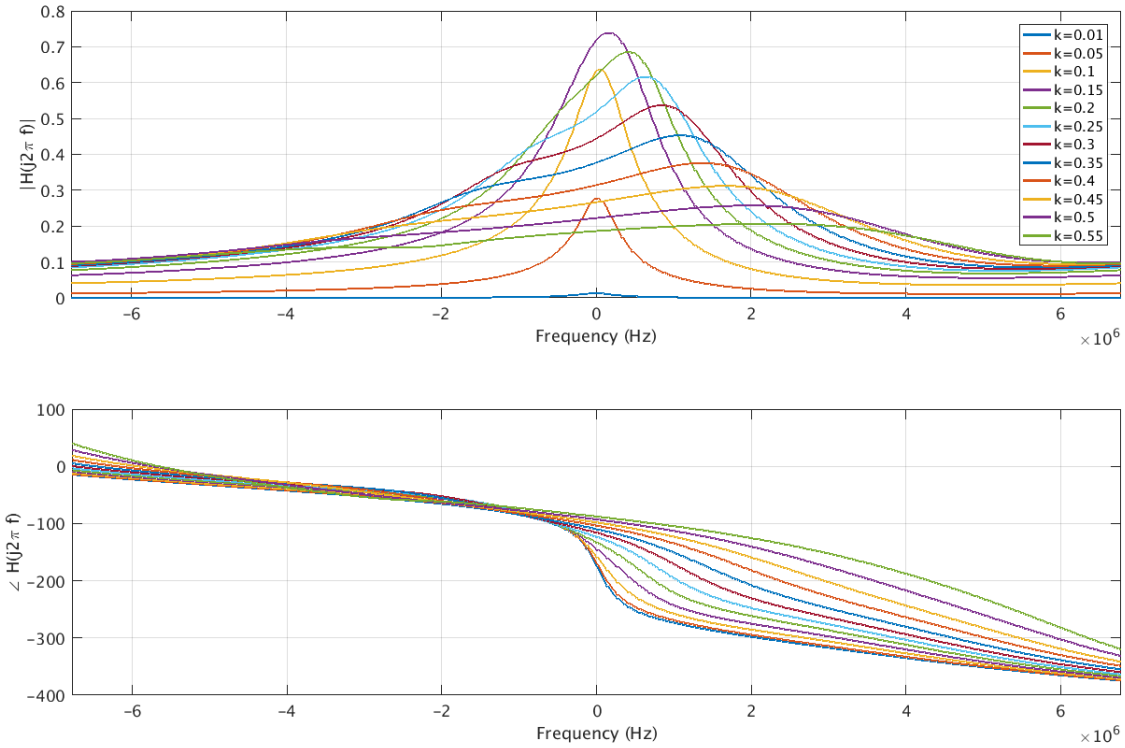



Figure 2.3: Baseband frequency response of the coupling system presented in Figure 2.2 for different coupling factors.

Figure 2.4 presents the coupling system effects in time domain. The input signal is a BPSK modulated, rectangular signal. All other shown signals are output signals of the coupling system for different coupling factors. The blue waveform represents the real part of the output signal, while the orange waveform represents the imaginary part.

For a coupling factor of k = 0.01, the output signal (which is the received signal at the reader side) has such a low amplitude that the receiver cannot detect it. The corresponding magnitude response for k = 0.01 in Figure 2.3 shows that nearly all frequencies are blocked. Increasing the coupling factor leads to an increase in signal strength (until k reaches a value of around 0.25). For low coupling factors, a strong distortion of the signal shape can be observed (see Figure 2.4 for k = 0.1). This distortion is caused by the narrowband characteristic of the coupling system for low coupling factors. More specifically, high-frequency components necessary for a rectangular-shaped signal are suppressed.

Because the coupling system causes a larger phase shift with k further increasing, the Q-Channel gains in signal strength compared to the I-Channel. For coupling factors above 0.5, the phase shift amounts to nearly 90°. Thus, the Q-Channel carries most of the information, and the I-Channel becomes negligible. The waveform strongly resembles the input signal due to the broadband frequency response of the

coupling system for high k.

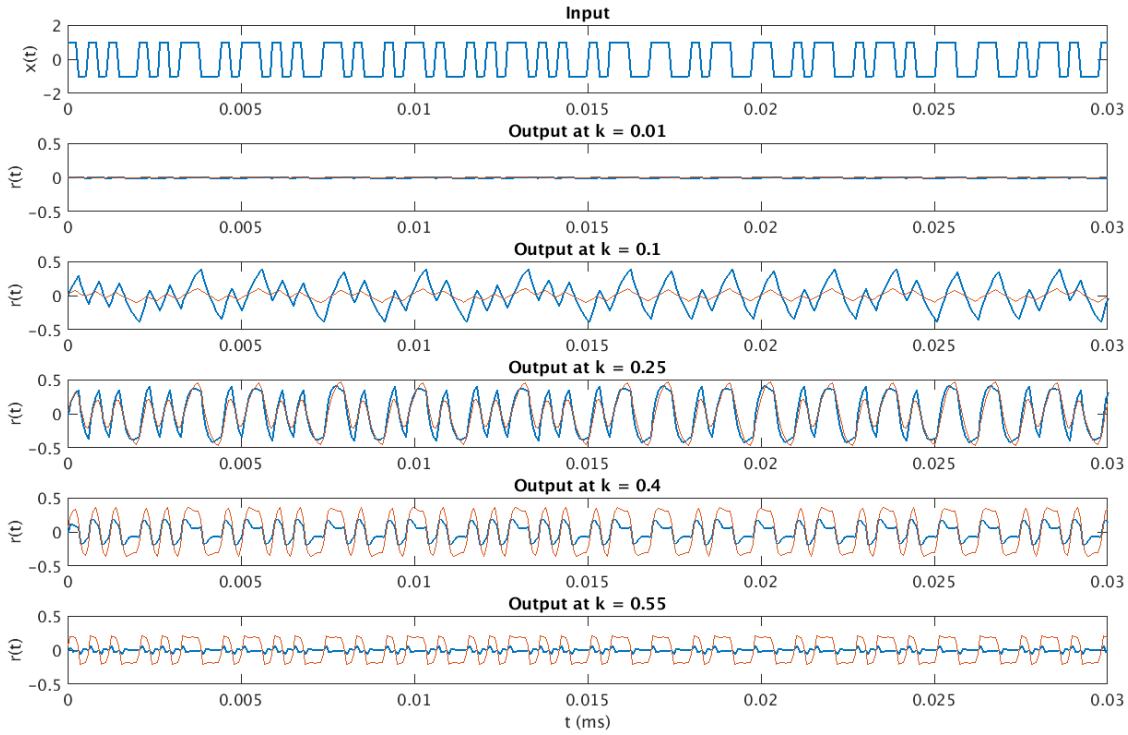



Figure 2.4: Coupling system effects in time domain for different coupling factors. The input signal is a BPSK modulated, rectangular signal. All other shown signals are output signals of the coupling system for different coupling factors. The blue waveform represents the real part of the output signal, while the orange waveform represents the imaginary part.

## 2.2 Research Projects

### 2.2.1 Equalization Approaches for NFC

The issue of encountering strong ISI when transmitting data via NFC at high data rates is widely known in research and industry. [12] tries to mitigate ISI by a processor, which extracts certain channel characteristics from the incoming signal at the receiver side to then choose one of multiple available equalizers. The reason for this rather complicated approach is the issue of convergence time. A decision for taking either one or the other equalizer can be taken much more quickly than letting an equalizer converge into a set of coefficients. [13] deals with ISI by pre-equalizing data at the transmitter side, where a test sequence is sent and received. Based on the sent and received test sequence, an inverse estimation can be performed.

[14] suggests using an LS approach, which the authors argue is better suited for PLM. The reason is that data transmission via load modulation requires to switch between two different coupling systems. In one system, the modulation resistance is switched on, and in the other system, the modulation resistance is switched off. This switching of the modulation resistance can either be interpreted as linear and time variant, or as non-linear and time-invariant. Due to this reason, the authors argue that ZF and MMSE equalization is not a reasonable approach for NFC and RFID. The suggested LS approach is shown in Figure 2.5. However, also an LS approach requires a training sequence, and therefore does not fall into the category of blind equalization.

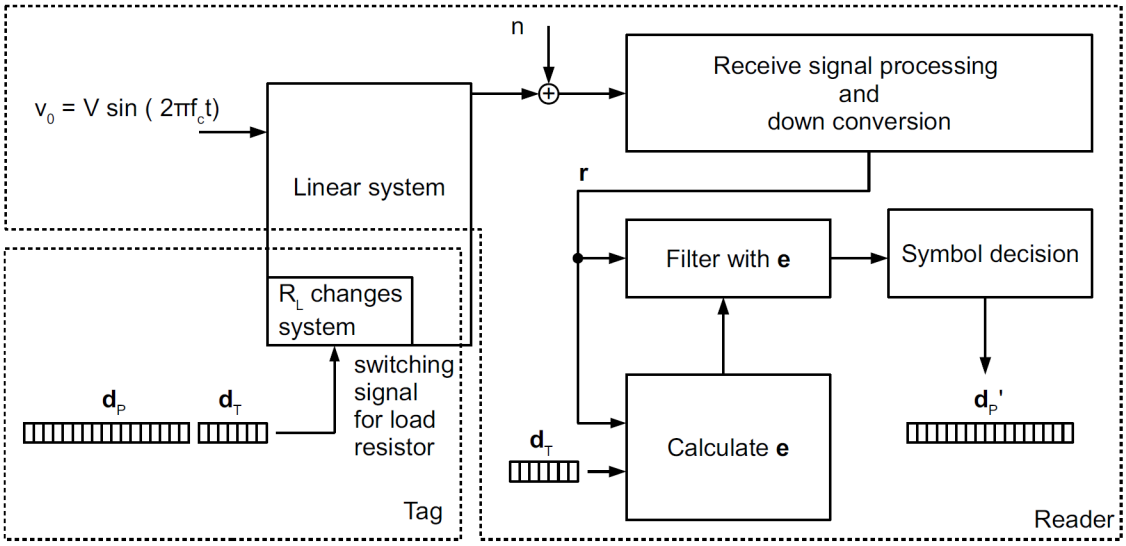



Figure 2.5: Schematical description of an equalization approach based on LS. The tag transmits a training sequence $\mathbf{d}_T$, as well as a payload $\mathbf{d}_P$. The knowledge of $\mathbf{d}_T$ at the reader side can be used to determine a set of filter coefficients, which mitigate ISI for the subsequent reception of the payload $\mathbf{d}_P$.

### 2.2.2 Implementation of Equalization Algorithms

Research regarding wNCMA has been conducted in [5], however, it has never been implemented. Nevertheless, implementations for other equalizers with a similar structure have been developed. [15] deals with implementing the Euclidean Direction Search (EDS) algorithm. The authors reduced the circuit size considerably by quantizing the filter coefficients to values of power of two. This quantization simplifies the filter, since shifters can be used instead of multipliers. As demonstrated in Section 5.3, the filter consumes little area compared with the rest of the equalizer. Thus, the approach of replacing multipliers with shifters to save area is not applied in this thesis.

In [16], the authors propose an implementation of the Least Mean Square (LMS) algorithm. The approach here is that instead of trying to simplify the basic mathematical operations (like multiplications), the LMS algorithm itself is simplified such that a more efficient implementation can be achieved. Specifically, the authors targeted the update calculation of the LMS, which is

$$\mathbf{c}[n+1] = \mathbf{c}[n] - \mu\, e[n]\, \mathbf{x}[n], \tag{2.2}$$

where $\mathbf{c}$ is the coefficient vector, $\mu$ is the step-size, $e$ the error and $\mathbf{x}$ the input signal. Instead of multiplying $\mu$, $e$ and $\mathbf{x}$, the authors propose to only consider the sign of the error and the input-signal, leading to the modified update calculation

$$\mathbf{c}[n+1] = \mathbf{c}[n] - \mu\, \mathrm{sgn}(e[n])\, \mathrm{sgn}(\mathbf{x}[n]). \tag{2.3}$$

No multipliers are needed anymore for calculating the coefficient update. An additional advantage is the fact that for real numbers, taking the sign also performs an implicit normalization, since $\mathrm{sgn}(x) = \frac{x}{|x|}$. In the end, Equation 2.3 simplifies to

$$\mathbf{c}[n+1] = \mathbf{c}[n] \pm \mu, \tag{2.4}$$

which in terms of complexity is probably optimal. For complex numbers, however, applying the signum function does not show the same benefits as with real numbers. The signum of a given complex number $z$ is the point on the unit circle of the complex plane that is nearest to z. Then, for $z \neq 0$,

$$\mathrm{sgn(z)} = \exp(i \arg z). \tag{2.5}$$

It is also possible to only perform the sign-operation on either the error or the input signal. [17] demonstrates the implementation of a complex blind adaptive decision feedback equalizer on an Field Programmable Gate Array (FPGA). For the coefficient update, only the signum of the error was calculated, not the input. However, the signum of the error was calculated by

$$\mathrm{sgn}(e[n]) = \mathrm{sgn}(\mathrm{Re}\{e[n]\}) + j\, \mathrm{sgn}(\mathrm{Im}\{e[n]\}), \tag{2.6}$$

which does not correctly normalize the error, but anyhow avoids multipliers. Nevertheless, as discussed in Section 5.3, reducing the error multipliers to signum-operations does not massively pay off in area, since the input signal multipliers in the coefficient update are the dominant factor. Simplifying those multipliers could be a valuable improvement.

According to [18], it is actually more efficient to not use signum-operations, but rather quantize the input to factors of two, such that multipliers can be reduced to shifters. The authors state the following argument.

> The performance degradation suffered in using sign-bit-only multiplication is significant. For digital implementations, a correlation multiplier that multiplies its inputs quantized to the nearest power of two is almost as inexpensive to implement as the sign-bit-only multiplier and its performance is very close to that of a true multiplier [18].

In [19], effects of limited precision in adaptive systems are studied. The author derives an expression for the steady-state error of the LMS algorithm due to limited precision, and subsequently shows that this steady-state error increases with decreasing step-size. At first this result might seem unintuitive because in the infinite-precision case, the steady-state error decreases with step-size (since the equalizer is "fluctuating less" around the optimum). However, a low step-size in finite-precision prevents the equalizer from reaching the optimum in the first place. This effect is discussed in detail in Section 4.2.1.

# Chapter 3

# Design

> For me, creativity includes problem-solving. That's the broad definition of it.
>
> *Edwin Catmull*

## 3.1 System Design Flow

Creativity is defined as "the ability to produce original and unusual ideas, or to make something new or imaginative" [20]. In that sense, technical problem-solving is often a highly creative process. Nevertheless, engineers also follow already established work flows, to be efficient in their creative act. Figure 3.1 shows the work flow, or more specifically the system design flow, followed during the course of this thesis. It shows great resemblance to already suggested flows in literature such as in [9], however, it has also unique characteristics.

The first step is to gather and clearly define the requirements. This step requires special care, as a mistake might need a lot of effort to correct. The requirements for this thesis are given in Section 3.2. Based on the requirements, the Initial System Model is developed in floating-point format. This format allows to focus on the algorithmic correctness instead of implementation details. The model is described as initial, because the designer can use any kind of functionality he or she consideres useful to implement the system, without careful thinking of how to realize this functionality in hardware. For this thesis, the floating-point system was developed in MATLAB, as it provides a great number of mathematical functions and libraries, as well as powerful debug capabilities. Section 3.3 describes the design of the Initial System Model.

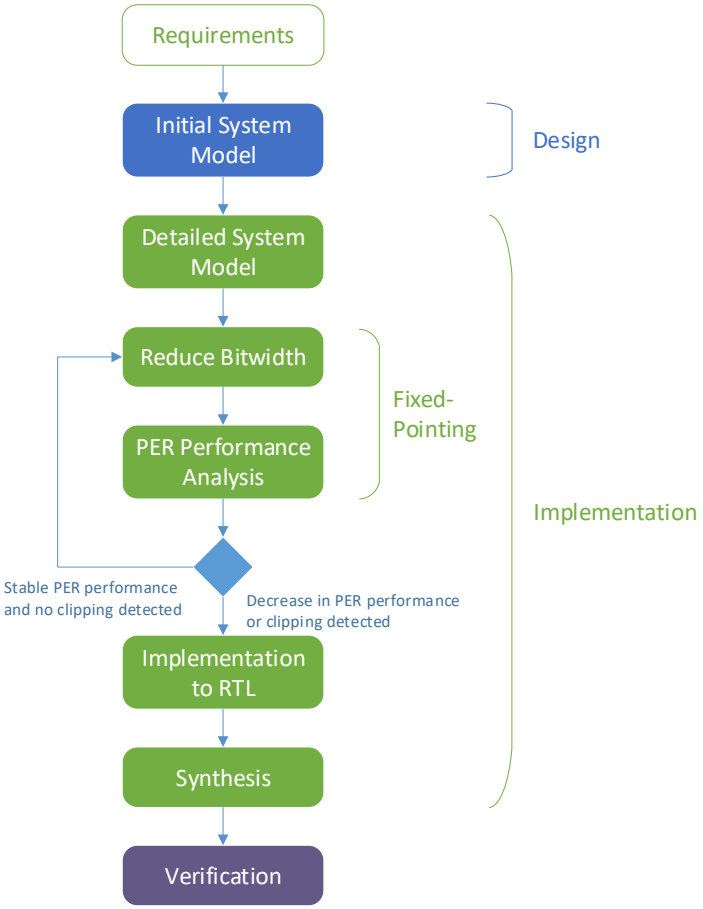



Figure 3.1: System design flow, starting from requirements, then designing the complete system, the conversion to fixed-point format, RTL implementation, synthesis and verification.

Once the Initial System Model fulfills all requirements, the designer needs to find ways of how to implement this system with a set of simple components available in hardware. As an example, one can easily use the logarithmic function in MATLAB, but how exactly is this logarithmic function implemented in hardware? As shown in Figure 3.1, this new system is called the Detailed System Model. Section 4.1 describes the design of the Detailed System Model.

The next step is to move from floating-point to fixed-point format. To start this conversion, the designer needs to assign a certain bitwidth to the input signals, and, if the signal is not a control signal, split this bitwidth between the integer and fractional part. Then, the bitgrowth throughout the system is analyzed, and whenever the designer sees the need for a reduction in bitwidth, quantizers and clippers can be inserted. Now, the bitwidth reduction of the quantizers and clippers can be tuned in an iterative fashion, while tracking the performance of the equalizer.

For further details regarding the conversion from floating-point to fixed-point format, refer to Section 4.2.

As soon as a fixed-pointed system model is available, the implementation to RTL starts. Here, the designer can also rely on High-Level Synthesis tools for accelerating the process. Finally, the RTL code is synthesized and verified. For reference, High-Level Synthesis tools are discussed in Section 4.3, while the process of verifying the equalizer is described in Section 4.4.

## 3.2 Requirements

The following list describes all requirements the equalizer needs to fulfill.

1. The equalizer must process complex-valued baseband signals. Therefore, also the filter coefficients are complex-valued.
2. The equalizer must run on a clock of $f_c = 13.56$MHz.
3. The equalizer output must have a bitwidth of 13bit, just like the input.
4. The coefficients must only be updated during a frame.
5. The equalizer must not exceed the area limit of $30k$ gates.
6. The equalizer must show an improvement of the PER for a data rate of 1.695Mbit/s.
7. The equalizer must show an improvement of the PER for Type-B frames.
8. The equalizer must be based on the wNCMA.
9. The equalizer must be converted from floating-point to fixed-point, such that a more efficient implementation on an Application Specific Integrated Circuit (ASIC) can be achieved.

## 3.3 Initial System Model

A block-level representation of the equalizer is shown in Figure 3.2. The equalizer consists of an FIR filter, the error calculation and the coefficient update. This filter-update-error loop is based on the wNCMA, which is explained in Section 3.5. One input of the filter is denoted with I/Q In, meaning that the input consists of two separate signals, the I-Channel and Q-Channel. The I-Channel represents the real part and the Q-Channel the imaginary part of the incoming signal. The other input of the filter is called Coefficients and determines the value of each filter coefficient. The filter output is connected to a Multiplexer (MUX), which forwards the filter output to the equalizer output if and only if the signal Out Mux Sel is 1. The

error calculation implements Equation 3.2, and the coefficient update implements Equation 3.3.

For the equalizer being able to process complex-valued baseband signals, the filter coefficients need to be complex-valued. This fact leads to an increase in complexity of the implementation, see Section 4.1. Processing baseband signals compared to passband signals has the advantage that the equalizer can run on a slower clock.

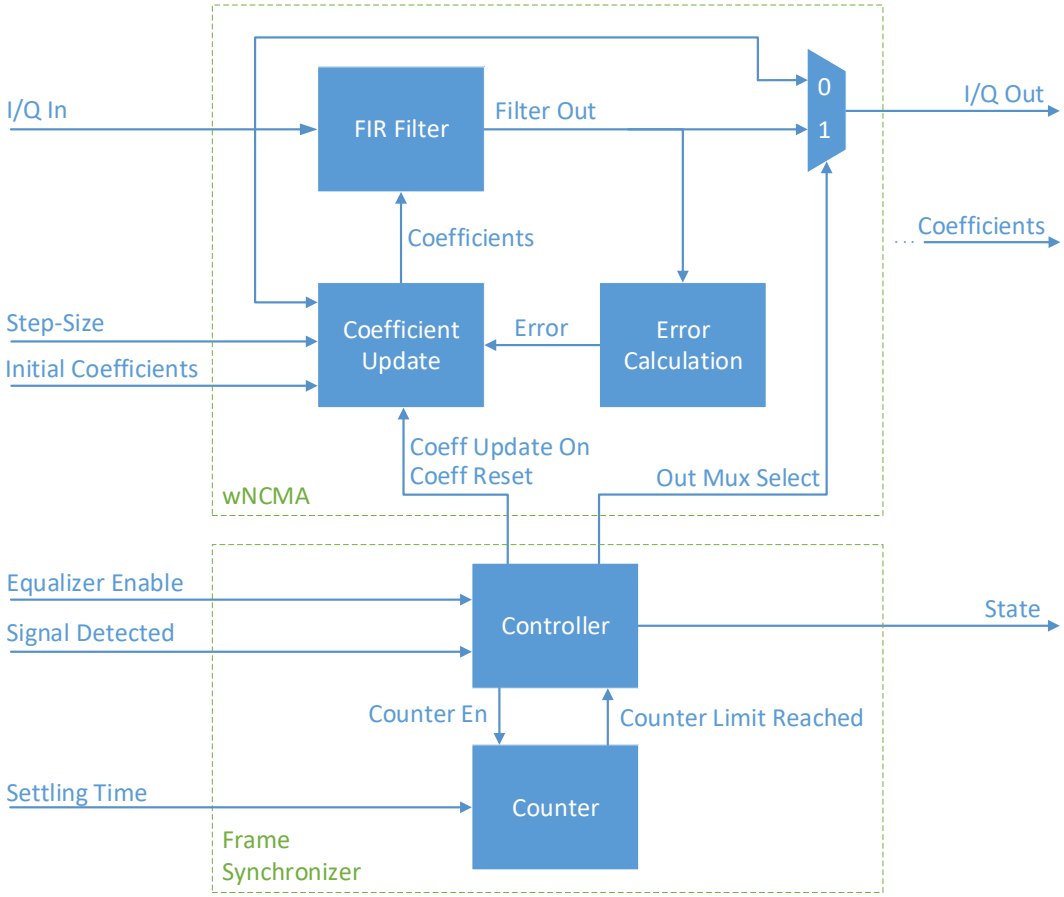



Figure 3.2: Block-level representation of the equalizer. The equalizer consists of an FIR filter, the error calculation and coefficient update. Additionally, a frame synchronizer is necessary, which consists of a controller and counter.

A second component, called the Frame Synchronizer (FS), consists of a controller and counter. The controller has two inputs, Equalizer Enable and Signal Detected. Depending on the internal FSM state of the controller, the controller activates the coefficient update (Coeff Update On), resets the coefficients to their initial values (Coeff Reset), or changes the select-signal of the MUX (Out Mux Sel). The initial values of the coefficients are given by the input Initial Coefficients. Also, the controller has one output called State, which informs about the internal state of the

controller FSM. The counter has only one input called Settling Time, which determines the initialization value of the counter. Section 3.6 explains the purpose and functionality of the FS.


Figure 3.3: UML Class Diagram of the wNCMA equalizer. Each class represents a Verilog module.

## 3.4 Use Cases

A UML class diagram of all Verilog modules and their input- and output ports is shown in Figure 3.3. Some of the signals comprise a postfix `_reg`. This postfix indicates that the signal is connected to a register, which can be accessed by firmware. The equalizer supports multiple use cases, which are depicted in Figure 3.4, as well as explained in the following list.

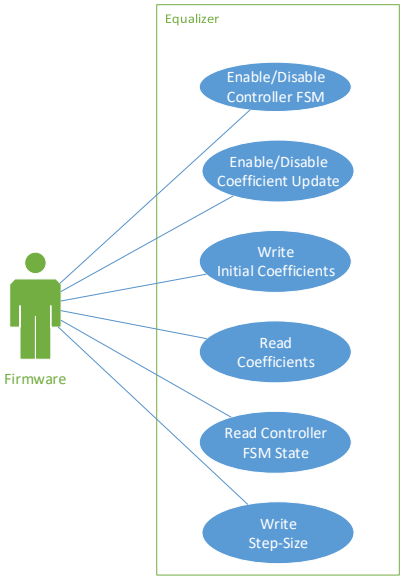



Figure 3.4: UML Use-Case diagram of the equalizer.

**Enable/Disable Controller FSM:** By setting the signal `eq_cont_en_reg` to 1, the controller of the FS is enabled. If disabled, the controller FSM always stays in state IDLE. In this state, the equalizer is turned off, meaning that no coefficient update is performed and the equalizer input is forwarded to its output (select-signal of Output-MUX is 0).

**Enable/Disable Coefficient Update:** By setting the signal `eq_coeff_update_en_reg` to 1, the equalizer updates the filter coefficients, if the state is either SETTLING_ON or ACTIVE. By setting `eq_coeff_update_en_reg` to 0, firmware can force static equalization. In other words, the equalizer acts as a static, complex-valued FIR filter (the coefficients being the initial coefficients).

**Write Initial Coefficients:** Firmware can specify the initial value of each coefficient.

**Write Step-Size:** Firmware can adjust the step-size of the wNCMA, see Equation 3.3.

**Read Coefficients:** Firmware can read out the current coefficients of the FIR filter.

**Read Controller State:** Firmware can read the current state of the controller FSM.

Figure 3.5 demonstrates one possible sequence of interactions between firmware and equalizer via registers. Before the reception of a frame, firmware would typically configure the desired step-size and initial coefficients first. After that, firmware enables the controller of the equalizer. As soon as the controller transitions from IDLE to SETTLING_ON and further to ACTIVE, the firmware knows (by reading the controller state register) that a frame is being received. After reception, the controller transitions to SETTLING_OFF. At this point, firmware can read and store the current filter coefficients of the equalizer. It makes sense to use those coefficients for subsequent frames as initial coefficients, since stationarity can be assumed. In other words, the transfer function typically does not change significantly between subsequent frames.

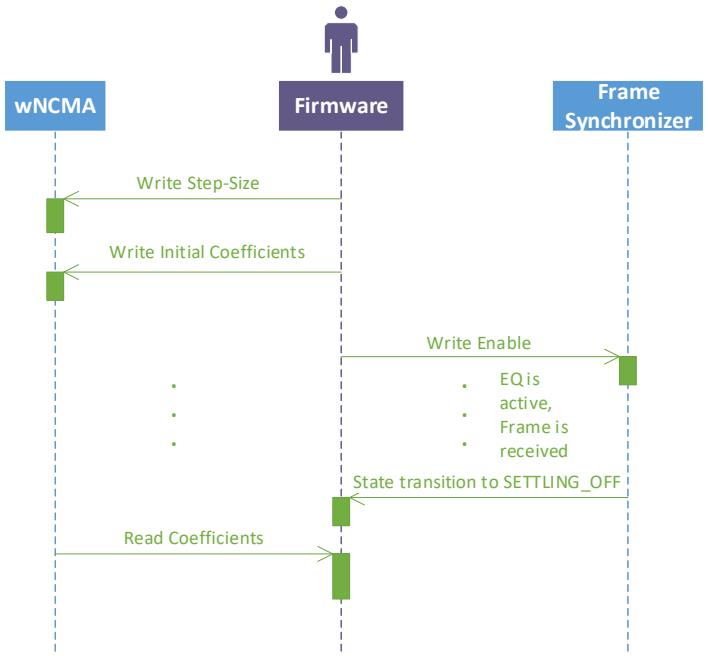



Figure 3.5: UML Sequence diagram, demonstrating a typical sequence of interactions between firmware and equalizer.

## 3.5 The wNCMA Algorithm

The equalizer implemented in the course of this thesis is based on the wNCMA devised in [5]. In [5], the author demonstrates that the wNCMA improves the BER of a signal corrupted by a typical NFC channel shown in Figure 2.2. However, what finally matters is if the equalizer can significantly reduce the PER in a real system. The number of packet errors is the number of faulty data frames received over a communication channel. In chapter 5, simulation results of the PER are shown.

The wNCMA consists of three steps: Filtering, error calculation and coefficient update. The filter output is calculated by

$$y[n] = \mathbf{c}^H[n]\mathbf{r}[n], \tag{3.1}$$

where $\mathbf{c} \in \mathbb{C}$ is the coefficient vector and $\mathbf{r} \in \mathbb{C}$ is the input vector. Equation 3.1 indicates that an FIR filter implementation is required, which can handle a complex-valued input signal and complex-valued coefficients. Next, the error is calculated by

$$\mathrm{e}[n] = \mathrm{Re}\{\mathrm{y}[n]\}^3 + j\,\mathrm{Im}\{\mathrm{y}[n]\}^3 - \mathrm{Re}\{\mathrm{y}[n]\}, \tag{3.2}$$

where $\mathrm{e} \in \mathbb{C}$ is the error and $\mathrm{y} \in \mathbb{C}$ the output of the FIR filter. Calculating the error for $y = \pm 1$ results to zero, showing that a BPSK-signal is the optimal case, which the equalizer tries to reach. Refer to Figure 3.7 for a complex-valued plot, which shows that the equalizer over time adjusts its coefficients such that the output reaches $\pm 1$ as close as possible. Finally, the coefficient update is determined by

$$\mathbf{c}[n+1] = \mathbf{c}[n] - \frac{\mu}{\alpha + ||\mathbf{r}[n]||^2}\,\mathbf{r}[n]\,\mathrm{e}^*[n], \tag{3.3}$$

where c is the coefficient vector, $\mu$ the step-size, $\alpha$ a constant to avoid division by zero, $\mathbf{r}$ the input vector and $\mathrm{e}^*$ the complex conjugate of the error. The division of the coefficient update term by the power of the input signal is called power normalization. It is useful for potentially increasing the convergence time by making the coefficient update independent of the input power. For more details regarding the wNCMA refer to [5].

Subsection 2.1.2 explores how a typical NFC coupling system behaves in frequency- and time domain. For a low coupling factor k, the coupling system is narrowband and causes strong ISI. In contrast, the coupling system is broadband for high coupling factors but also causes a larger phase shift. Furthermore, Figure 2.4 shows the effects of the coupling system in time domain. The question now is if the wNCMA algorithm can significantly improve the signal, such that it more closely resembles the input. Figure 3.6 demonstrates this improvement.

The left-hand side of Figure 3.6 shows the equalizer output from the first sample onwards for around two-hundred samples. Clearly, the output does not resemble the input. However, the equalizer did not have enough time to learn. In other words, the

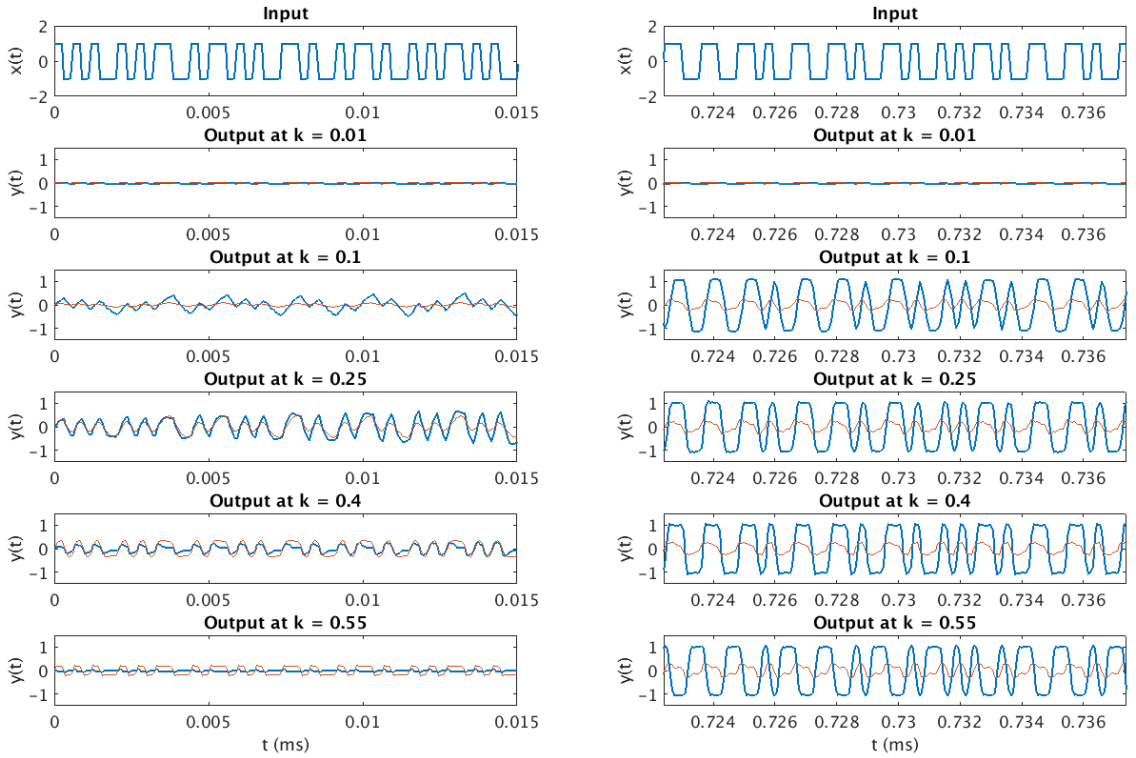



Figure 3.6: Equalization in time domain for different coupling factors. The input signal of the coupling system is a BPSK modulated, rectangular signal. All other shown signals are output signals of the equalizer for different coupling factors. The blue waveform represents the real part of the output signal, while the orange waveform represents the imaginary part. On the left-hand side, the very beginning of the frame is shown, whereas on the right-hand side, a later part of the frame is shown.

equalizer could not yet adjust its coefficient such that the error becomes minimal. The right-hand side of Figure 3.6 shows the equalizer output much later in time when the equalizer has already long settled. The similarity of the output compared to the input is clear, leading to a significantly easier retrieval of information. How fast an equalizer, and more generally any adaptive system, settles, is a major parameter of such a system and is formally known as the convergence behavior. Chapter 5 will demonstrate convergence behavior to be highly critical for NFC application.

Regarding Figure 3.6, another observation can be made, namely that for the highest coupling factor k = 0.55, the output approximates the inverse of the input. The equalizer might run into the inverse solution because both minimize the error. Thus, it does not make a difference to the equalizer if the output is inverted or not.

Lastly, refer to Figure 3.8 for a demonstration of multiple eye-diagrams, which show that the wNCMA can successfully reverse channel effects and create an open-eye pattern in the I-Channel. Clearly, the channel at low-coupling of k = 0.1 causes strong ISI, and the eye in both channels is closed. For high coupling of $k$k = 0.55,

the channel is broadband and hence does not cause ISI, but a phase shift of nearly 90°.

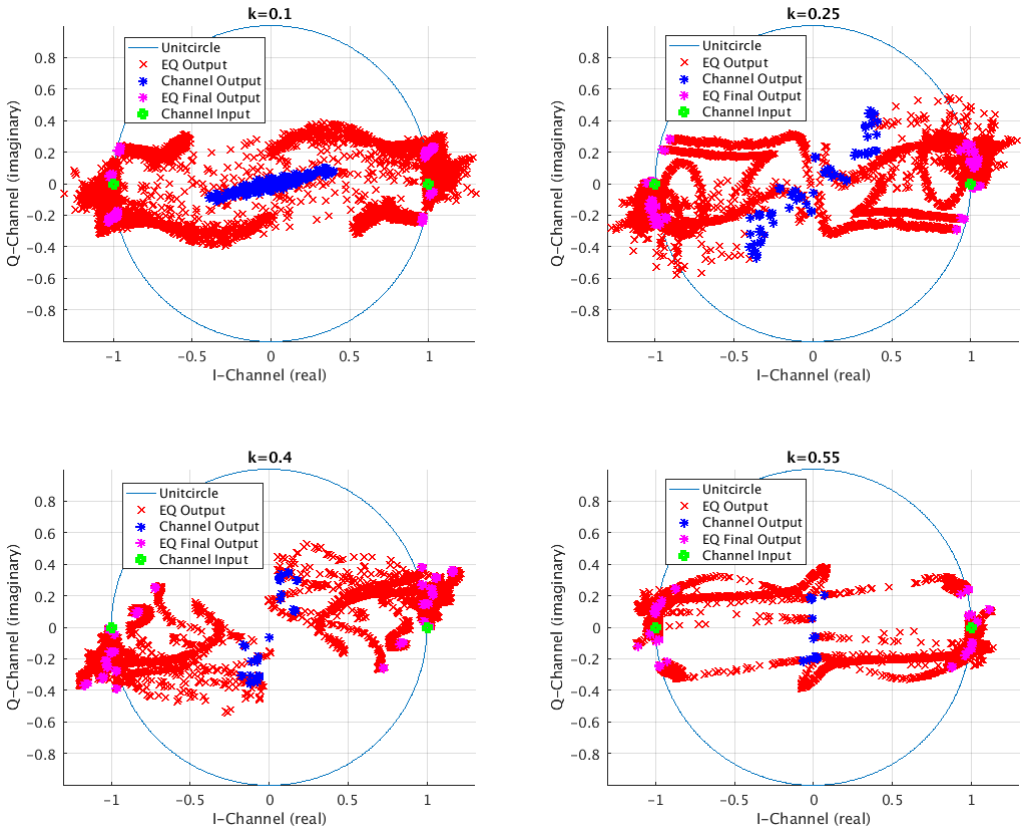
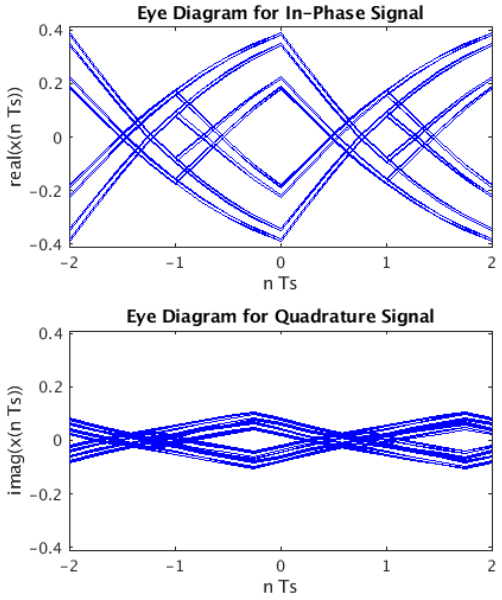
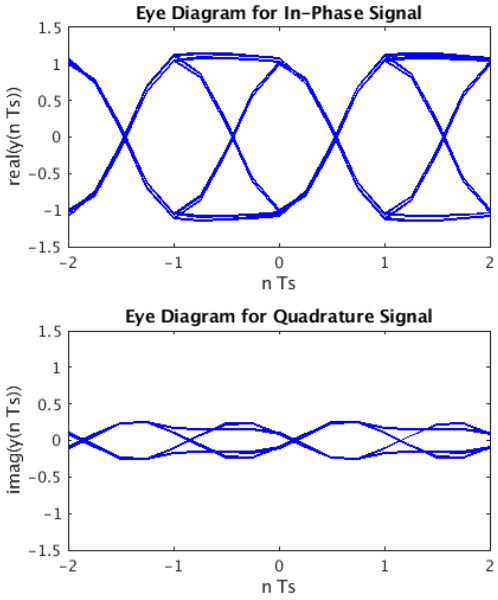


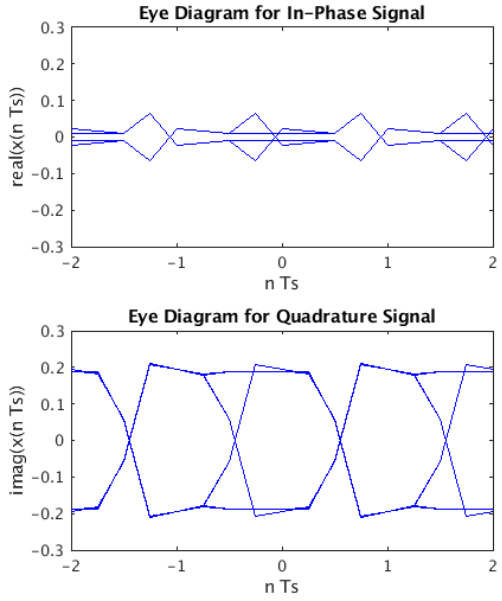

Figure 3.7: Complex-valued plot, which shows for different coupling factors that the equalizer over time adjusts its coefficients such that the output reaches ±1 as close as possible. The green circles represent a perfect BPSK input signal, the blue stars represent the channel output, the red crosses represent samples of the equalizer output and finally the pink stars represent the equalizer output after it settled. The coupling model shown in Figure 2.2 was used for this simulation. Additional parameters are: $f_{sampling} = 13.56\text{MHz}$, data rate = 1.695Mbit/s.



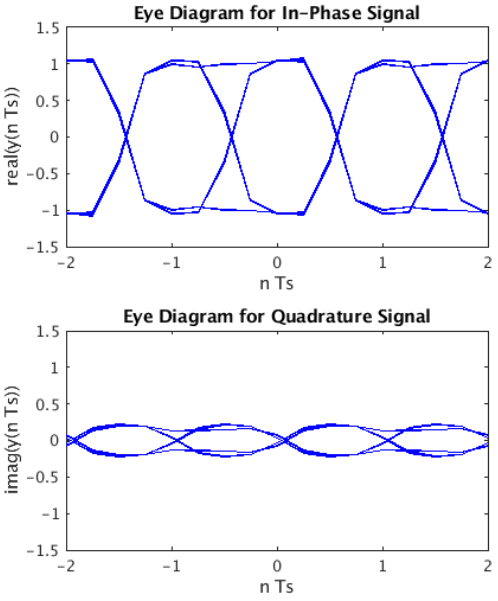

(a) Received signal at k = 0.1.




(b) Equalized signal at k = 0.1.




(c) Received signal at k = 0.55.




(d) Equalized signal at k = 0.55.

Figure 3.8: Eye diagrams at coupling k = 0.1 (above) and k = 0.55 (below), as well as before (left) and after (right) the equalizer. Note the different scaling of the signals before and after the equalizer.

### 3.5.1 Initial Coefficients

For any adaptive system, an initial state needs to be configured. Equation 3.3 shows that to determine new coefficients, old coefficients are required. Choosing initial coefficients has a significant impact on the equalization performance. In general, the closer the initial coefficients are to the optimal coefficients, the faster the equalizer converges [5]. However, to make a close-to-optimal choice for the initial coefficients, detailed knowledge about the channel is necessary.

Choosing close-to-optimal initial coefficients can be an issue when the equalizer needs to work for multiple different channels. Furthermore, the quality of the incoming signal is not extremely low. Rather, the signal quality is just not good enough to robustly receive incoming frames (at higher data rates), see Figure 3.9. This fact suggests that a set of initial coefficients that simply pass through the incoming signal to the output is a reasonable and sufficient starting point (which is also flexible across different channels). The coefficients

$$\mathbf{c}_{\text{init}} = [1; \text{zeros}(\text{N} - 1, 1)] \tag{3.4}$$

realize the mentioned effect of passing through the incoming signal, where N is the filter order.

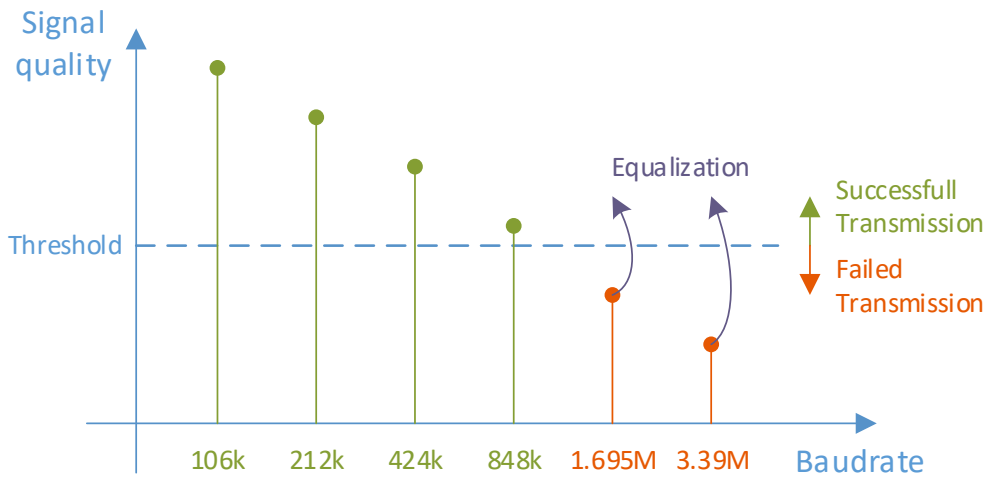



Figure 3.9: Moving to higher baudrates gradually reduces the quality of the incoming signal, which is disturbed by the channel. If the signal quality crosses a certain threshold, the receiver cannot completely restore the transmitted information anymore, leading to a failed transmission. The purpose of the equalizer is to raise the signal quality such that the receiver can completely restore the transmitted information.

### 3.5.2 Error and Coefficient Development over Time

It is insightful to study the error and coefficient development over time for different coupling factors. Figure 3.10 shows the error over time on the left-hand side, as well as the output signal of the equalizer over time. Seeing the output signal right next to the error is helpful since the error is calculated solely on the basis of the output signal, as shown in Equation 3.2.

For a low coupling factor such as $k = 0.01$, the card is so far away that the transmitted frame is not detectable. In that case, the equalizer output is zero and the error is zero as well. For a coupling factor of $k = 0.1$, a frame can already be clearly detected, but due to the strong narrowband channel at that coupling, ISI is strong and the error is large. However, the error is not large immediately. Since the output of the equalizer is low in amplitude at the beginning of the frame, the error is automatically low as well, although ISI is strong. Only when the equalizer starts to scale up the coefficients, the error actually increases. Over a significant timespan of a few milliseconds, the equalizer manages to minimize the error and the resulting output signal at the end of the frame can be seen in Figure 3.6. Similar effects can be observed for higher coupling factors, although the error does not become as high due to less ISI (the channel for higher coupling is more broadband).

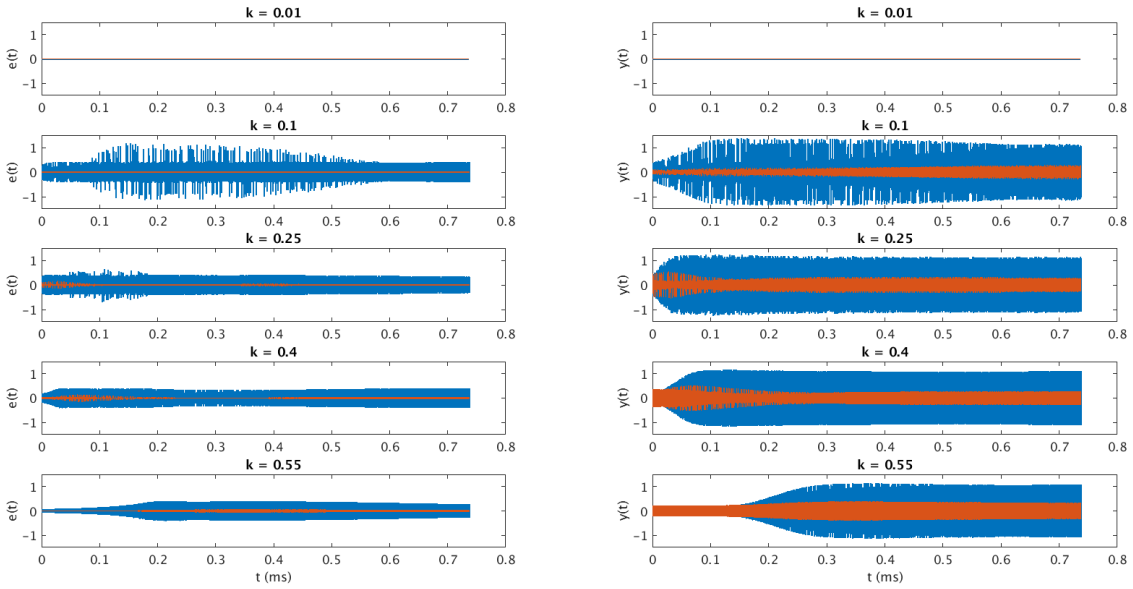



Figure 3.10: The wNCMA error over time for different coupling factors on the left-hand side, compared to the respective wNCMA output on the right-hand side.

Figure 3.11 shows the filter coefficients over time for different coupling factors. Since the error is zero throughout the whole frame for a coupling factor of k = 0.01, the coefficients are not updated. Therefore, the coefficients stay at their initial value. For k = 0.1, the real parts of all coefficients are heavily adjusted, to both scale up the input and decrease ISI. However, the imaginary parts stay basically zero, since

no rotation is required. As can be seen in Figure 3.7, the channel does not cause a significant phase shift at low coupling. Going further up in coupling, the phase shift becomes stronger and the equalizer adjusts also the imaginary parts of the coefficients. For a coupling of k = 0.55, the channel is broadband, but causes a phase shift of around 90 degrees (see Figure 3.7). Therefore, only the imaginary parts of the coefficients are adjusted to counteract the phase shift, but the real parts stay more or less at their initial value.

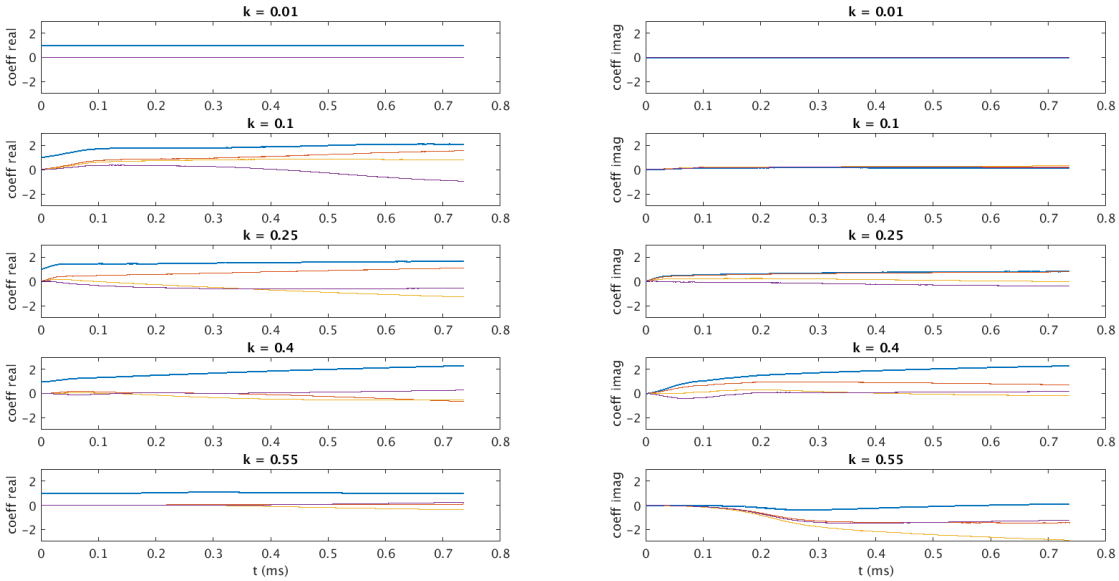



Figure 3.11: The filter coefficients over time for different coupling factors. The real part is shown on the left-hand side and the imaginary part on the right-hand side. The blue curve shows the first coefficient, orange the second, yellow the third and purple shows the fourth coefficient.

## 3.6 Frame Synchronizer

In practice, the equalizer is not supposed to update the filter coefficients the entire time. If no data is received, the equalizer would otherwise scale up noise at the input, which might lead to detect a non-existent data transmission by the subsequent components in the signal processing chain. Also, if after a period of silence (noise only) a data transmission starts, the equalizer might be unable to propagate into an optimal set of coefficients. For this reason, the coefficient update needs to be disabled until an incoming signal is detected. In the course of this thesis, it was decided that the equalizer can rely on an external component of the existing signal processing chain to detect an incoming signal.

However, it turns out there is an issue in that approach. Figure 3.12 shows that in the signal processing chain, the equalizer is part of a module called Preprocessor,

which itself is placed before another module called Receiver. Amongst other things, the receiver is providing the information of a signal being detected. At the moment a signal is detected, the coefficient update is turned on, with the coefficients being initialized to their initial values. It is possible that right after the coefficient update has been turned on, the output of the equalizer does not resemble what the external detector classifies as an active signal (since the initial coefficients can be arbitrarily configured by firmware). In this case, the coefficient update is turned off, since the external detector does not indicate an active signal anymore. However, not only is the coefficient update turned off, but also the coefficients themselves are reset to their initial values. During frame reception, this behavior continues in a loop, resulting in the frame itself being lost.

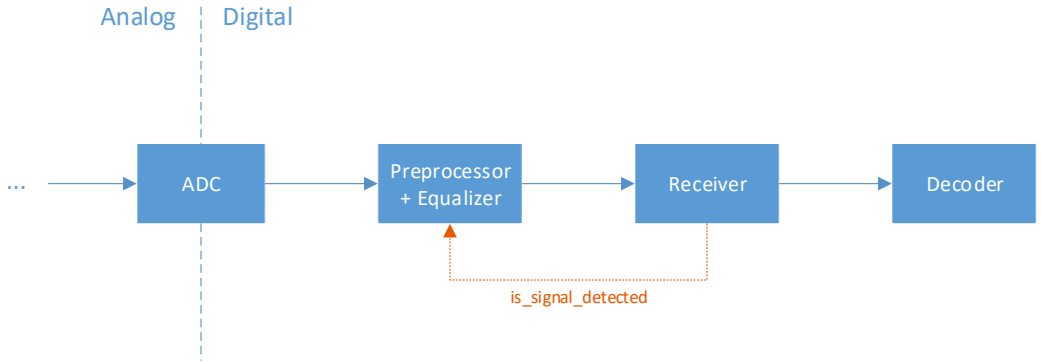



Figure 3.12: High-Level depiction of the receiving signal processing chain. The equalizer is part of the Preprocessor and is turned on by the Receiver, which is placed after the Preprocessor.

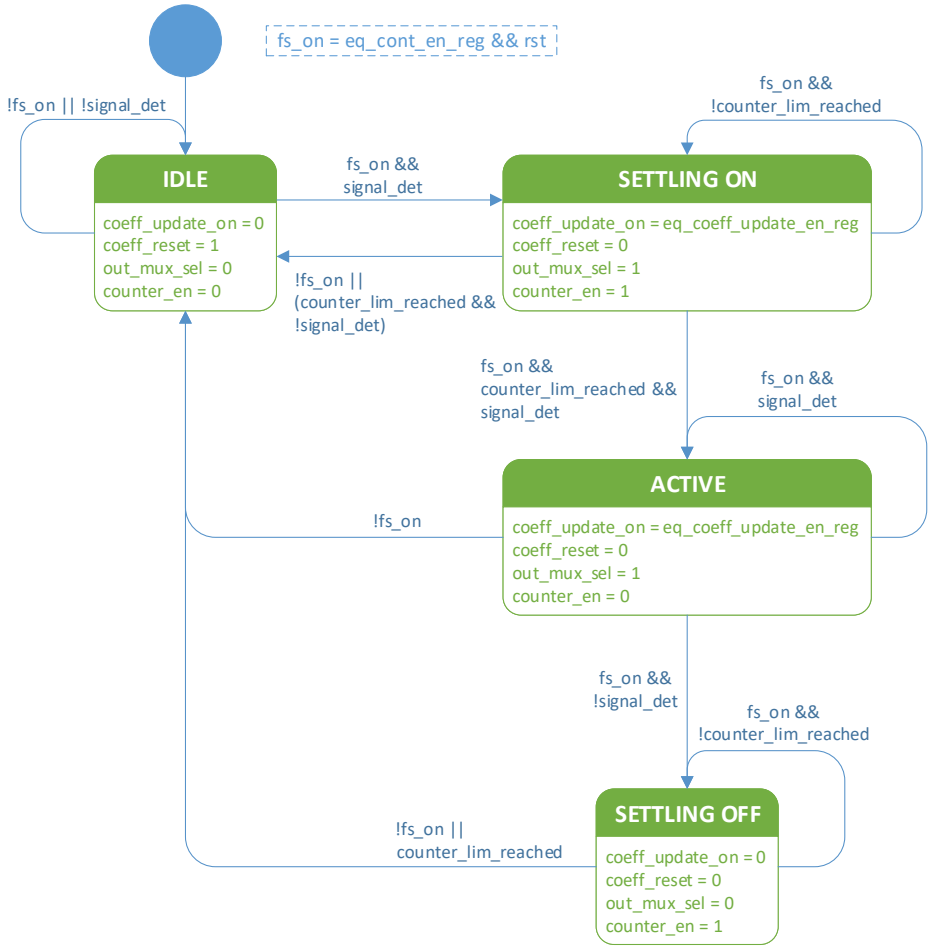



Figure 3.13: A UML State Machine Diagram of the controller FSM, which is a Moore machine, consisting of four states. The logical conditions in blue determine when transitions to other states happen, while the assignments in green show the value of each output wire for every state.

The controller in Figure 3.2 is enabling and disabling the coefficient update, as well as switching the output of the equalizer between pass-through and the filter output. The controller itself is an FSM, which is shown in Figure 3.13. The controller FSM has four states, which are IDLE, SETTLING ON, ACTIVE and SETTLING OFF. In IDLE, the output is directly connected to the input (pass-through) and the equalizer is keeping the filter coefficients at their initial value. As soon as the external detector in the receiver detects a signal, the controller changes its state to SETTLING ON. Hereby, a counter is decreasing its internal value by one in each cycle until the value reaches zero. If a signal is detected when the counter reaches zero, the controller changes its state to ACTIVE. Only in the ACTIVE state, the output of the equalizer is switched to the filter output. As soon as IS_signal_detected

goes back to zero (in other words, the frame is over), the controller changes its state to SETTLING OFF. Again, the counter (initialized to the same value as in SETTLING ON) is decrementing its value by one per cycle until the value reaches zero. Then, the controller ends up again in the IDLE state. The described FSM makes sure that the coefficient update is only enabled if a frame is being received. Furthermore, the equalizer in SETTLING ON has time to settle to an approximate optimum before the filter output is forwarded to the receiver. Also, see Figure 3.14 for a depiction of how the FSM states change during a frame.

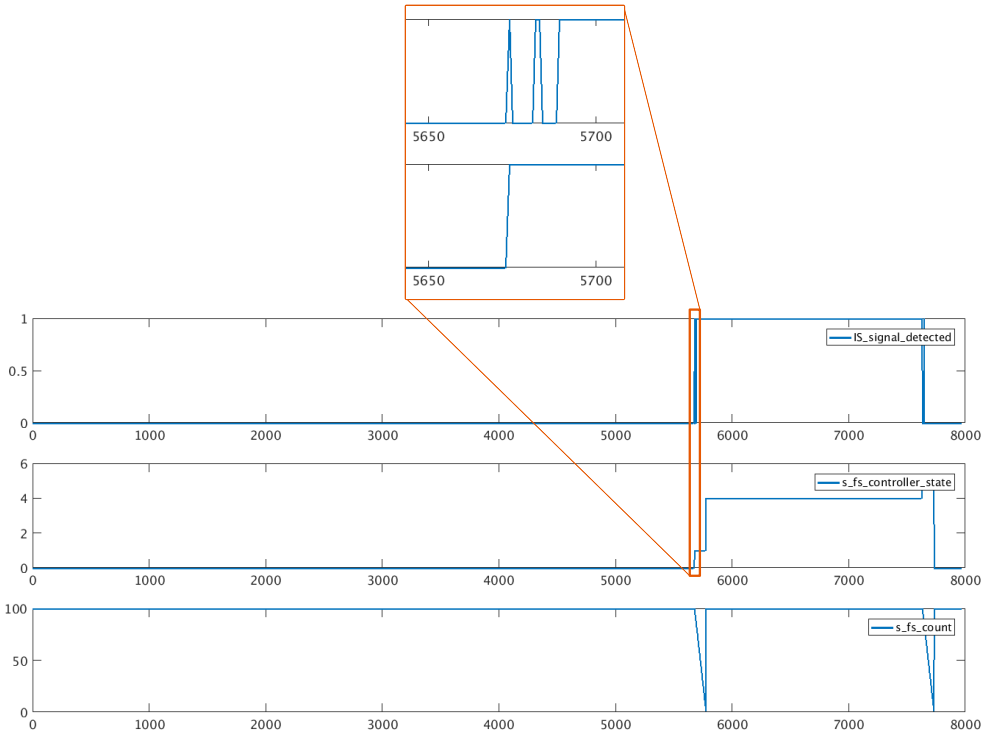



Figure 3.14: Depiction of controller signals during a frame. The frame starts at sample 5674 and ends at sample 7628. States: IDLE = 0, SETTLING ON = 1, ACTIVE = 4, SETTLING OFF = 5.

## 3.7 Issues Arising From the Type-B Frame Format

Section 2.1.2 demonstrates what effects a simplified NFC channel has on the transmitted BPSK signal. As it turns out in Section 3.5.1, the equalizer is capable of restoring the BPSK signal no matter the coupling (as long as the coupling is high enough that a signal can be detected). However, Section 3.5.2 demonstrates that the equalizer needs a significant time to converge. Therefore, it is of high importance

to understand how an actual data frame looks like and in what timespan and under which conditions the equalizer needs to converge (not necessarily to the optimum, but to a set of coefficients which allows for correct bit detection).

To begin with, Figure 3.15 shows the format of a Type-B frame, as given by the ISO14443-3:2011 standard [6]. There are also other protocols, such as Type-A. However, the equalizer is only active when a Type-B format is used, as given by project requirements. As shown in Figure 3.15, a Type-B frame consists of a Start of Frame (SOF), the characters (data), and an End of Frame (EOF).

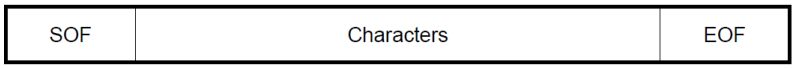



Figure 3.15: Type-B frame format [6].

Figure 3.16 shows the character format of a Type-B frame. It consists of a start bit, eight-bit of data (Least Significant Bit (LSB) first) and a stop bit. The start bit is equivalent to a logical 0, while the stop bit is equivalent to a logical 1. What represents a logical 0 and logical 1, as well as how the SOF looks like, is given by the Bit Coding and Representation section in [7]:

- After any command from the Proximity Coupling Device (PCD), a guard time TR0 shall apply in which the Proximity Card or Object (PICC) shall not generate a subcarrier. TR0 shall be greater than 1024/fc ( 75,5 $\mu s$).
- The PICC shall then generate a subcarrier with no phase transition for a synchronization time TR1. This establishes an initial subcarrier phase reference Ø0. TR1 shall be greater than 80/fs.
- This initial phase state Ø0 of the subcarrier shall be defined as logic "1" so that the first phase transition represents a change from logic "1" to logic "0".
- Subsequently, the logic level is defined according to the initial phase of the subcarrier.

  Ø0 Represents logic "1"

  Ø0 + 180° Represents logic "0"

Note that the standard uses the abbreviation PCD for referring to a reader, and PICC for referring to a card. Especially important for equalization is the SOF (the EOF is less important - the interested reader can refer to [7] and [6]). During the SOF, the equalizer needs to converge to coefficients which allow for correct bit detection, because right after the SOF, the data bits are transmitted.

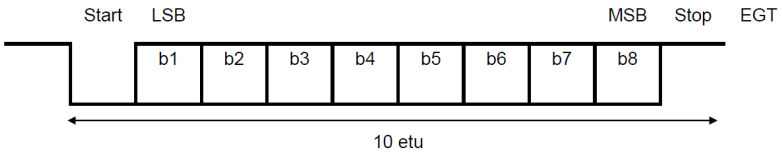



Figure 3.16: Type-B character format [6].

There are two problems arising from the presented frame format regarding equalization. First, the time during which the equalizer needs to converge is only $\frac{80}{f_s}$ long. For a subcarrier frequency $f_s = 1.695\,\text{MHz}$, we get $\frac{80}{1.695\,\text{MHz}} \approx 47.2\,\mu s$. From a different perspective, the equalizer has only $80 \times \frac{f_c}{f_s} = 80 \times \frac{13.56\,\text{MHz}}{1.695\,\text{MHz}} = 640$ samples to converge, since the clock frequency according to project requirements is $f_c = 13.56\,\text{MHz}$. Secondly, the convergence to a suitable set of coefficients during TR1 is impacted by the fact that during TR1, the transmitted subcarrier does not contain phase changes. Since bits are transmitted by phase changes of the subcarrier, the signal during TR1 and after TR1 looks very different.

For the sake of completeness, [7] defines the relationship between high bit rates and subcarrier frequency as shown in Figure 3.17. For bit rates below what is shown in Figure 3.17, the subcarrier frequency is always $\frac{f_c}{16} \approx 848\,\text{kHz}$.

| Bit rate | Subcarrier frequency |
|---|---|
| *fc*/8 (~1,70 Mbit/s) | *fc*/8 |
| *fc*/4 (~3,39 Mbit/s) | *fc*/4 |
| *fc/2* (~6,78 Mbit/s) | *fc*/2 |

Figure 3.17: Relationship between high bit rates and subcarrier frequency as defined in [7].

# Chapter 4

# Implementation

> Theory is the first term in the Taylor series expansion of practice.
>
> *Thomas M. Cover*

## 4.1 Detailed System Model

It is one thing to understand theoretical concepts, but applying them in practice is another. In a similar way, this principle also applies when developing a digital system. It is one thing to design a digital system, but implementing it is another since the implementation gives rise to a whole new set of challenges. The first challenge is that, in hardware, only a limited set of simple components is available. For this reason, Table 4.1 breaks down the wNCMA algorithm in terms of arithmetic components.

| **Step** | **Equation** | **# ADD** | **# MULT** | **# DIV** |
|---|---|---|---|---|
| Filter | $y[n] = \mathbf{c}^H[n]\mathbf{r}[n]$ | 4N-2 | 4N | 0 |
| Error | $\mathrm{e}[n] = \mathrm{Re}\{\mathrm{y}[n]\}^3 + j\mathrm{Im}\{\mathrm{y}[n]\}^3 - \mathrm{Re}\{\mathrm{y}[n]\}$ | 1 | 4 | 0 |
| Update | $\mathbf{c}[n+1] = \mathbf{c}[n] + \mu\mathbf{r}[n]\mathrm{e}^*[n]$ | 4N | 4N+2 | 0 |
| Normalization | $\frac{1}{\alpha+\|\|\mathbf{r}[n]\|\|^2}$ | 2 | 3 | 1 |
| | | 8N+1 | 8N+9 | 1 |

Table 4.1: Required arithmetic components necessary for each step of the wNCMA algorithm, where N is the filter order.

Table 4.1 demonstrates a general principle of the implementation of any adaptive system: Making a given system adaptive increases the required number of components approximately by a factor of two. The first step of the wNCMA algorithm

is filtering. If **c** and **r** were real-valued, N-1 adders and N multipliers would be necessary. However, since **c** and **r** are complex-valued, the number of multipliers and adders increases. The addition of two complex-valued numbers is calculated by

$$(a + jb) + (c + jd) = a + c + j(b + d). \tag{4.1}$$

Figure 4.1 shows how Equation 4.1 can be realized in hardware. In the implementation shown in Figure 4.1, two adders are needed to perform the addition of two complex numbers. The multiplication of two complex numbers is calculated by

$$(a + jb)(c + jd) = ac + bd + j(bc - ad). \tag{4.2}$$

Figure 4.2 shows how Equation 4.2 can be realized in hardware. In the implementation shown in Figure 4.2, four multipliers and two adders are needed to perform the multiplication of two complex numbers. [21] demonstrates that there are in total sixteen different ways how this multiplication can be implemented, some of them being able to eliminate a multiplier for the cost of increasing the number of adders.

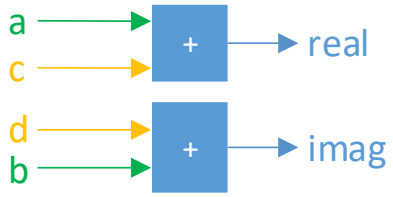



Figure 4.1: Addition of two complex-valued numbers $(a+jb)+(c+jd)$ in hardware.

Back to the filter step in Table 4.1, it is now evident that for each multiplication of the FIR filter, four multipliers and two adders are needed. Scaling this up by the filter order N, 4N multipliers and 2N adders are needed. Last but not least, the real and imaginary result of each multiplication needs to be summed up to produce the final filter output. For this summation, $2(\mathrm{N}-1)$ additional adders are needed.

The next step in Table 4.1 is the error calculation. The analysis of the error calculation in terms of its operational count is simple, since it does not scale with the filter order. Two multipliers each are needed for calculating the real- and imaginary part to the power of three. Also, one adder is needed for calculating $\mathrm{Re}\{\mathrm{y}[n]\}^3 - \mathrm{Re}\{\mathrm{y}[n]\}$. In total, this results in four multipliers and one adder for the error calculation.

The update step is again more complex to analyze since it depends on the filter order. First of all, two multipliers are needed to compute $\mu \mathrm{e}^*[n]$. The result then needs to be multiplied by $\mathbf{r}[n]$, which requires 4N multipliers and 2N adders. Finally, 2N adders are needed to add the result of $\mu \mathbf{r}[n]\mathrm{e}^*[n]$ with $\mathbf{c}[n]$.

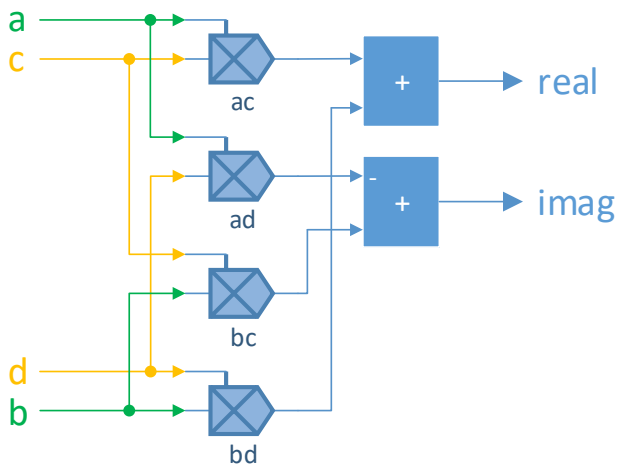



Figure 4.2: Multiplication of two complex-valued numbers $(a + jb)(c + jd)$ in hardware.

For the filter, error and update step, 8N-1 adders and 8N+6 multipliers are needed. However, the normalization has not yet been taken into account. The normalization is defined as $\frac{1}{\alpha+||\mathbf{r}[n]||^2}$. It scales the coefficient update as a function of the input power. In other words, it makes the coefficient update independent of the input power. The input power itself is defined as

$$||\mathbf{r}[n]||^2 = \left(\sqrt{\mathbf{r}^*\mathbf{r}}\right)^2 = \mathbf{r}^*\mathbf{r} = |r[n]|^2 + |r[n-1]|^2 + \ldots + |r[n-N-1]|^2. \quad (4.3)$$

$$= \sum_{k=0}^{N-1} |r[n-k]|^2 \quad (4.4)$$

When implementing Equation 4.3 in hardware, only the power of the newest input sample needs to be calculated in one cycle. The other values have been calculated in the past and can be reused. This fact becomes evident when rewriting Equation 4.4 as

$$\sum_{k=0}^{N-1} |r[n-k]|^2 = |r[n]|^2 + \sum_{k=1}^{N} |r[n-k]|^2 - |r[n-N]|^2. \quad (4.5)$$

In Equation 4.5, only $|r[n]|^2$ needs to be calculated. The other values are available from previous calculations. Figure 4.3 shows how Equation 4.5 can be implemented in hardware. First, the real and imaginary part of the input sample are squared each and then added together. Note that the squared absolute value of a complex number is calculated as $|x|^2 = \left(\sqrt{\mathrm{Re}(x)^2 + \mathrm{Im}(x)^2}\right)^2 = \mathrm{Re}(x)^2 + \mathrm{Im}(x)^2$. Next, the

calculated power of the new input sample is added with the past sum (calculated in the previous cycle), as indicated by Equation 4.5. Finally, $|r[n-N]|^2$ needs to be subtracted to get the correct result and prevent overflow. For this calculation, a buffer for storing N power values is needed.

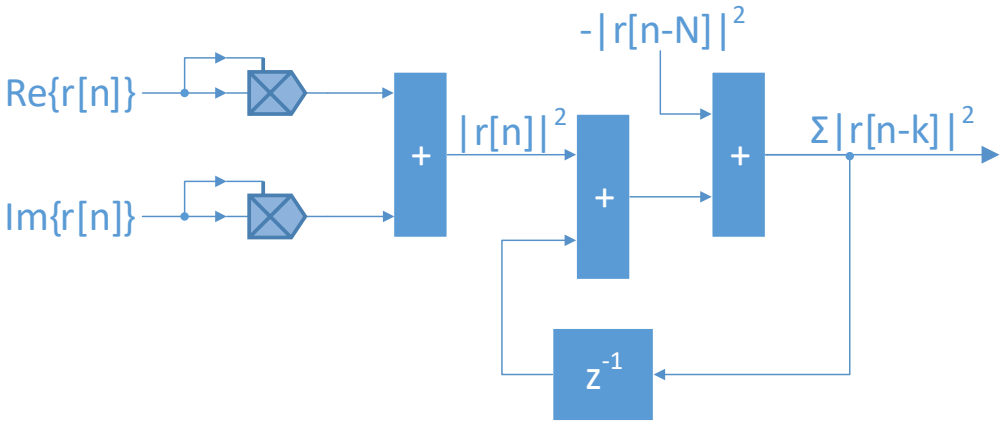



Figure 4.3: The implementation of Equation 4.5 in hardware.

### 4.1.1 Simplifying Multiplications and Divisions

The previous Section discusses the number of basic arithmetic components needed for implementing the wNCMA. It was assumed that those arithmetic operations can be implemented in hardware. While this is true, in fact, both multiplication and division are problematic in hardware design, especially in a low-power application. Low-power goes hand-in-hand with low-area consumption. That is why a digital designer typically tries to avoid or simplify multiplications and divisions since they consume much more area than, for example, additions. A popular way of simplifying multiplications and divisions is to determine the logarithm of the input values, then either add or subtract the logarithms and finally calculate the antilogarithm again.

$$a \times b = 2^{\log_2 a + \log_2 b} \tag{4.6}$$

$$\frac{a}{b} = 2^{\log_2 a - \log_2 b} \tag{4.7}$$

A simple method for calculating the logarithm and antilogarithm of a number was developed by John N. Mitchell [8]. The basic idea is that the leading 1 of an unsigned number gives the correct integer part of the logarithm of that number. For example, consider the number $x = 16$. In binary, this number is expressed as $b10000$. The leading 1 is placed at the bit location 5. If 1 is subtracted from this bit

location $(5-1=4)$, the result is the logarithm. Indeed, $\log_2 x = \log_2 16 = 4$. If $x$ is a number with not only a single 1 in its binary representation, Mitchell suggests an approximation: Simply copy all bits after the leading 1 into the fractional part of the logarithm, leading to a straight-line approximation. Figure 4.4 demonstrates this method visually. The antilogarithm can be calculated in the reverse order. Namely, the integer part of the logarithm determines the location of the leading 1. Right after the leading 1, the fractional part of the logarithm is copied in. [22] suggests that in order to improve the accuracy of the approximation, an additional error-term can be added to the logarithm. However, for the purpose of this thesis, the accuracy of Mitchell's method is sufficient.

| $N$ | $N$ (binary) | Approx 1g $N$ | Approx 1g $N$ (binary) |
|---|---|---|---|
| 1 | 00001 | 0.000 | 000.0000 |
| 2 | 00010 | 1.000 | 001.0000 |
| 3 | 00011 | 1.500 | 001.1000 |
| 4 | 00100 | 2.000 | 010.0000 |
| 5 | 00101 | 2.250 | 010.0100 |
| 6 | 00110 | 2.500 | 010.1000 |
| 7 | 00111 | 2.750 | 010.1100 |
| 8 | 01000 | 3.000 | 011.0000 |
| 9 | 01001 | 3.125 | 011.0010 |
| 10 | 01010 | 3.250 | 011.0100 |
| 11 | 01011 | 3.375 | 011.0110 |
| 12 | 01100 | 3.500 | 011.1000 |
| 13 | 01101 | 3.625 | 011.1010 |
| 14 | 01110 | 3.750 | 011.1100 |
| 15 | 01111 | 3.875 | 011.1110 |
| 16 | 10000 | 4.000 | 100.0000 |
| 17 | 10001 | 4.0625 | 100.0001 |

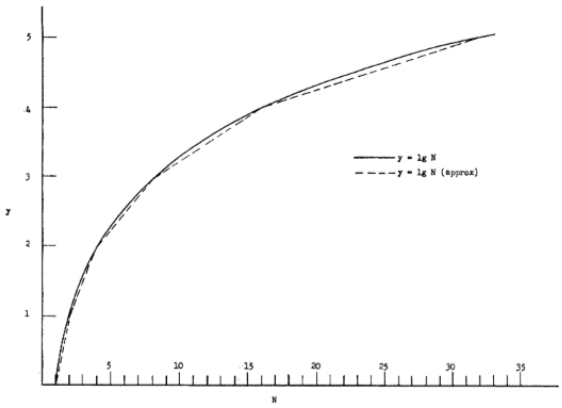



Figure 4.4: Approximation of binary logarithm according to Mitchell [8].

However, there is a crucial aspect that the method by Mitchell does not consider: Fractional numbers at the input and hence the possibility of negative results of the logarithm calculation for inputs between zero and one. As an example, consider the number $x = 0.375$, which in binary is represented as $b0.011$ $(x = 0.375 = 0.25 + 0.125)$. For calculating the logarithm of $x$ similar to the method of Mitchell, again the position of the leading 1 needs to be determined. However, since the leading 1 is in the fractional part of $x$, the leading 1 position is negative. Specifically, the position of the leading 1 in the example is $-2 = b1110$. Finally, the remaining bits of $x$, which are located on the right of the leading 1, are copied into the fractional part. Assuming a fractional length of the logarithm of 3, the fractional part becomes $b100$. The final result is then $\log_2 x = \log_2 0.375 \approx b1110.100 = -1.5$.

Figure 4.5 shows how a logarithmic multiplier or divider can be implemented. First, the absolute values of both inputs are determined, since a logarithm does not produce a real-numbered output for negative inputs. Subsequently, the logarithm is calculated. The logarithms are added or subtracted, depending if multiplication or division shall be performed. Then, the antilogarithm of the multiplication or division is calculated. Special care is necessary for input values that are zero. The Zero-Correction block in Figure 4.5 sets its output to zero if at least one of the inputs A and B is zero. Otherwise, the Zero-Correction block is simply passing-

through its input. Finally, the Sign-Conversion block is performing unsigned-to-signed conversion, based on the signs of both inputs A and B. To be specific, if A and B have opposite signs, the output of the Sign-Conversion block is negative. Otherwise, it is positive.

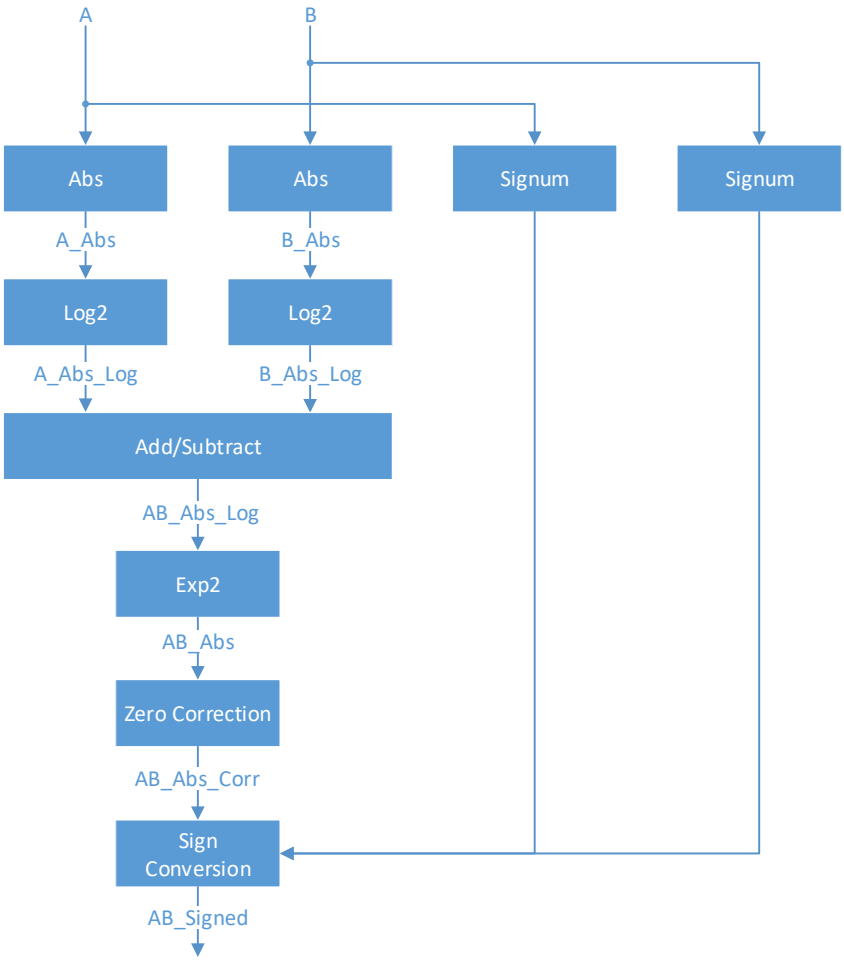



Figure 4.5: Implementation of a logarithmic multiplier or divider in hardware.

### 4.1.2 High-Level wNCMA Implementation

Section 4.1 demonstrated what and how many arithmetic components are necessary to implement each step of the wNCMA. Subsequently, Section 4.1.1 explained how multiplications and divisions can be simplified, such that area is saved when implementing those operations in hardware. Based on this previous analysis, the current Section examines the wNCMA implementation as a whole. Figure 4.6 shows the high-level implementation of the wNCMA. In this high-level implementation, the

input signal x[$n$], the error signal e[$n$], as well as the output signal y[$n$] are complex signals. Performing mathematical operations on them requires arithmetic components that are able to handle complex numbers. For addition and multiplication, the high-level implementation relies on the components shown in Figure 4.1 (addition) and 4.2 (multiplication). The exception is the error calculation shown in the bottom of Figure 4.6. The error calculation specifically uses the real and imaginary part of the output signal y[$n$] separately, making it possible to use standard adders and multipliers. See Equation 3.2 for how the error is calculated.

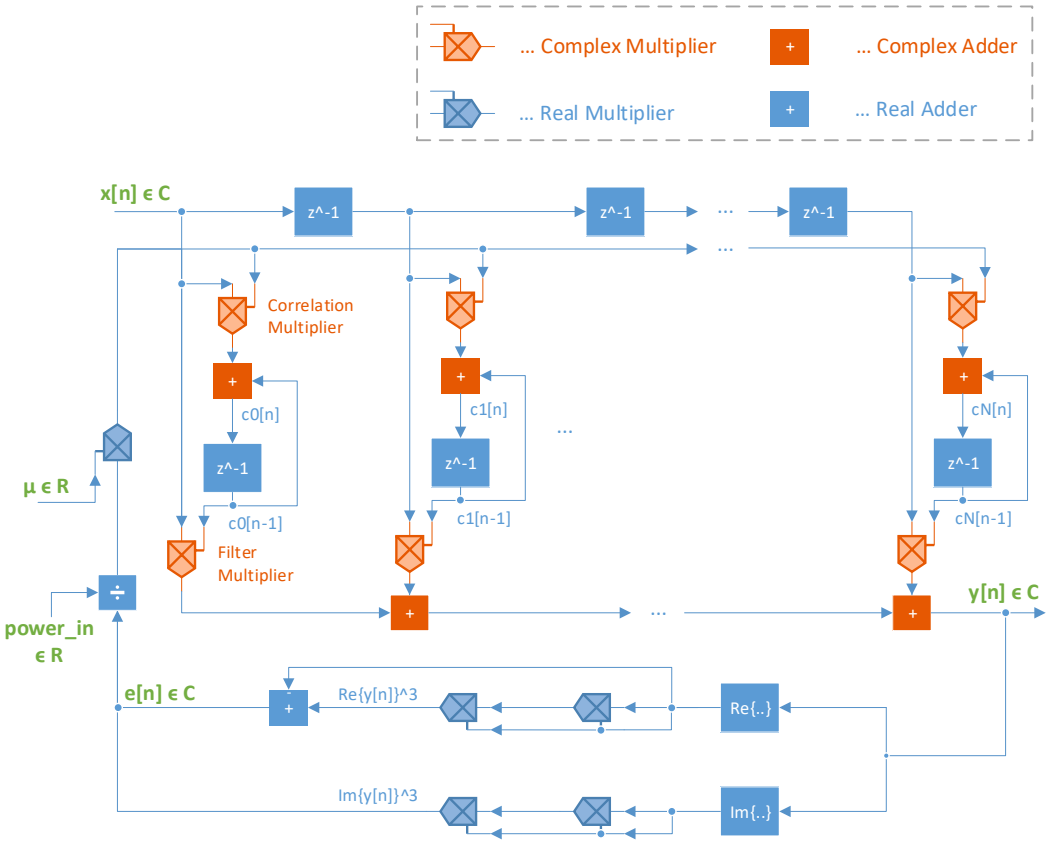



Figure 4.6: A high-level implementation of the wNCMA algorithm. Four different types of arithmetic components are used: Standard real multipliers, standard real adders, complex multipliers as shown in Figure 4.2 and complex adders as shown in Figure 4.1.

Visualizing a high-level implementation of the wNCMA algorithm as in Figure 4.6 helps to understand how different parts of the algorithm work together. The high-level implementation shows that there is a feedback-loop in the system (as expected from an adaptive system), which is formed by the error calculation and coefficient update, as already seen in Figure 3.2. However, the feedback-loop in Fig-

ure 4.6 additionally reveals interdependencies of components. The most prominent dependency is that the correlation multipliers need to wait for the filter multipliers to finish. This means that the filter and correlation multipliers are never active at the same time. Thus, parallel filter and correlation multipliers cannot be fully utilized, while the critical path of the equalizer is increased. This issue can be avoided by inserting delay elements between x[$n$] and the correlation multipliers, as well as before the step-size multiplier. In Chapter 6, the delayed wNCMA is noted as potential further improvement, in case a shorter critical path is required.

## 4.2 Floating-Point to Fixed-Point Conversion

The most convenient way of implementing a Digital Signal Processing (DSP) system is by using a floating-point number format. A floating-point number consists of a sign, mantissa and exponent. The resulting value is calculated by $x_{\text{float}} = \text{sign} \times \text{mantissa} \times \text{base}^{\text{exponent}}$. Typically, a floating-point number either consumes 32 bits or 64 bits. Figure 4.7 shows how many bit each part of a 32 bit floating-point number consumes.

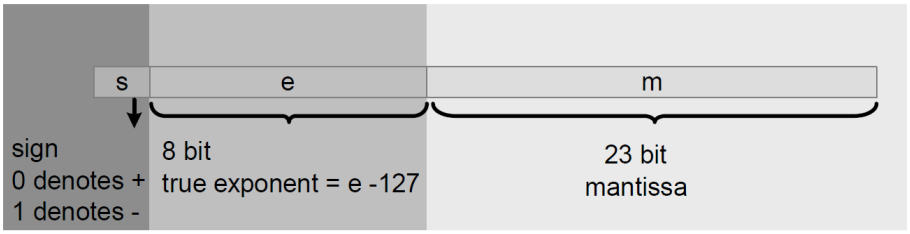



Figure 4.7: IEEE format for single-precision 32bit floating point number [9].

As Khan writes in his book Digital Design of Signal Processing Systems:

> Floating point representation works well where variables and results of computation may vary over a large dynamic range. In signal processing, this usually is not the case. In the initial phase of algorithm development, though, before the ranges are conceived, floating point format gives comfort to the developer as one can concentrate more on the algorithmic correctness and less on implementation details. If there are no strict requirements on numerical accuracy of the computation, performing floating point arithmetic in HW is an expensive preposition in terms of power, area and performance, so is normally avoided [9, p. 94-95].

An alternative to floating-point format is the $Qn.m$ fixed-point format. In this format, n bits are used for the integer part (left-hand side of the binary point) and m bits are used for the fractional part (right-hand side of the binary point). As an example, the bit fields and their weights of a signed Q2.7 fixed-point number are shown in Figure 4.8.

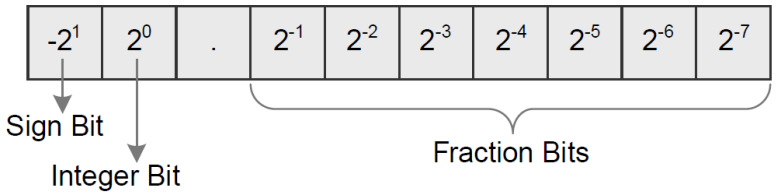



Figure 4.8: Bit fields and their weights of a signed Q2.7 fixed-point number [9].

There are multiple ways of how a floating-point algorithm in MATLAB can be converted to fixed-point. One possibility is to rely on the Fixed-Point Designer from MathWorks.

> Fixed-Point Designer™ provides data types and tools for developing fixed-point and single-precision algorithms to optimize performance on embedded hardware. Fixed-Point Designer analyzes your design and proposes data types and attributes such as word length and scaling. You can specify detailed data attributes such as rounding mode and overflow action, and mix single-precision and fixed-point data. You can perform bit-true simulations to observe the impact of limited range and precision without implementing the design on hardware [23].

Another way is to manually convert the value of each variable in MATLAB to its fixed-point representation. A floating-point value is transformed to fixed-point format by

$$x_{\text{fx}} = \text{round}\left(x_{\text{fl}} \times 2^{m}\right), \tag{4.8}$$

where $x_{\text{fl}}$ is the floating-point value, $m$ the number of fractional bits of the chosen $Qn.m$ fixed-point format and $x_{\text{fx}}$ the fixed-point value. In other words, m bits of the number are brought to the integer part and the rest of the fractional bits are dropped with rounding. Additionally, the result of Equation 4.8 might be saturated, meaning that the result is higher or lower than the number of integer bits allow. Therefore, the result also needs to be clipped.

I chose the second method for fixed-point conversion for multiple reasons:

- The educational benefit is higher when using the second method, instead of letting a tool do the job.
- Relying on the Fixed-Point Designer by MATLAB slows down simulation performance.
- It is not transparent what exactly the Fixed-Point Designer is doing in the background.

### 4.2.1 Numerical Aspects

Bitgrowth is one of the most serious issues when implementing algorithms in fixed-point format, especially if the algorithm contains feedback loops. If two fixed-point numbers are multiplied, the resulting bitwidth is given by

$$Qn.m = Q(n1 + n2).(m1 + m2), \tag{4.9}$$

where n1 and n2 are integer widths and m1 and m2 fractional widths of the numbers that are multiplied. If two fixed-point numbers are added, the resulting bitwidth is given by

$$Qn.m = Q\max(n1, n2).\max(m1, m2). \tag{4.10}$$

Furthermore, overflow needs to be considered. Overflow occurs if two positive or negative numbers are added and the sum requires more than the available number of bits. Figure 4.9 shows the effect of overflow. If overflow occurs, an error equal to the dynamic range of the number is introduced. To avoid this effect, the bitwidth needs to be increased by 1 or overflow needs to be detected and the number saturated. Saturation means that the number is clamped to the maximum or minimum value.

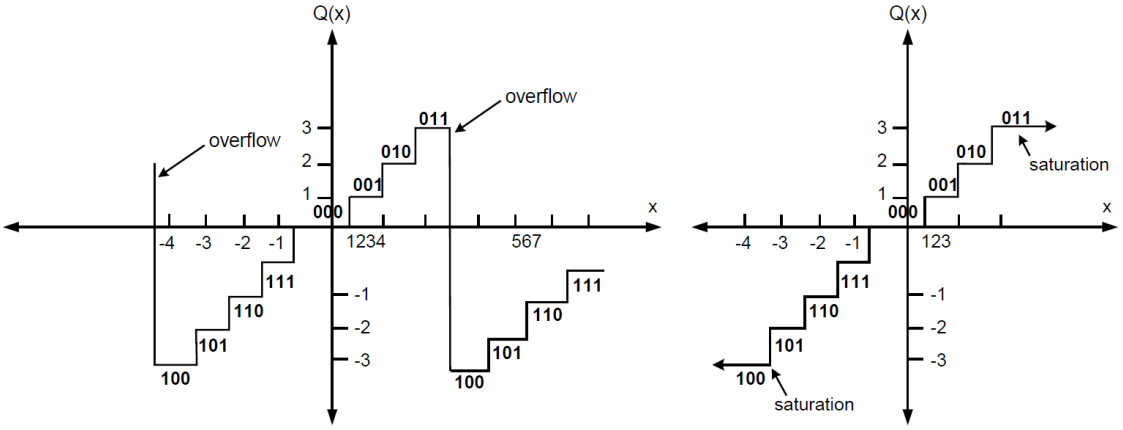



Figure 4.9: The error introduced by overflow for a 3-bit signed number on the left-hand side. Saturating a 3-bit signed number to avoid overflow on the right-hand side [9].

Not controlling the bitgrowth of fixed-point numbers can have catastrophic consequences in adaptive systems, since they typically contain feedback loops. In feedback loops, fixed-point numbers keep growing in every cycle, if the designer is not taking the right measures. For dealing with bitgrowth, the designer can apply clipping and quantization, which are shown in Figure 4.10. Clipping reduces bitwidth from the MSB (Most Significant Bit) side by saturating a number to a given maximum or minimum value. However, in the allowed range, the accuracy stays the

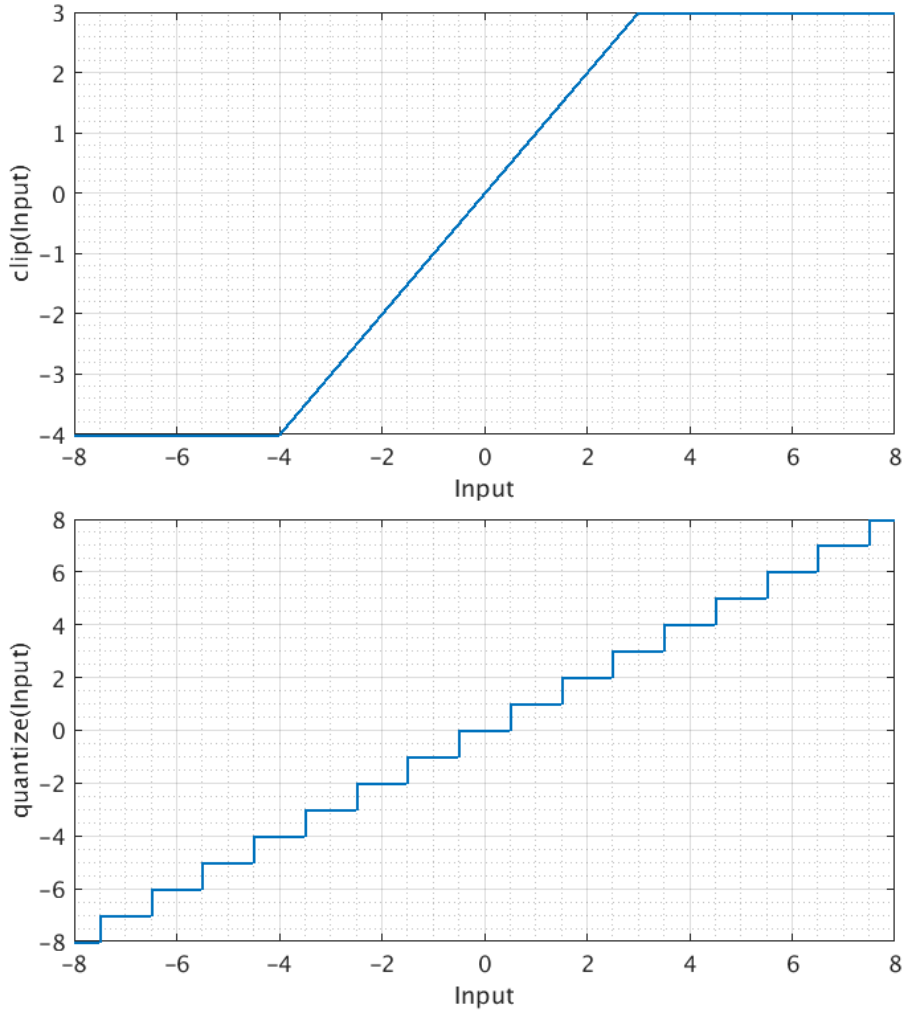



Figure 4.10: The top figure shows clipping, where a signed number can grow only until a certain maximum and minimum value. The bottom figure shows quantization, where the accuracy of a signed number is reduced.

same. Quantization reduces bitwidth from the LSB side, by either truncation or rounding. While quantization does not limit range, it affects accuracy.

The measures taken to control bitgrowth in the equalizer are demonstrated in Figure 4.11. In the top left corner, the FIR filter calculates the dot-product of the input signal and the filter coefficients. For this operation, the results of several multipliers in parallel are added together to produce the filter output. Thinking about bitgrowth, the result of each multiplier has a bitwidth of $Q(3+6).(10+10) = Q9.20$, as indicated by Equation 4.9. Furthermore, the result of the addition is increased in bits to avoid overflow. Assuming a filter order of four, four multiplication results need to be added in every cycle. However, that does not mean that the bitwidth of the result of the addition needs to be increased by four bits. In gen-

eral, if N numbers with bitwidth M are added, the bitwidth of the resulting number needs to be ceil(log2($(2^M - 1) \times N$)). Thus, the filter output needs to have ceil(log2($(2^{29} - 1) \times 4$)) = 31 bit. Therefore, the integer part increases by 2bit, resulting into a Q11.20 format.

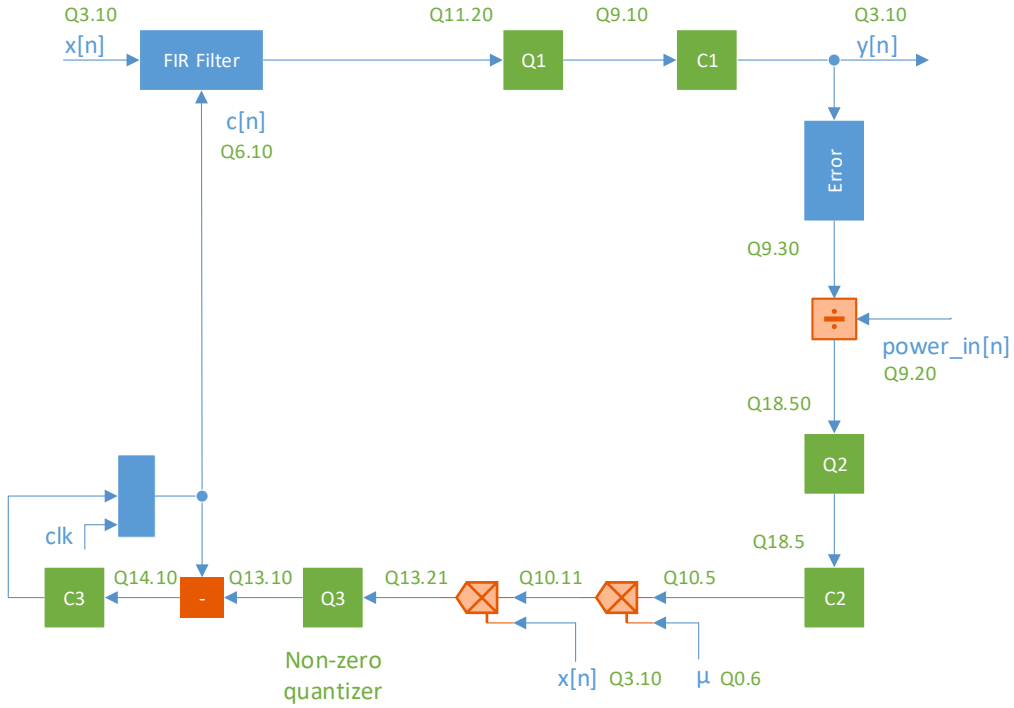



Figure 4.11: Quantizers and clippers are used to control bitgrowth in the wNCMA equalizer.

A project requirement was that the output of the equalizer has the same format as the input. Therefore, the output of the filter is quantized and clipped to a Q3.10 format. Next, the error is calculated. The bitgrowth due to the error calculation can be analyzed in a similar fashion as the filter. Squaring the equalizer output results in a $Q(3+3).(10+10) = Q6.20$ format. Multiplyng that result again with the equalizer output results in a $Q(6+3).(20+10) = Q9.30$. ceil(log2($(2^{13}-1)^3 + (2^{13}-1)$)) = 39 confirms that the whole bitwidth of 39 bits is needed.

To normalize the coefficient update term, a division by the input power is required. The input power has a Q9.20 format and can be implemented as shown in Figure 4.3. The division itself is implemented as a simple shift operation. Considering the Q9.20 format of the input power, the error can experience at maximum a shift of 9 bits to the left and 20 bits to the right. Therefore, the result of the division has a format of $Q(9+9).(30+20) = Q18.50$. At this point, the number of bits has increased drastically. To save area in the subsequent multiplier, the normalized error

is clipped and quantized.

After the multiplication of the error with the step-size and the input signal, the coefficient update term is complete. The fractional part of the coefficient update term now is brought down to the desired fractional coefficient width. The quantizer performing this reduction of the fractional coefficient width takes on a special role, as it is the last quantization taking place in the signal chain. To understand its importance, it is necessary to first understand the changing convergence behavior of the equalizer due to fixed-pointing. This effect is demonstrated in Figure 4.12. What happens is that due to the quantization in the feedback path of the equalizer, the coefficient update becomes zero before the optimum is reached. Because the coefficient update term becomes smaller and smaller over time, at some point it is simply quantized to zero, since its low value is below resolution. Formally it can be said that loss of convergence is possible if

$$|\mu e[n] x_k[n]| < \frac{1}{2}\text{LSB}, \tag{4.11}$$

leading to $\mathbf{c}[n] = \mathbf{c}[n-1]$, also called stalling. Stalling can be interpreted as an asymptotic bias.

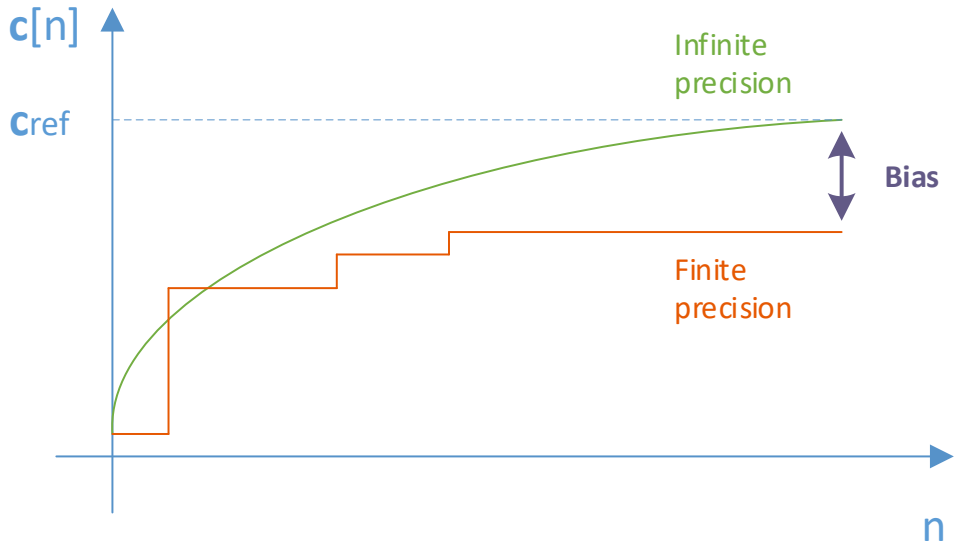



Figure 4.12: Reducing the feedback precision of an adaptive system leads to introducing an asymptotic bias.

There are at least two possible workarounds to deal with this problem. First, the step-size can be increased, which directly increases the coefficient update. This

solution requires a lot of care by the designer. By how much must the step-size be increased? Is the system still converging at all? A second solution is to use a non-zero quantizer (also called mid-rise quantizer) for Q3 in Figure 4.11. As the name suggests, a non-zero quantizer never outputs the value 0. Therefore, the coefficients are always updated, no matter how small the coefficient update term is and no matter if the optimal coefficients are already reached. This effect leads to a greater variance of the final coefficients, but in the end the final coefficients are closer to the optimal coefficients than without a non-zero quantizer.

Another potential issue is the undamped loop in the feedback of the equalizer (after Q3). In a typical scenario with proper signal excitation, the quantization noise introduced by Q3 can be viewed as white-noise. However, this representation might not always be true, and in such a case would lead to the accumulation of quantization noise. To prevent the uncontrolled numerical growth in the coefficient loop, a damping factor can be introduced such that

$$\mathbf{c}[n] = \lambda \mathbf{c}[n-1] + \mu e[n]\mathbf{x}[n]. \tag{4.12}$$

$\lambda$ gradually pulls the coefficients towards zero, working against the accumulation of deterministic noise. However, $\lambda$ also introduces a bias in steady-state (pull towards zero). For the wNCMA equalizer, this effect is not expected to cause problems. Because the equalizer is only active during a frame (proper signal excitation can be assumed) and the coefficients are reset after a frame, the accumulation of deterministic quantization noise is not a relevant issue.

## 4.3 High-Level Synthesis

A recent trend, which now also the semiconductor industry seems to catch on to, is HLS. The idea is that RTL code is not written by hand, but generated by a tool on the basis of high-level language code. For implementing DSP algorithms in hardware, it is common to first do the whole design in MATLAB, move from floating-point to fixed-point format and finally write the RTL code. In a way, the same algorithm needs to be written in two different languages. Since only the high-level language code is necessary when relying on an HLS tool, the designer needs less time and thus time-to-market is reduced. Furthermore, since the RTL code is automatically generated, the pure translation task is not prone to human error.

MathWorks offers multiple tools regarding HLS and highlights additional benefits on their website:

> Working at a high level of abstraction lets hardware designers focus on developing the functionality in the context of a hardware architecture that meets their project requirements. Since many ASIC and FPGA designs start as algorithms in MATLAB® and Simulink®, these are natural environments to perform this design and verification.

> With high-level synthesis, hardware designers can focus at a high level without implementation detail enables easy adjustment to changes, reuse across projects, and more productive functional verification.
>
> High-level synthesis does require some amount of hardware architecture detail, such as parallelism, some notion of timing where appropriate, and hardware data types, which are usually fixed point. Most high-level synthesis users rely on graphical environments such as Simulink to visualize the architecture and data flow. Some high-level synthesis offerings such as HDL Coder™ offer automatic fixed-point conversion or even RTL implementation of native floating-point operations [24].

In the course of this thesis, the HDL Coder by MathWorks was used to automatically generate RTL code for the FS. MathWorks offers a detailed description of how workflow for HDL Coder [25]. The basic approach is to use the `persistant` keyword to model registers. Any other variable is interpreted as a wire. However, the designer needs to be aware that the resulting RTL code is not easily readable by a human being. Consequently, the resulting RTL modules need to be viewed as blackbox.

## 4.4 Verification

Verification of a hardware design is not just "nice-to-have", but absolutely necessary. Nowadays, verification engineers in the semiconductor industry go to great lengths to make sure the design works as expected, since the consequences of an error can be so costly. Depending on the size, verification is performed on different levels, going from the lowest component-level up to system-level. Some advanced testing methods are exhaustive testvector-generation, random testing and model-based testing. For the verification of this equalizer however, the simplest method of all is used, which is blackbox-testing, as shown in Figure 4.13.

In blackbox-testing, the internals of the Device under Test (DUT) are invisible to the tester. Therefore, the tester can only set the input signals of the DUT and compare the output signals against their expected values. If no unexpected output is observed, the DUT conforms to the model available to the tester (in the input-output conformance sense). Clearly, blackbox-testing does not reveal any errors of internal states, which simply by luck produce the expected output.

The activity diagram of the equalizer testbench is shown in Figure 4.14. A demonstration of such an executed blackbox-test in SimVision is shown in Figure 4.15. The signals `eq_in_i` and `eq_in_q` are the real and imaginary part of the input signal, `eq_out_correct_i` and `eq_out_correct_q` the expected output and `eq_out_i` and `eq_out_q` the actual output. As soon as there is a mismatch between the expected and actual output, the test stops immediately. The input signals driving the DUT are stimuli exported from MATLAB. The commands used for the MATLAB stimuli export are shown in Listing 4.1.

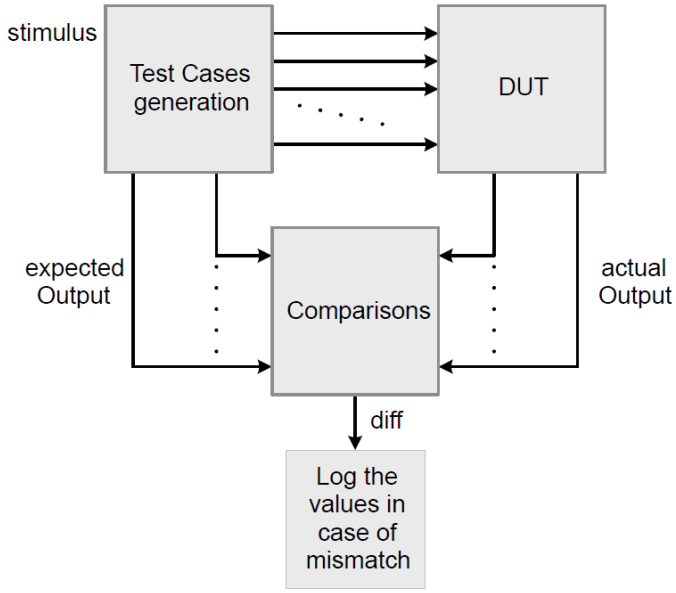



Figure 4.13: Concept of blackbox-testing [9].

```
1 fileID = fopen('eq_in_I.dat', 'w'); fprintf(fileID, '%.f\n', [ real(EQ.debug.eq_in(:)) ]);
2 fileID = fopen('eq_in_Q.dat', 'w'); fprintf(fileID, '%.f\n', [ imag(EQ.debug.eq_in(:)) ]);
3 fileID = fopen('eq_out_I.dat', 'w'); fprintf(fileID, '%.f\n', [ real(EQ.debug.eq_out(:)) ]);
4 fileID = fopen('eq_out_Q.dat', 'w'); fprintf(fileID, '%.f\n', [ imag(EQ.debug.eq_out(:)) ]);
5 fileID = fopen('is_signal_detected.dat', 'w'); fprintf(fileID, '%.f\n', [ EQ.debug.
      IS_signal_detected(:) ]);
```

Listing 4.1: MATLAB commands used for RTL stimuli export.

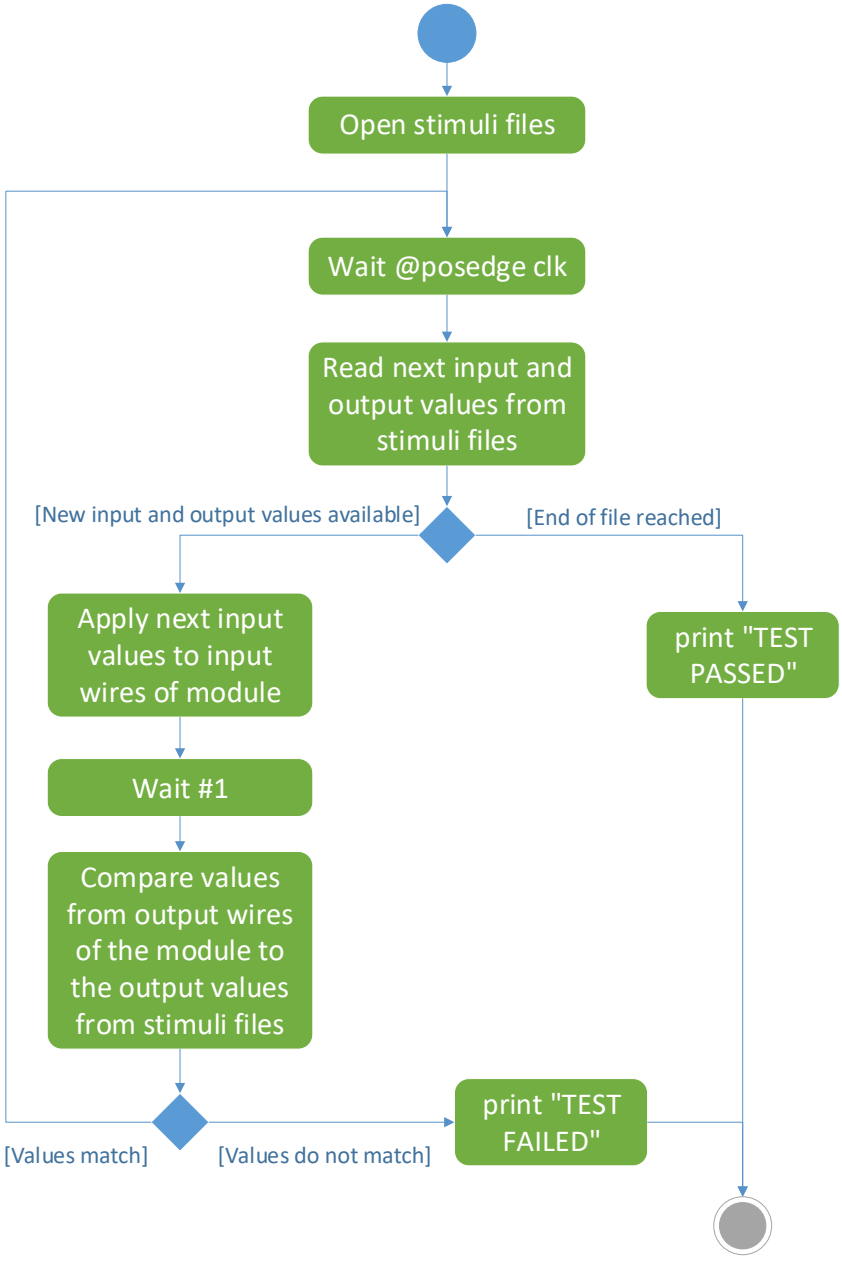



Figure 4.14: UML activity diagram of the equalizer testbench.

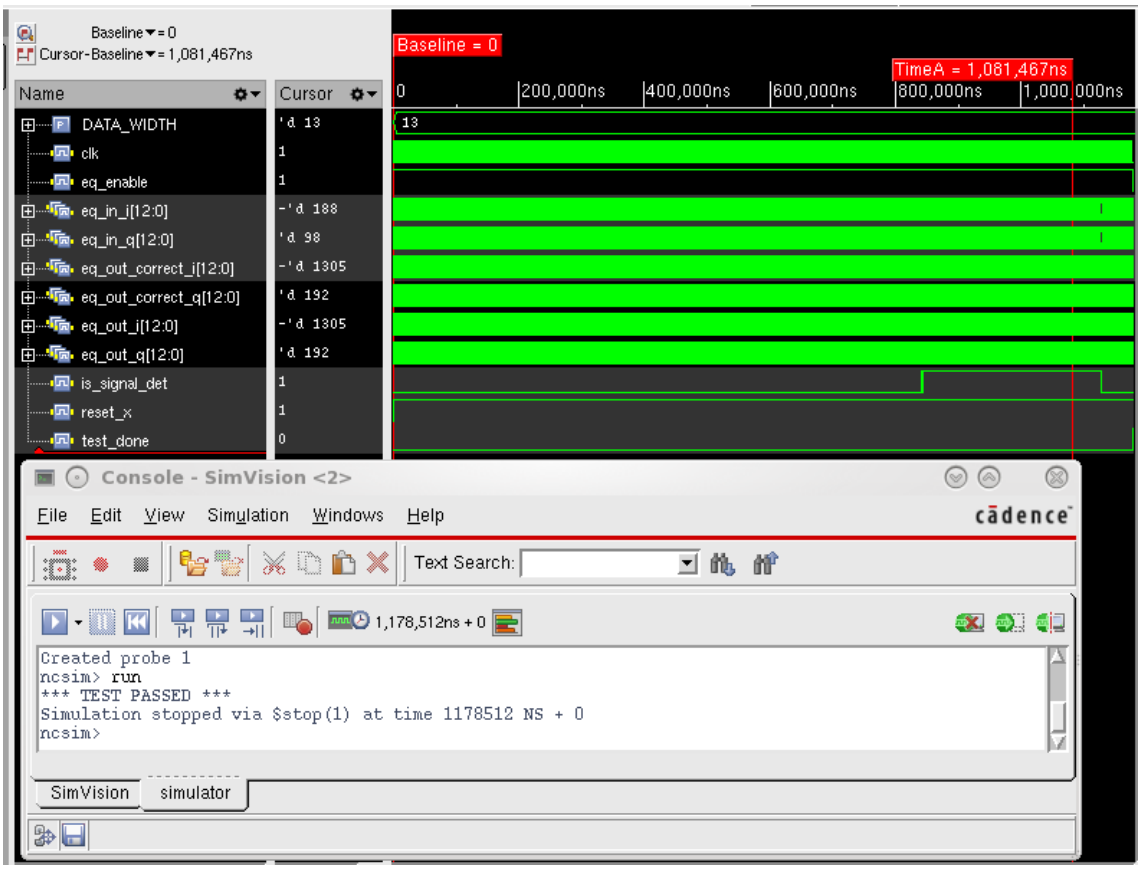



Figure 4.15: Screenshot from SimVision, showing an executed blackbox-test for verifying the equalizer implementation.

## 4.5 Development Environment

Table 4.2 presents the operating system and tools used in the course of this thesis. Because this thesis was performed in cooperation with the company NXP in Gratkorn, Austria, I had access to industry-level tools for RTL development from Cadence (Incisive and SimVision). To perform simulations with Incisive, I used the command *irun*.

```
1 irun <testbench file> <RTL file 1> <RTL file 2> +gui -access rw
```

Listing 4.2: Irun command used for RTL simulations.

| | **Name** | **Version** |
|---|---|---|
| Operating System | Red Hat Enterprise Linux Server 64-bit | Release 6.7 (Santiago) |
| Concept Development Tool | MATLAB | R2019b |
| Verilog Editor | VIM | 7.4 |
| RTL Simulator | Cadence Incisive | 15.20 Build 229 |
| Waveform Analyzer | SimVision | 15.20-s047 |

Table 4.2: Description of the development environment.

# Chapter 5

# Results

## 5.1 MATLAB Simulations

### 5.1.1 Floating-Point

This chapter explores the optimal configuration and behaviour of the equalizer for one specific channel model. This channel model is not the same as the one presented in Section 2.1.2, but rather a real-world channel from an NFC product developed by NXP. Instead of evaluating the equalizer performance on a simplified model, it is more interesting to see how the equalizer performs on the mentioned real-world channel.

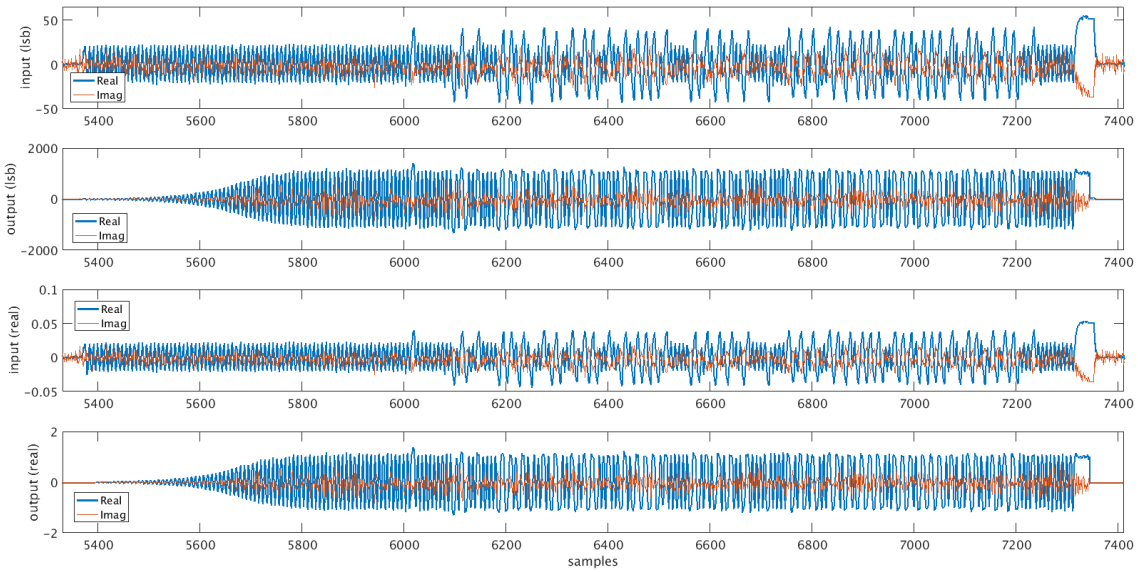



Figure 5.1: Input and output of the equalizer, when receiving a frame with twelve bytes at a coupling of $k = 0.08$.

Input and output of the equalizer, when receiving a frame with twelve bytes at

a coupling of $k = 0.08$, are shown in Figure 5.1. In fact, there are two plots each for the input and output. The upper two plots show the amplitude of the input and output as fixed-point values (see Equation 4.8), whereas the lower two plots show the input and output as floating-point values. Be reminded that ten bits are used for the fractional part, so for example a floating-point number $x_{\text{fl}} = 1.0$ in fixed-point format is $x_{\text{fx}} = \text{round}\,(1.0 \times 2^{10}) = 1024$. Therefore, the equalizer is massivly scaling up the output, since the wNCMA error is only reduced if the output gets closer to $\pm 1$. An additional effect can be observed in Figure 5.1, namely that the channel causes a sudden increase in amplitude whenever there is a phase change. This effect is removed by the equalizer.

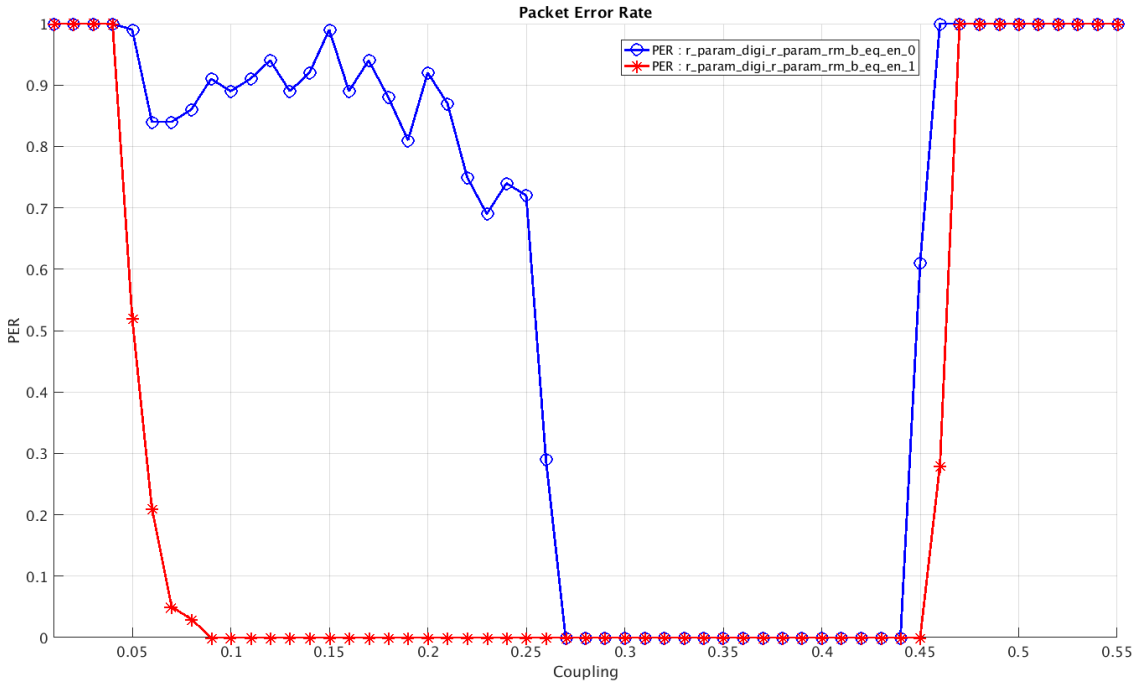



Figure 5.2: Packet Error Rate over coupling with (red curve) and without (blue curve) equalization.

Figure 5.2 shows the PER over coupling with the real-world channel for a data rate of 1.695 Mbit/s. The blue curve shows the PER without equalization, which serves as a baseline for comparison. The red curve shows the PER with equalization, demonstrating a huge increase in performance thanks to the wNCMA. For every dot in this plot, one-hundred frames with four bytes of data each are sent from card to receiver (in simulation). The receiver then tries to decode the information sent by the card. As soon as a single bit detected by the receiver is wrong, the whole frame counts as failed. Such a simulation gives an in-depth view of how well the receiver performs overall. Figure 5.2 shows that only in the coupling range from 0.27 to 0.44, frames can be received with approximately no errors without equalization. Subsequent paragraphs explain how the equalizer parameters are chosen, such that

a result as the red curve in Figure 5.2 can be achieved.

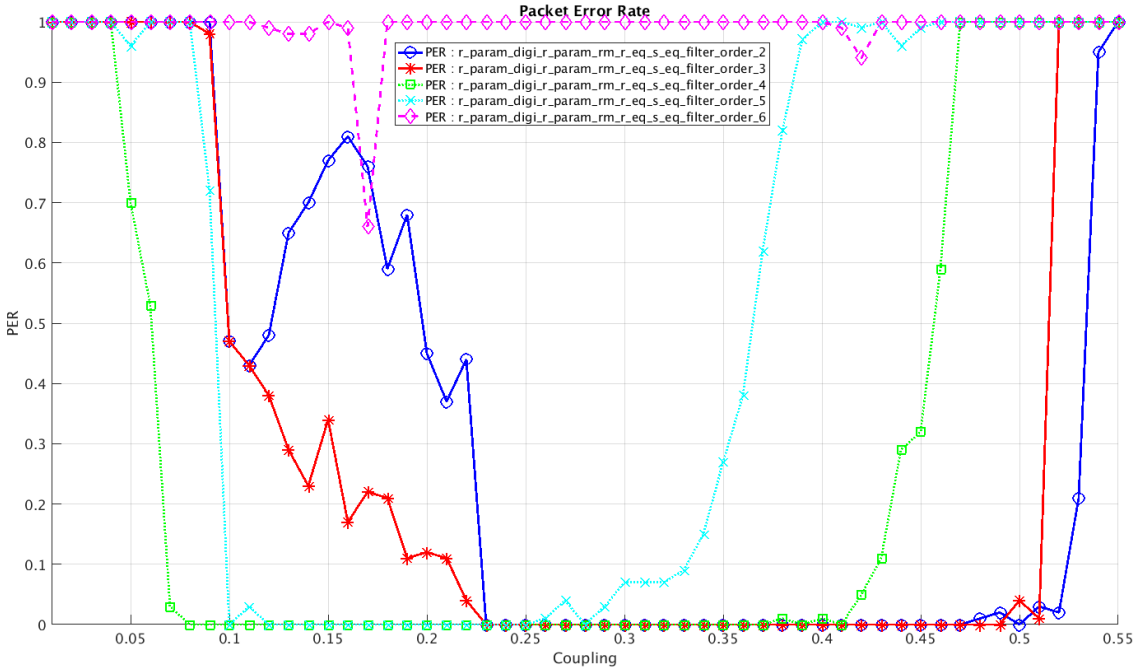



Figure 5.3: Effects of filter order variation on the PER over coupling.

The first parameter to investigate is the filter order. For the step-size, a value of 0.01 is proposed by [5], whereas the nearest factor of two is $\frac{1}{64}$. By using factor of two only, we can replace the step-size multiplier with a shifter and thus save additional area. Figure 5.3 shows the PER over coupling when varying the filter order. It turns out that a filter order of 4 seems to be the best choice since a robust communication is already possible at a coupling value as low as 0.07. In other words, the receiver is most sensitive with a filter order of 4. Also, the PER stays zero until a coupling value of 0.41.

Figure 5.4 shows the effects of step-size variation on the PER. Decreasing the step-size from $\frac{1}{64}$ to $\frac{1}{128}$ clearly worsens equalization performance. However, increasing it to $\frac{1}{32}$ improves the performance. Further increase does not have any benefit, therefore the optimal step-size is $\mu = \frac{1}{32}$.

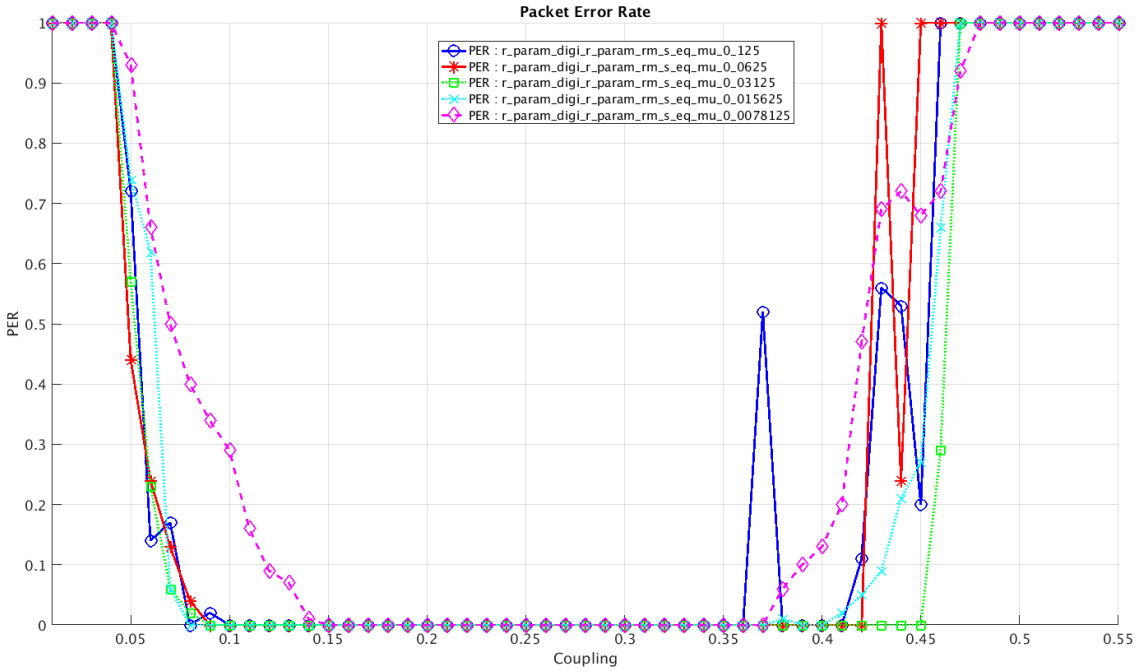



Figure 5.4: Effect of step-size variation on the PER over coupling.

### 5.1.2 Fixed-Point Simulations

For an efficient implementation of the equalizer on an ASIC, it was necessary to convert the equalizer to fixed-point format. Amongst others, a lower area is an advantage compared to a floating-point implementation. Section 4.2 presents how the floating to fixed-point conversion was done, while Section 4.11 explains the effects of a fixed-point format in an adaptive system. This chapter explores the optimal configuration of each clipper and quantizer presented in Figure 4.11.

The most straightforward components for bitwidth reduction to analyze are the quantizer and clipper after the filter, namely Q1 and C1 (see Figure 4.11). Because the required bitwidth for the equalizer is already given by project requirements, no analysis of different bitwidths needs to be done. Project requirements state that the output of the equalizer needs to have the same bitwidth as the input, which is thirteen bits. These thirteen bits are split into three bits for the integer part, and ten for the fractional part (Q3.10). Ideally, the output of the equalizer is $\pm 1$, which requires two bits in the integer part. However, as a safety margin, an additional bit in the integer part makes sense. Three bits result into an output range of -4 to 3. Consequently, the other ten bits can be used for the fractional part, resulting into a final output range of -4 to 3.999. Figure 5.5 shows that fixed-pointing the forward path (Q1 and C1 only) does not impact the overall performance.

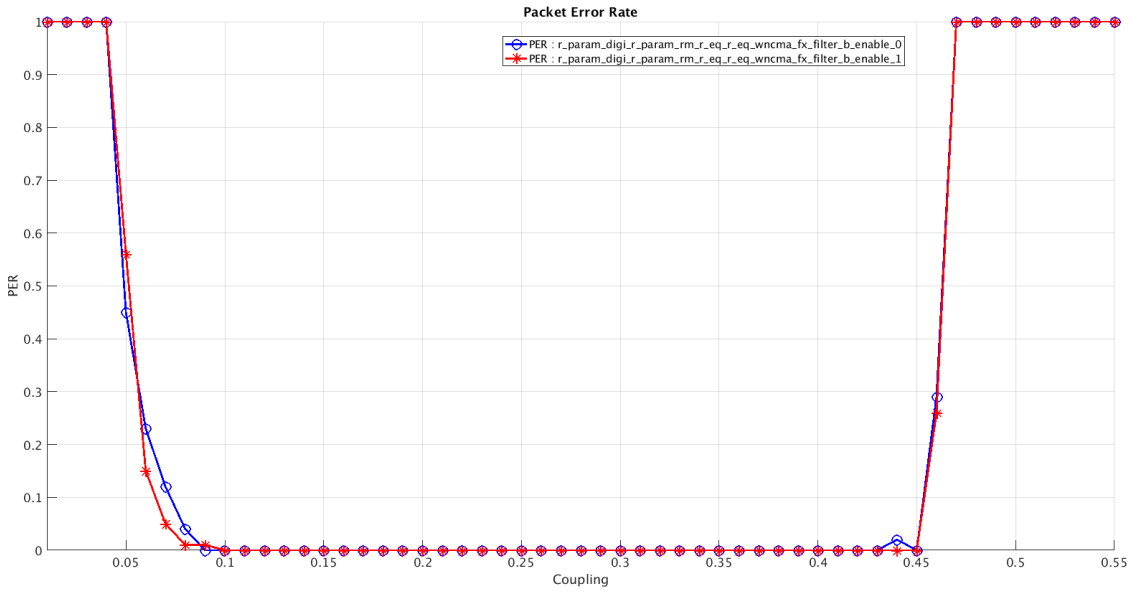



Figure 5.5: Effect of fixed-pointing the forward path (Q1 and C1 only) on the PER over coupling.

Figure 5.6 shows the effects of clipping and quantizing the error signal after normalization (Q2 and C2 in Figure 4.11). A fixed-point format of Q18.50 guarantees no information loss. In other words, with a Q18.50 format, no clipping and quantization is necessary. Figure 5.6 shows that greatly reducing the integer width (from 18 to 5) negatively impacts sensitivity. Furthermore, completely neglecting the fractional part of the error signal increases PER for high coupling scenarios (above $k = 0.42$). A Q10.5 fixed-point format seems the most reasonable choice, as it delivers similar performance to the full resolution format of Q18.50, as well as greatly reduces bitwidth and therefore saves area in the subsequent multipliers (see Figure 4.11).

The effect due to fixed-pointing the error shown in Figure 5.6 is clearly a coarse analysis. Only nine samples of the whole set of possible bitwidth configurations have been simulated. Therefore, the PER over coupling for a few more configurations is shown in Figure 5.7. As discussed in the previous paragraph, the performance for a Q18.50 and Q10.5 fixed-point format is nearly the same. Therefore, it makes sense to analyze more closely the performance for an integer bitwidth slightly below 10 and a fractional bitwidth slightly below 5. As can be seen in Figure 5.7, the PER already slightly worsens when using a Q8.3 format for the error. For that reason, it makes sense to stick with a Q10.5 format.

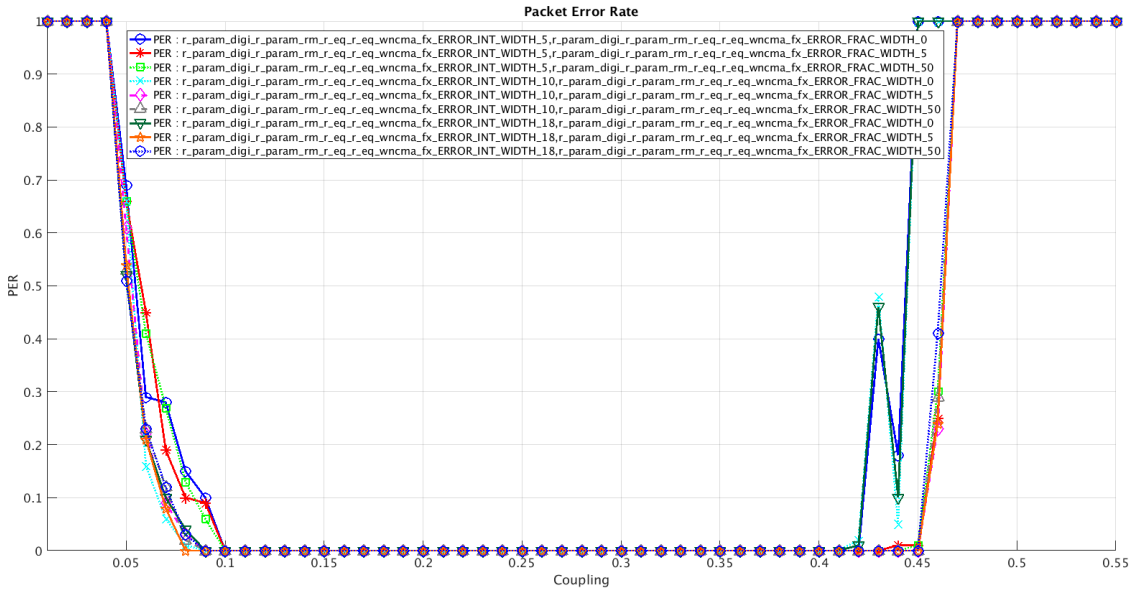



Figure 5.6: Effect of partly fixed-pointing the backward path (Q2 and C2) on the PER over coupling (coarse).

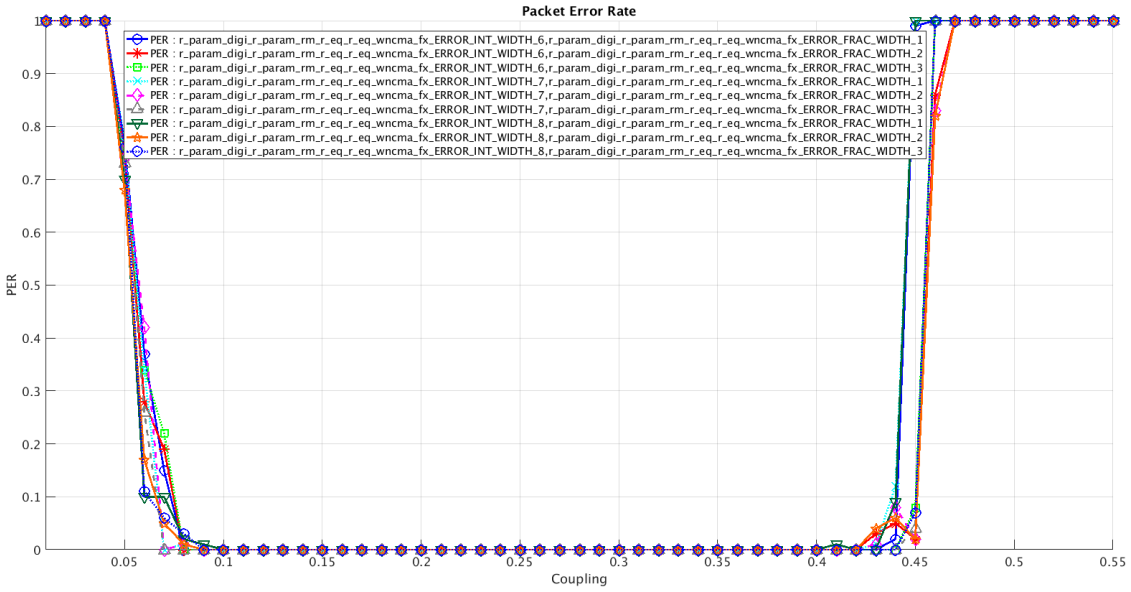



Figure 5.7: Effect of partly fixed-pointing the backward path (Q2 and C2) on the PER over coupling (fine).

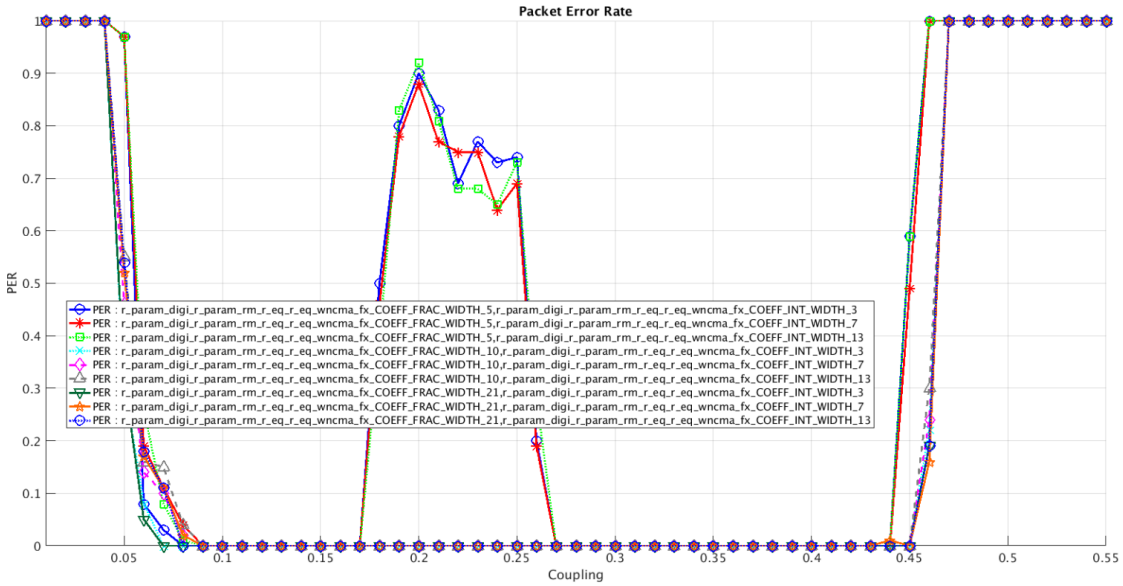



Figure 5.8: Effect of fixed-pointing the backward path (Q3 and C3) on the PER over coupling (coarse).

Figure 5.8 shows the effect of fixed-pointing the coefficients on the PER. A bitwidth of 5bit for the fractional part causes a sudden increase of the PER in the mid-coupling range. Surprisingly, even a bitwidth of 3bit for the integer part is sufficient for proper performance. Figure 5.9 demonstrates that for a coupling $k = 0.08$, the coefficients actually grow much higher than 3bit in the integer part would allow. Therefore, the coefficients experience clipping throughout the majority of the frame. Thus, although the performance does not seem to suffer from heavy clipping of the coefficients in low-coupling range, it is a better choice to increase the integer width until 6bit.

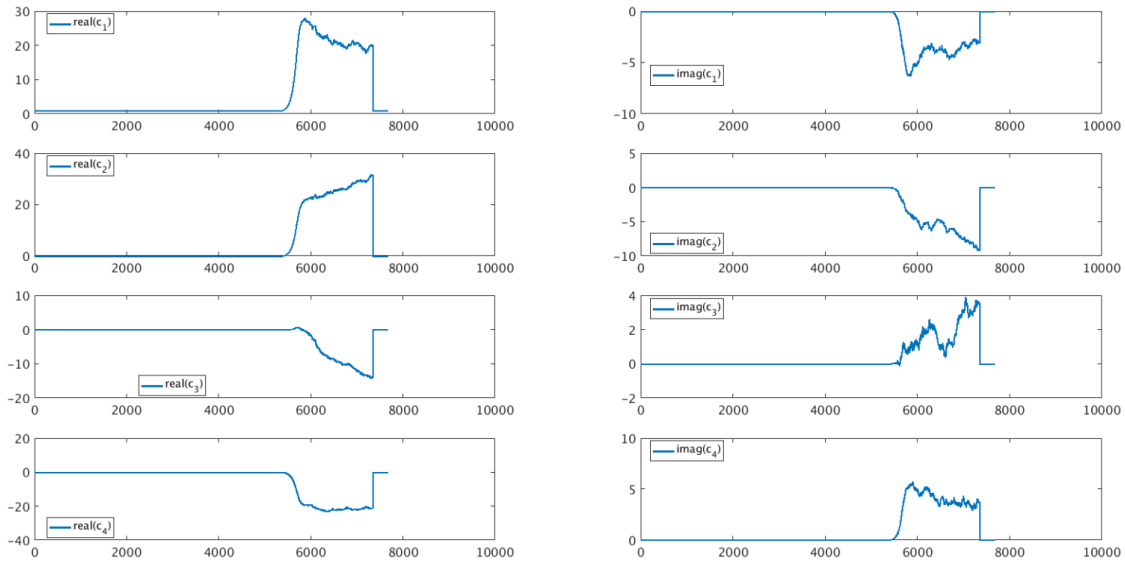



Figure 5.9: Coefficient development over time for a twelve-byte frame at a coupling of $k = 0.08$. In low-coupling range, the equalizer needs to scale heavily, therefore enough bits need to be available in the integer part. The coefficient with the highest value is $c_1$, where the real part reaches a value close to 30 during the frame. Therefore, a integer bitwidth of 6 bit is recommended.

Also for the coefficients fixed-point format, a fine analysis makes sense. In this case, the 6bit for the integer part are not changed, however the fractional bitwidth can be tuned. Figure 5.10 shows the PER for four different fractional bitwidths. From that result we can conclude that decreasing the fractional bitwidth further than 10bit results in an increase of the PER in the high-coupling range. Therefore, the final fixed-point format for the coefficients is Q6.10.

Lastly, Figure 5.11 compares PER when using a mid-rise (non-zero) quantizer compared to a standard quantizer. There is a slight advantage when using the mid-rise quantizer, however not very significant. Nevertheless, it is suggested to use a mid-rise quantizer for quantizing the coefficient update, since it does not cost more area and delivers better convergence properties (see Section 4.2.1), also if, with the particular channel model, the performance does not get significantly better.

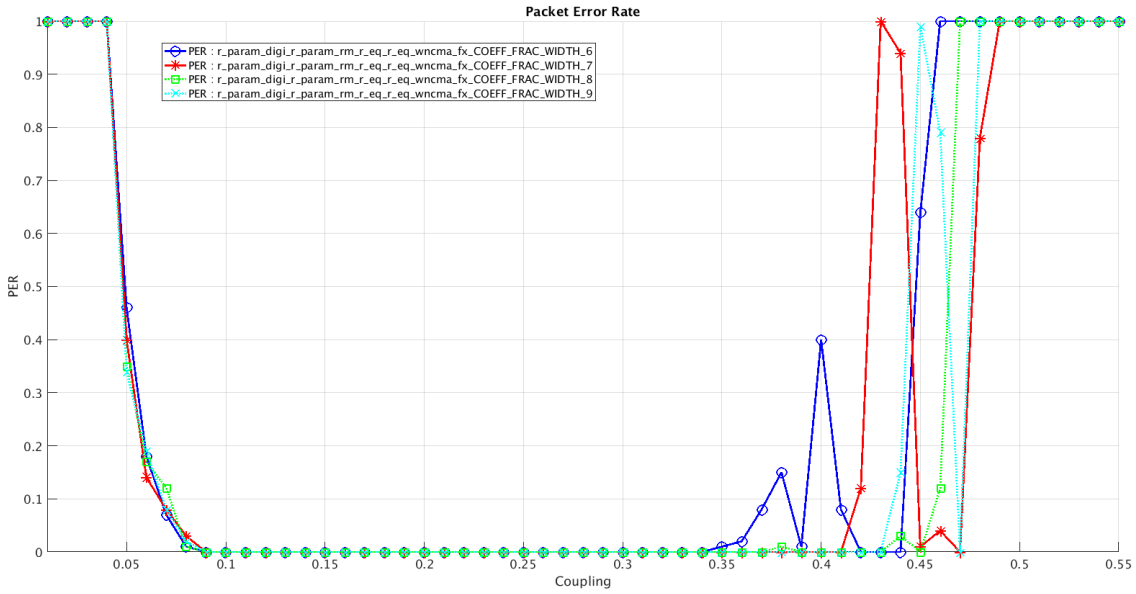



Figure 5.10: Effect of fixed-pointing the backward path (Q3 and C3) on the PER over coupling (fine).

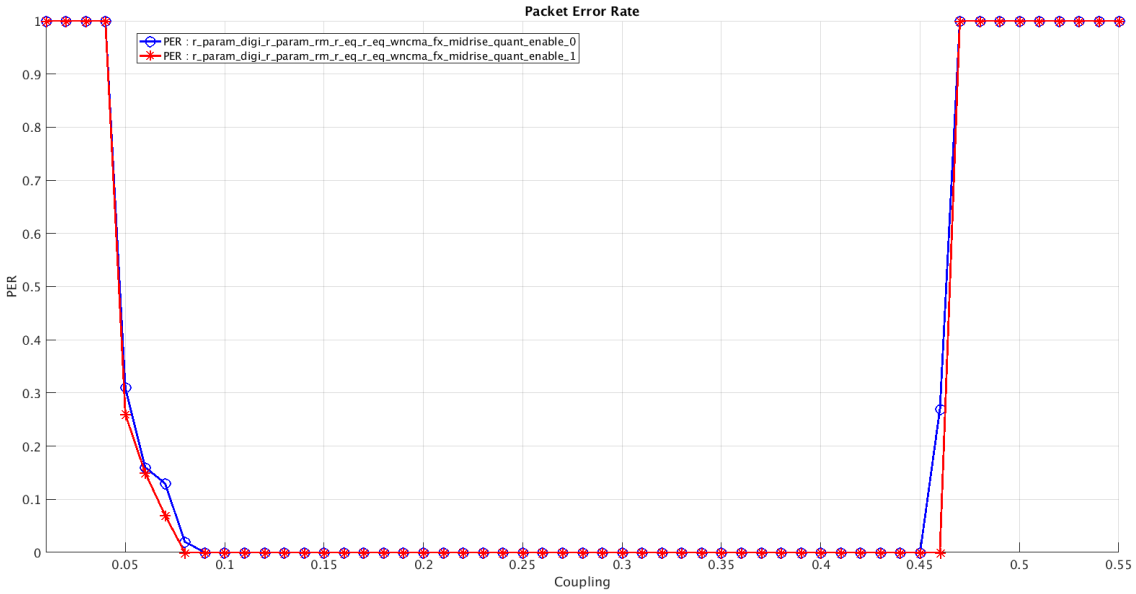



Figure 5.11: Effect of using a mid-rise (non-zero) quantizer for Q3 on the PER.

## 5.2 RTL Simulations

Figure 5.12 shows RTL simulation results of a low-coupling scenario (k=0.08) in SimVision. Because the simulated coupling is low, only the real part of the equalizer input `eq_in_i_float` and equalizer output `eq_out_i_float` are shown. The signal `is_signal_det` originates from an external signal detector, and therefore is not generated by the equalizer itself. The clock `clk` runs at a frequency of 13.56MHz.

Before a signal is detected, the controller state `s_fs_controller` is IDLE. During this state, `eq_in_i_float` is simply forwarded to `eq_out_i_float`. In other words, the equalizer is configured for pass-through. Also, the filter coefficients are kept at their initial setting, which is a 1 for the first coefficient, and 0 for the others. Figure 5.12 shows the real part of each coefficient (`eq_coeff_<n>_real_i_float`). Finally, the filter input and all delayed filter inputs are kept at 0. Therefore, the filter outputs `eq_filter_out_real_o` and `eq_filter_out_imag_o` are 0 as well.

As soon as a signal is detected, the controller transitions from IDLE to SETTLING_ON. In the simulation example shown in Figure 5.12, the settling time is set to 100. During this state, the filter is receiving the equalizer input, the equalizer output is switched to the filter output and the coefficients are updating, even when `is_signal_det` temporarily goes down to 0. After 100 cycles have passed, the controller transitions to the ACTIVE state, because `is_signal_det` is still 1. Otherwise, the controller would transition back to IDLE. The ACTIVE state is similar to SETTLING_ON, just that as soon as `is_signal_det` goes to zero, the controller transitions to SETTLING_OFF. In the simulation example shown in Figure 5.12, this transition from ACTIVE to SETTLING_OFF happens as soon as the frame is over. During SETTLING_OFF, the filter input, as well as the delayed inputs, are set to 0, the equalizer input is forwarded to the output, the coefficients are set to their initial values and the coefficient update is stopped, even when `is_signal_det` temporarily goes to 1. After 100 cycles of settling wait time, the controller transitions back to IDLE.

In the simulation example shown in Figure 5.12, the scaling effect of wNCMA can clearly be seen. The coefficients are scaled up, such that the output signal reaches a certain amplitude, which the wNCMA interprets as $\pm 1$. The potential issues arising from this scaling effect and possibilities for avoiding it are discussed in Section 6.1.

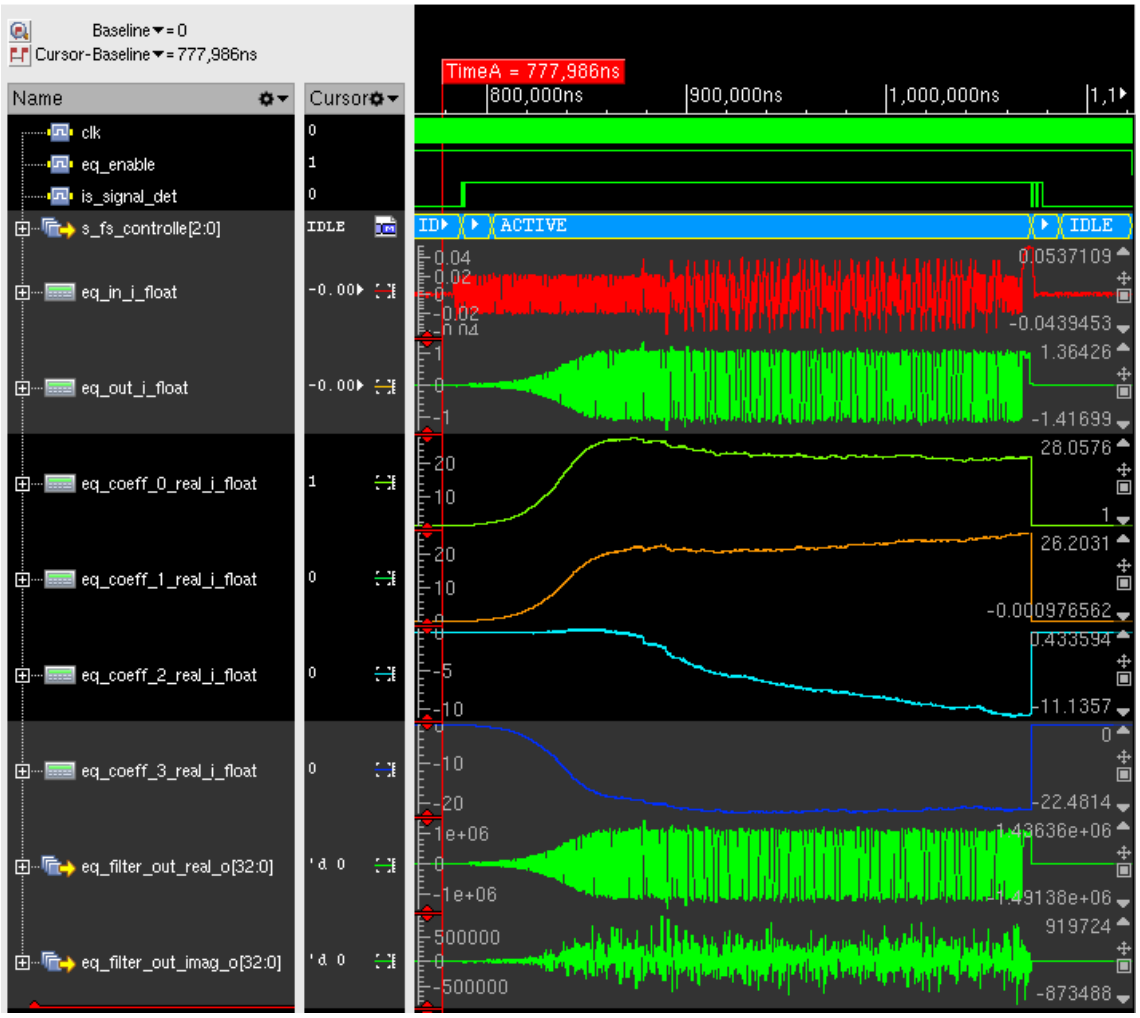



Figure 5.12: RTL simulation results of a low-coupling scenario (k=0.08).

## 5.3 Synthesis Results

Table 5.1 presents the final synthesis results of the equalizer, based on the fixed-pointing formats discussed in Section 5.1.2. Project requirements stated that the equalizer should have at maximum $30k$ gates, which was easily achieved with $22k$ gates. The biggest contributor to the total area is the coefficient update module, which amounts to around $11k$ gates. This block is the biggest contributor because it contains the largest multipliers, which are the correlation multipliers (input signal times scaled error). Since the filter order is $N = 4$, four correlation multipliers are necessary, which already require $4 \times 2111 = 8444$ gates. The FIR filter requires around $6.7k$ gates, significantly less than the coefficient update. Although also the filter needs $N$ complex multipliers, their bitwidths are lower (Q3.10 x Q6.10 for the filter, Q3.10 x Q10.11 for the coefficient update).

A surprising result was that the logarithmic multiplier turned out to consume more area than a standard multiplier (2406 gates compared to 1560 gates of a standard multiplier), as shown in Figure 5.2. Possibly, the logarithmic optimization only pays off for higher bitwidths, due to a weaker area growth when increasing bitwidth. However, this was not confirmed in the course of this thesis. There was also no incentive to do so, since the goal in terms of area consumption has already been reached with standard multipliers.

| Component | Cell Count | Cell Area [$\mu m^2$] | Net Area [$\mu m^2$] | Total Area [$\mu m^2$] |
|---|---|---|---|---|
| Equalizer | 21846 | 23315 | 10739 | 34054 |
| Update | 10849 | 11526 | 5116 | 16642 |
| Update Mult | 2111 | 2312 | 931 | 3243 |
| Update Quant | 106 | 68 | 32 | 100 |
| Update Clip | 22 | 14 | 5 | 19 |
| Filter | 6653 | 7671 | 3211 | 10882 |
| Filter Mult | 1560 | 1749 | 660 | 2409 |
| Filter Quant | 32 | 29 | 9 | 38 |
| Filter Clip | 29 | 18 | 7 | 25 |
| Normalization | 2481 | 2154 | 1028 | 3182 |
| Error | 1646 | 1788 | 713 | 2501 |
| Counter | 56 | 48 | 20 | 68 |
| Controller | 12 | 9 | 3 | 12 |

Table 5.1: Final synthesis results of the equalizer.

| Component | Cell Count | Cell Area [$\mu m^2$] | Net Area [$\mu m^2$] | Total Area [$\mu m^2$] |
|---|---|---|---|---|
| Filter Mitchell Mult | 2406 | 1300 | 968 | 2268 |
| Exp2 | 306 | 126 | 100 | 226 |
| Log2 | 139 | 78 | 53 | 131 |
| Quantizer | 36 | 38 | 9 | 47 |

Table 5.2: Synthesis results of a logarithmic filter multiplier.

# Chapter 6

# Conclusion and Future Works

This thesis suggests an area efficient and robust implementation of the wNCMA, taking into account the effects that occur in adaptive systems when moving from a floating-point to a fixed-point format. An introduction into the fundamental topics of Channel Equalization and NFC can be found in Chapter 1. Chapter 3, after explaining every step of the wNCMA, clearly shows that the algorithm works and can successfully equalize the effects of a simplified NFC channel over a wide coupling range. As discussed in Section 3.7 however, the Type-B frame format does not allow for long convergence times. Also, the SOF, during which the equalizer is supposed to converge, actually looks different than the data part of the frame. Nevertheless, Chapter 5 demonstrates that the equalizer improves PER significantly in the low to mid-coupling range.

I suggest the following improvements for future work.

- Section 3.7 describes the Type-B frame format, which is the only format supported by the equalizer. This format imposes heavy constraints on the convergence time and behaviour of the equalizer. Nevertheless, information about the structure of a frame is known. This information can be used to create a training sequence for faster convergence, or perform an inverse estimation of the channel for initializing the equalizer.

- The fixed-pointing approach applied in this thesis is rudimentary. PER simulations were performed for a few different fixed-point formats of the error and coefficients. There exist far better approaches for fixed-pointing, such as an iterative Signal to Quantization Noise Ratio (SQNR) analysis (how much quantization noise does a certain fixed-point format induce in the system) [9]. In a way, fixed-pointing also represents a discrete optimization-problem. Thus, approaches from the mathematical field of optimization could be applied here as well.

- To further reduce area consumption, the correlation multipliers could be replaced by shifters. The effect of this simplification on the PER needs to be studied.

- The error calculation of the wNCMA can be adjusted such that the scaling of the output signal is avoided, see Section 6.1.
- Timing analysis was not done in the course of this thesis. Because the clock frequency of $f_c = 13.56\,\text{MHz}$ is quite slow, timing analysis was considered less important. Nevertheless, if timing does become an issue in the future, the delayed version of the wNCMA can be implemented, as described in Section 4.1.2. The delayed version should decrease the critical path by roughly a factor of two.

If the equalizer is used together with firmware, more possibilities open up. For example, the equalizer does not always need to start from the initial coefficients $\mathbf{c}_{\text{init}} = [1; \text{zeros}(\text{N} - 1, 1)]$. After the first successful frame, the coefficients can be stored and reused for the next frame, since during card-reader communication, multiple frames are exchanged. The assumption that the coupling does not significantly change during those frames seems reasonable.

The final parameter configuration (mainly filter order and step-size) and fixed-point formats (for the error and coefficients) suggested in Chapter 5 make the equalizer tailor-made to the specific coupling system used in this thesis. For using the equalizer in conjunction with another coupling system, parameter and fixed-point formats need to be evaluated again, to guarantee optimal performance.

## 6.1 Avoid Scaling Effect of the wNCMA

The wNCMA adjusts the filter coefficients over time such that the output signal becomes closer to $\pm 1$. Consequently, if the input is significantly lower in amplitude than $\pm 1$, the coefficients of the filter will be scaled up to increase the amplitude of the output. For a fixed-point format of Q3.10, as used for the input and output of the equalizer, the floating-point value $+1$ becomes $x_{\text{fx}} = \text{round}\,(1.0 \times 2^{10}) = 1024$. The described scaling effect, as shown in Figure 5.1, is an unnecessary and undesired effect. The wNCMA is supposed to mitigate ISI and phase distortion, but not scale the signal.

One way to avoid the scaling effect is by adjusting the error calculation of the wNCMA such that

$$\text{e}[n] = \frac{\text{Re}\{\text{y}[n]\}^3}{\text{E}\{\text{Re}\{\text{x}[n]\}^2\}} + j\,\text{Im}\{\text{y}[n]\}^3 - \text{Re}\{\text{y}[n]\}, \tag{6.1}$$

where $\text{E}\{\text{Re}\{\text{x}[n]\}^2\}$ is the expected value of the real part of the input signal power. In that way, the term $\text{Re}\{\text{y}[n]\}^3$ is scaled up if the real part of the input signal power is low, acting against an otherwise large error term, which would lead to scaling up the coefficients and subsequently the output signal. This change is inspired by the original $W_{\text{new}}$ algorithm [2], which the wNCMA is a derivative from.

Figure 6.1 shows the equalizer input and output when receiving a frame with twelve-bit data at a coupling of 0.08. The error is calculated according to Equation 6.1. Clearly, the equalizer output is not scaled up anymore as it was in Figure 5.1. However, there is some variability in the amplitude of the output, since $\mathrm{E}\{\mathrm{Re}\{\mathrm{x}[n]\}^2\}$ can only be calculated over a limited number of samples. For the simulation results shown in Figure 6.1, a sample vector of length 32 was used.

Out of time constraints, the potential improvement of avoiding the scaling effect of the wNCMA could not be investigated further. It needs to be demonstrated that the wNCMA with modified error has finally a better performance than the original wNCMA, since an implementation of the modified error requires additional logic.

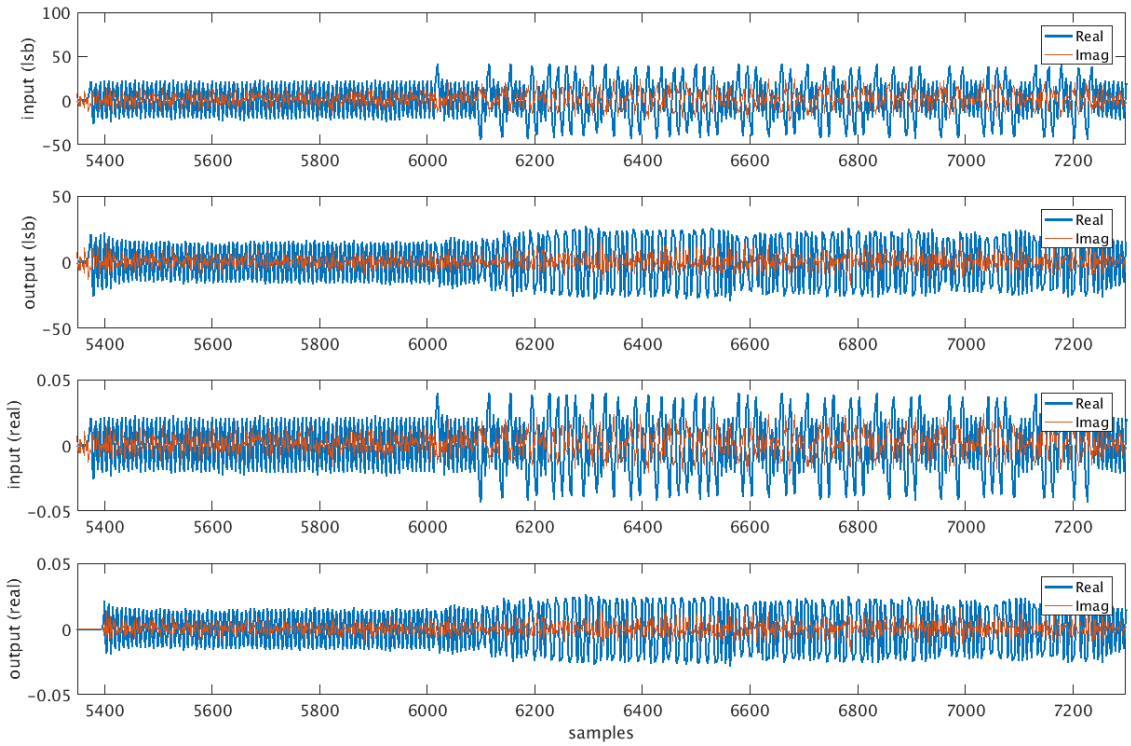



Figure 6.1: Input and output of the equalizer, when receiving a frame with twelve bytes at a coupling of $k = 0.08$ and a data rate of 1.695 Mbit/s. The wNCMA error is calculated according to Equation 6.1.

# Appendix A

# MATLAB Code

### A.0.1 NFC Communication Channel

Table A.1 shows the numerator and denominator polynomial of the coupling model for different coupling values. Note that this coupling model is a simplified baseband model of the coupling system shown in Figure 2.2. By stating the coefficients of the polynomials, the curious reader can easily generate a random BPSK sequence, send this sequence through the coupling model for a number of different coupling values, and apply the wNCMA. The wNCMA itself consists only of three simple equations, as explained in Section 3.5. Therefore, the interested reader can easily reproduce the results presented in Chapter 3.

| **Coupling Factor k** | **Numerator Polynomial B** | **Denominator Polynomial A** |
|---|---|---|
| 0.01 | -0.0000 - 0.0002i<br>-0.0013 + 0.0002i<br>0.0003 + 0.0002i | 1.0000 + 0.0000i<br>-1.1855 - 0.1158i<br>0.2567 + 0.1026i |
| 0.05 | -0.0001 - 0.0057i<br>-0.0279 + 0.0031i<br>0.0067 + 0.0045i | 1.0000 + 0.0000i<br>-1.2176 - 0.1386i<br>0.2918 + 0.1194i |
| 0.1 | -0.0001 - 0.0192i<br>-0.0840 + 0.0059i<br>0.0322 + 0.0142i | 1.0000 + 0.0000i<br>-1.3457 - 0.1577i<br>0.4215 + 0.1261i |
| 0.15 | -0.0002 - 0.0328i<br>-0.1260 + 0.0009i<br>0.0631 + 0.0154i | 1.0000 + 0.0000i<br>-1.4309 - 0.1006i<br>0.5156 + 0.0682i |
| 0.2 | -0.0004 - 0.0435i<br>-0.1452 - 0.0126i<br>0.0818 + 0.0107i | 1.0000 + 0.0000i<br>-1.3867 - 0.0389i<br>0.5102 + 0.0130i |

| | | |
|---|---|---|
| 0.25 | 0.0003 - 0.0510i<br>-0.1493 - 0.0327i<br>0.0866 + 0.0040i | 1.0000 + 0.0000i<br>-1.2752 + 0.0058i<br>0.4692 - 0.0235i |
| 0.3 | 0.0009 - 0.0556i<br>-0.1436 - 0.0575i<br>0.0820 - 0.0039i | 1.0000 + 0.0000i<br>-1.1092 + 0.0269i<br>0.4055 - 0.0374i |
| 0.35 | 0.0004 - 0.0592i<br>-0.1302 - 0.0848i<br>0.0732 - 0.0123i | 1.0000 + 0.0000i<br>-0.8851 + 0.0362i<br>0.3256 - 0.0423i |
| 0.4 | 0.0004 - 0.0620i<br>-0.1110 - 0.1118i<br>0.0621 - 0.0217i | 1.0000 + 0.0000i<br>-0.5999 + 0.0348i<br>0.2392 - 0.0475i |
| 0.45 | 0.0008 - 0.0635i<br>-0.0871 - 0.1390i<br>0.0441 - 0.0387i | 1.0000 + 0.0000i<br>-0.2430 + 0.0226i<br>0.1568 - 0.0633i |
| 0.5 | 0.0006 - 0.0647i<br>-0.0506 - 0.1470i<br>0.0648 - 0.0383i | 1.0000 + 0.0000i<br>-0.0468 + 0.1415i<br>0.1620 - 0.0357i |
| 0.55 | 0.0007 - 0.0654i<br>-0.0082 - 0.1207i<br>0.1004 + 0.0148i | 1.0000 + 0.0000i<br>-0.1869 + 0.3163i<br>0.1259 + 0.1310i |

Table A.1: Discrete-time transfer function coefficients, where B is the numerator polynomial and A the denominator polynomial. The coefficients are given for multiple coupling values, ranging from 0.01 to 0.55.

# Appendix B

# Abbreviations

**ASIC** Application Specific Integrated Circuit. 31, 70

**BER** Bit Error Rate. 15, 36

**BPSK** Binary Phase Shift Keying. 9, 10, 23–25, 36–38, 45

**DD** Decision Device. 15–17

**DSP** Digital Signal Processing. 55, 61

**DUT** Device under Test. 62

**EDS** Euclidean Direction Search. 26

**EOF** End of Frame. 46

**FIR** Finite Impulse Response. 10, 31, 32, 36, 49, 58, 77

**FPGA** Field Programmable Gate Array. 27

**FS** Frame Synchronizer. 32–34, 62

**FSM** Finite State Machine. 11, 32–35, 44, 45

**HLS** High-Level Synthesis. 20, 61

**ISI** Intersymbol Interference. 9, 15, 18, 19, 23, 26, 36–38, 41, 80

**LMS** Least Mean Square. 27, 28

**LS** Least-Squares. 9, 26

**LSB** Least Significant Bit. 46, 58

**MBER** Minimum Bit Error Rate. 16

**MMSE** Minimum Mean-Square Error. 16, 26

**MUX** Multiplexer. 31, 32, 34

**NFC** Near Field Communication. 9, 14, 15, 19–23, 26, 36, 37, 45, 79

**PCD** Proximity Coupling Device. 46

**PER** Packet Error Rate. 12, 31, 36, 68–75, 79

**PICC** Proximity Card or Object. 46

**PLM** Passive Load Modulation. 21, 22, 26

**RF** Radio Frequency. 23

**RFID** Radio Frequency Identification. 21, 26

**RTL** Register Transfer Level. 20, 31, 61, 62

**SOF** Start of Frame. 46, 79

**SQNR** Signal to Quantization Noise Ratio. 79

**wNCMA** well-behaved Normalized Constant Modulus Algorithm. 10–13, 15, 17, 20, 26, 31, 33, 34, 36, 37, 41, 48, 51, 53–55, 59, 61, 76, 79–82

**ZF** Zero Forcing. 16, 26

# Bibliography


[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[2] M. Pinchas and B. Z. Bobrovsky, "A Maximum Entropy Approach for Blind Deconvolution," *Signal Processing*, vol. 86, no. 10, pp. 2913 – 2931, 2006. Special Section: Fractional Calculus Applications in Signals and Systems.

[3] John G. Proakis and Masoud Salehi, *Digital Communications 5th Edition*. McGraw-Hill Higher Education, 2007.

[4] K. Finkenzeller, *RFID Handbook - Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication - Third Edition*. Wiley, 2010.

[5] T. Raspel, "Channel Equalization for 13.56 MHz RFID Systems," Master's thesis, TUGraz, 2018.

[6] "ISO/IEC 14443-3:2011 Identification cards — Contactless integrated circuit cards — Proximity cards Part 3: Initialization and anticollision," standard, International Organization for Standardization, Geneva, CH, Apr. 2011.

[7] "ISO/IEC 14443-2:2016 Identification cards — Contactless integrated circuit cards — Proximity cards — Part 2: Radio frequency power and signal interface," standard, International Organization for Standardization, Geneva, CH, June 2016.

[8] J. N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, pp. 512–517, Aug 1962.

[9] Shoab Ahmed Khan, *Digital Design of Signal Processing Systems: A Practical Approach*. John Wiley and Sons, Ltd, 2011.

[10] Simon O. Haykin, *Adaptive Filter Theory*. Pearson, 1996.

[11] Alexander M. Wyglinski, Maziar Nekovee, and Y. Thomas Hou, *Cognitive Radio Communications and Networks*. Academic Press, 2010.

[12] Narasimhan Balachander, Chien Charles, Zhou Qiang, Lin Chih-yuan, Zhan Cheng-chou, and Peng Bao-chi, "Electronic Device with Equalization, Integrated Circuit and Methods Therefor," 12 2012. U.S. Patent US9768984B2.

[13] Van De Beek Remco Cornelis Herman and Ciacci Massimo, "Adaptive Equalizer And/Or Antenna Tuning," 8 2012. European Patent EP2582069B1.

[14] M. Lunglmayr and M. Huemer, "Least Squares Equalization for RFID," in *2010 Second International Workshop on Near Field Communication*, pp. 90–94, April 2010.

[15] K. Rocha, T. Bose, and M. Larsen, "A Multiplier-Free Adaptive Algorithm for Channel Equalization," in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, vol. 1, pp. 82–86 vol.1, Nov 2002.

[16] S. Dasgupta, C. R. Johnson, and A. M. Baksho, "Sign-Sign LMS Convergence with Independent Stochastic Inputs," *IEEE Transactions on Information Theory*, vol. 36, pp. 197–201, Jan 1990.

[17] Doaa Ashmawy, Esam Abdel-Raheem, Hala Mansour, Mohamed Youssif, and Mahmoud Mohanna, "FPGA Implementation of Blind Adaptive Decision Feedback Equalizer," in *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 495–500, Dec 2009.

[18] D. Duttweiler, "Adaptive Filter Performance with Nonlinearities in the Correlation Multiplier," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, pp. 578–586, August 1982.

[19] J. Cioffi, "Limited-Precision Effects in Adaptive Filtering," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 821–833, July 1987.

[20] Cambridge Dictionary, "Overview - Creativity." https://dictionary.cambridge.org/de/worterbuch/englisch/creativity.

[21] A. Wenzler and E. Luder, "New Structures for Complex Multipliers and Their Noise Analysis," in *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, vol. 2, pp. 1432–1435 vol.2, April 1995.

[22] D. J. Mclaren, "Improved Mitchell-based Logarithmic Multiplier for Low-Power DSP Applications," in *IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings.*, pp. 53–56, Sep. 2003.

[23] MathWorks, "Fixed Point Designer." https://uk.mathworks.com/help/fixedpoint/index.html, 2019. Accessed on 2019-12-05.

[24] MathWorks, "High Level Synthesis." https://uk.mathworks.com/discovery/high-level-synthesis.html, 2019. Accessed on 2019-12-07.

[25] MathWorks, "Getting Started with MATLAB to HDL Workflow." https://uk.mathworks.com/help/hdlcoder/examples/getting-started-with-matlab-to-hdl-workflow.html, 2019. Accessed on 2019-12-07.