Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Tim Bröhan

FPGA Based Development of an Orthogonal Frequency Division Multiplexing System for Automotive Communication Networks

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Tim Bröhan

FPGA Based Development of an Orthogonal Frequency Division Multiplexing System for Automotive Communication Networks

Masterthesis based on the examination and study regulations for the
Master of Science degree programme
Information and Communication Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner : Prof. Dr.-Ing. Lutz Leutelt
Second examiner : Prof. Dr.-Ing. Jürgen Vollmer

Day of delivery June 13$^{\text{th}}$ 2016

Tim Bröhan

Title of the paper

FPGA Based Development of an Orthogonal Frequency Division Multiplexing System for Automotive Communication Networks

Keywords

FPGA, SoC, OFDM, DMT, Guard Interval, Equalizer, BER, System Generator

Abstract

This thesis describes the conception, implementation and verification of an baseband controller for an Orthogonal Frequency Division Multiplexing system. The design is implemented on an FPGA and shall be used in automotive data communication networks. This includes the description of the implementation with System Generator and the synthesis for the FPGA.

Tim Bröhan

Thema der Masterthesis

FPGA basierte Entwicklung eines Orthogonal Frequency Division Multiplexing Systems für Kommunikationsnetzwerke im Automobil

Stichworte

FPGA, SoC, OFDM, DMT, Guard Interval, Entzerrer, BER, System Generator

Kurzzusammenfassung

Diese Thesis beschreibt die Konzeption, Implementierung und Verifikation eines Basisband Controllers für ein Orthogonal Frequency Division Multiplexing System. Das Design wird auf einem FPGA implementiert und soll in Datenkommunikationsnetzwerken im Automobil angewendet werden. Hierbei wird die Implementierung mit System Generator beschrieben und die Synthese auf dem FPGA erläutert.

Contents

List of Tables

List of Figures

1. Introduction

Todays car drivers expect a lot of safety features like Anti-lock Breaking System (ABS) or Electronic Stability Program (ESP). Furthermore, cars have many entertainment and comfort systems like audio systems with speakers distributed all over the car or electrically adjustable seats. The ideas for additional features are never stopping, due to that the cabling effort is increasing continuously.

The car manufacturers handled this problem with the introduction of bus systems to reduce cabling, starting with the Controller Area Network (CAN) bus in the Mercedes-Benz 500E in 1991 [Rei10, S.120]. This bus is widely used in different variants with different data rates from motor management with Highspeed-CAN (CAN-C) supporting data rates up to 1 MBit/s, to electric window openers using Lowspeed-CAN (CAN-B) supporting data rates up to 125 kBit/s[Rei10, S.134]. Since 1991 many other bus systems got invented. One example is the lightweight Local Interconnect Network (LIN) in 2001 with data rates up to 20 kBit/s which was developed to support the CAN bus when even the CAN-B is oversized. LIN is used for systems with small spatial expansion, like all systems in a door, and is connected to the CAN bus.

In 1999 the FlexRay Consortium was founded to develop a new bus system to transmit safety relevant data with a guaranteed maximum transmission time. This is achieved using a timeslot system where every participant (node) has a fixed time slot to send data. FlexRay allows data rates upto 10 MBit/s and uses two redundant wires to make the transmission less error prone[Rei10, S.188]. There are projects which examine the possibility of using FlexRay to substitute the hydraulic power transmission of the breaking systems called Break-By-Wire. With the integration of the breaks in the bus system of the car this would allow to use the data from the breaks to improve the performance of other systems like ESP[Vec].

This thesis is part of the ”X-by-wire(less)” project at the Hamburg University of Applied Sciences (HAW) which tries to go even a step further and wants to create a wireless, bus-like system which has the deterministic message delivery time of FlexRay and an even higher data rate of up to 50 MBit/s. This would allow to reduce the cabling in cars even further and save weight resulting in less fuel consumption while making the cable harnesses less complex. The transmission rate will be high enough to handle the increasing data rate demand of future applications. The project is part of the Urban Mobility Lab, which houses several projects with the aim to develop integrated, user-oriented solutions for sustainable, convenient and cost effective mobility in large cities[HAW].

1.1. Task Description

Content of this thesis is the conception, implementation and verification of an Orthogonal Frequency Division Multiplexing (OFDM) system. This includes the identification and specification of all necessary system parameters. Values which can not be defined in this stage of the project shall be assumed to realistic values.
The system is supposed to be implemented using the Xilinx System Generator toolbox for MATLAB Simulink, this allows the use of the Simulink simulation environment for testing. The implementation shall allow an easy changing of the system parameters, which allows experimenting and an easy adjustment to the parameter settings used in the final version in a car. The created design shall be implemented on an Xilinx Zynq-7020 System on Chip (SoC) which is part of a pre-created development platform. The System boundaries are explained in the following list:

- **Transmitter input**: The digital input pin of the Field Programmable Gate Array (FPGA) for the data bits which shall be transmitted.
- **Transmitter output**: The digital output pins of the FPGA which are connected to the analogue front-end hardware with the Digital Analog Converter (DAC).
- **Receiver input**: The digital digital input pins of the FPGA which are connected to the analogue front-end hardware with the Analog Digital Converter (ADC).

- **Receiver output**: The digital output pin of the FPGA which outputs the received data bits.

Therefore, no hardware creation aside of the FPGA is part of this thesis, but the system parameters shall be chosen in a way that the resulting system allows the creation of the analogue front-end, refer to Figure 1.1.

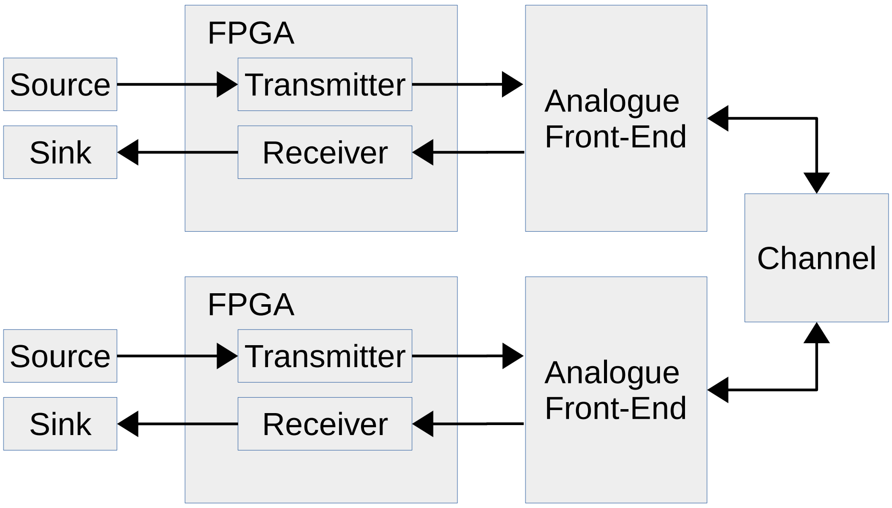


Figure 1.1.: Block diagram for usage in a car with two nodes.

Part of the pre-created hardware platform is the possibility to connect an oscilloscope through SMB connectors. This allows to verify the FPGA implementation in a physical measurement with an oscilloscope through the comparison of the transmitted and received bit stream.
The analogue front-end to connect a physical channel is part of a future thesis, hence, no measurements with a physical channel are possible. Therefore, a Additive White Gaussian Noise (AWGN) channel model shall be implemented in Simulink to simulate a transmission and compare the Bit Error Ratio (BER) to the theoretical value. To improve the system performance under real channel conditions, an equalizer shall be included. To verify the function of the equalizer, a channel model based on an Finite Impulse Response (FIR)-Filter shall be implemented, which allows to specify a given channel impulse response as filter coefficients. The final version used in the car has to transmit data organized in frames generated by a higher layer protocol engine, however, the design shall handle the incoming bits as a stream of random bits. A data source shall be implemented on the FPGA which generates random bits and

allows to verify the design on the hardware. For testing purposes, the design shall be implemented on a single FPGA which runs the transmitter and receiver, see Figure 1.2.

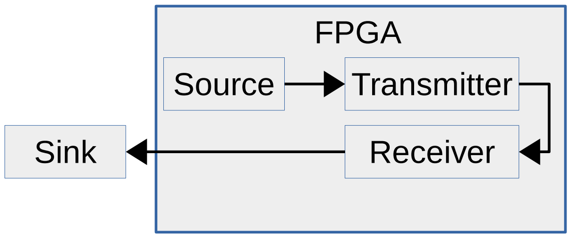


Figure 1.2.: Block diagram for the test environment.

An additional requirement is the possibility to activate a mode which generates a real-valued output signal at the ports connected to the analogue front-end. This allows to use the design with an alternative analogue front-end for a wired channel without using a quadrature modulator which creates the real-valued signal. This is useful to experiment with the easier channel behaviour of a wired channel.

In the context of the "X-by-wire(less)" project, the result of this thesis will be used as a baseband controller to prepare the data for the analogue front-end.

1.2. Chapter Overview

After an introduction in this chapter, the following Chapter 2 explains the fundamentals of an OFDM system.

Chapter 3 discussed how the system parameters have to be set to receive a working OFDM system and meet the requirements. It is discussed which BER is expected with the setting of the parameters and how it influences the selection of the ADC/ DAC hardware. Furthermore, the pre-constructed hardware platform is introduced.

Chapter 4 describes the implementation of the model with the System Generator blocks and the necessary decisions to achieve a synthesizable design. It is explained how the model can be configured.

Chapter 5 describes the synthesis and implementation of the System Generator model on the FPGA. It discusses the occurring timing issues and how they are handled. Furthermore, the final synthesis and implementation result are examined and the hardware resource utilization for several configuration modi is compared.

Chapter 6 describes how the function of the design is verified in Simulink and on the hardware.

Chapter 7 summarizes this thesis and lists several possibilities to improve the system and gives ideas for future thesises.

2. Fundamentals

This chapter introduces the fundamental theory of an OFDM system. In addition to the theory of the implemented components, a broader view of possible features of an OFDM system is given. This shall help to understand the concepts in general and give a starting point for a future continuation of the thesis.

2.1. Basics of OFDM

In an OFDM system [Kam11, S.582-587], the overall bandwidth B is divided into several subbands called subcarriers. This allows to assume that the channel frequency response is constant within a single subcarrier if the subcarrier spacing is small enough. Furthermore, the bits which shall be transmitted are distributed on the subcarriers, this increases the OFDM symbol duration T_{sym} in comparison to the symbol duration of a single carrier system, because each subcarrier is processed at a slower rate. The increased symbol duration results in less Inter Symbol Interference (ISI), because the impulse response of the channel can be longer without expanding over multiple symbols [1].

The bandwidth for a single subcarrier is the inverse of the OFDM symbol duration, see Equation 2.1.

$$B_{sc} = 1/T_{sym} \tag{2.1}$$

The symbol duration can be calculated with the subcarrier count N_{sc}, the size of the modulation alphabet M, and the bit period T_{bit} according to Equation 2.2. The term $ld()$ is equal to $log_2()$.

$$T_{sym} = N_{sc} \cdot ld(M) \cdot T_{bit} \tag{2.2}$$

[1] A detailed explanation of ISI can be found in [Kam11, S.234ff.]

The bandwidth is evenly distributed between all subcarriers, therefore, the total bandwidth of the transmission is $B = N_{sc} \cdot B_{sc}$.

Derivation of the Multicarrier Signal

With the symbol mapped to the n-th subcarrier, for the i-th OFDM symbol called $d_n(i)$, while the transmit filter is called $g_s(t-iT_{sym})$, and with the subcarrier frequencys at $f_n = n/T_{sym}$ for $n = 0, ..., N_{sc} - 1$, the general multicarrier signal is achieved in Equation 2.3 .

$$s_{MC}(t) = T_{sym} \sum_{n=0}^{N_{sc}-1} \sum_{i=-\infty}^{\infty} d_n(i) g_s(t - iT_{sym}) e^{j2\pi f_n t} \tag{2.3}$$

If the impulse responses of the transmit filters are set to a causal rectangular

$$g_s(t) = \begin{cases} 1/T_{sym} & \text{for} \quad 0 \leq t < T_{sym} \\ 0 & \text{otherwise} \end{cases}$$

the multicarrier signal is simplified to Equation 2.4. The equation describes the signal for the i-th OFDM symbol independently and allows to set the transmit filter $g_s(t - iT_{sym})$ to $1/T_{sym}$, which removes the factor at the beginning of the equation.

$$s_{MC\_simp}(t) = \sum_{n=0}^{N_{sc}-1} d_n(i) e^{j2\pi f_n t} \ \text{ for } \ iT_{sym} \leq t < (i+1)T_{sym}) \tag{2.4}$$

Because it shall be derived a time discrete model, the sampling frequency is set to $f_s = 1/T_s = N_{sc}/T_{sym}$. Applied to the above equation, the resulting formula is 2.5.

$$s_{MC\_disc}(k'Ts) = \sum_{n=0}^{N_{sc}-1} d_n(i) e^{j2\pi nk'/N_{sc}} \ \text{ for } \ iN_{sc} \leq k' \leq (i+1)N_{sc} - 1) \tag{2.5}$$

The index k' iterates through all N_{sc} samples of one OFDM symbol while the sampling period T_s is the time interval between the samples. The equation is valid for each N_{sc} values of one OFDM symbol.

The term represents the Inverse Discrete Fourier Transformation (IDFT) operation, only the scaling factor $1/N_{sc}$ is missing. This allows to further simplify the term to Equation 2.6.

$$s_{MC\_disc}(k'Ts) \equiv s(i,k) = N_{sc} \cdot IDFT^{n}_{N_{sc}}\{d_n(i)\} \quad \text{for} \quad \begin{array}{l} k = k' - iN_{sc} \\ k \geq 0 \\ n \leq N_{sc} - 1 \end{array} \tag{2.6}$$

The index k normalizes the index k', in such a way that it is between 0 and $N_{sc} - 1$ for each OFDM symbol. Likewise, index n at the IDFT operation shall clarify that the symbols of each subcarrier $d_n(i)$ are involved in the N_{sc}-IDFT output values within a OFDM symbol. Equation 2.6 shows that OFDM can be easily and effectively implemented using the Inverse Fast Fourier Transformation (IFFT).

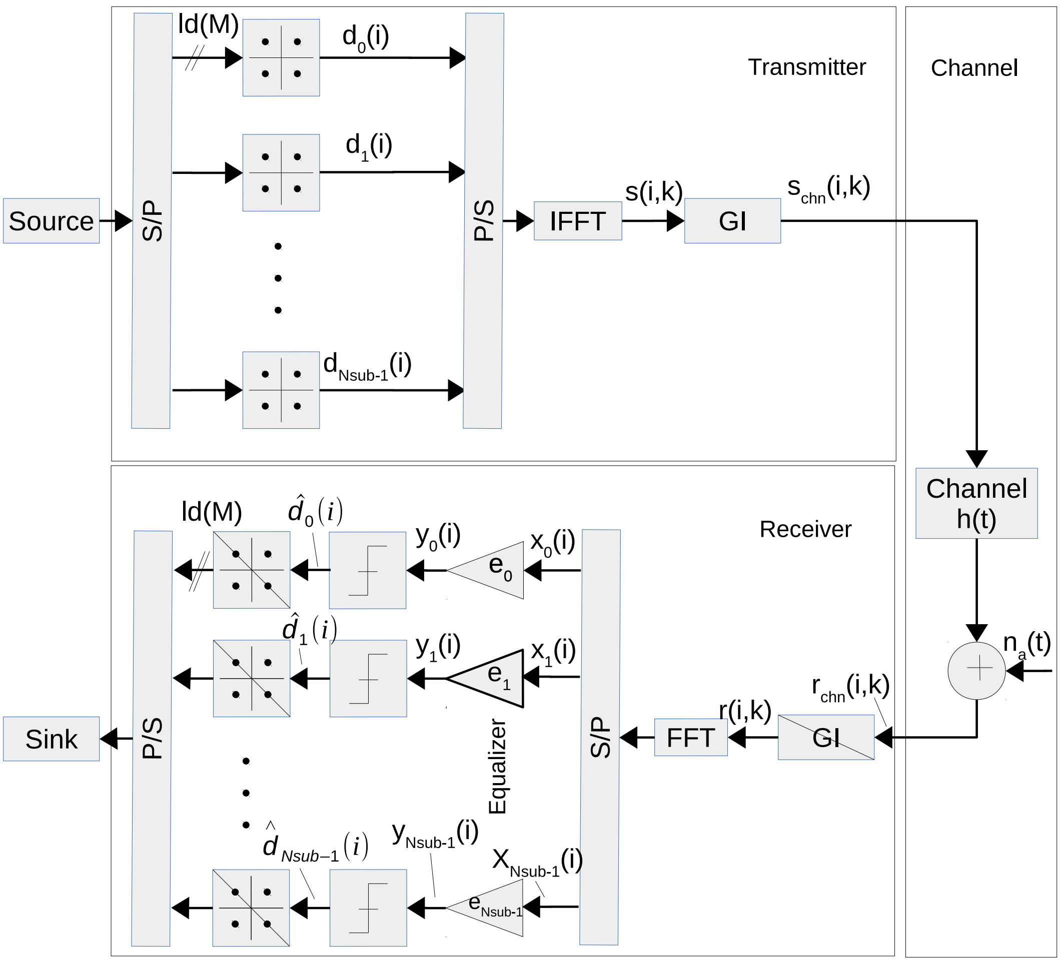


Figure 2.1.: Block diagram of the system [Kam11, S.587]. The crossed blocks represent the inverse operation.

Figure 2.1 gives an overview of the system. The channel, the block Guard Interval (GI), the modulators and the equalizer will be discussed in a later section.

Orthogonality

Figure 2.2 shows an example OFDM spectrum, all other subcarriers are zero in the maximum of a specific subcarrier which shows the orthogonality.

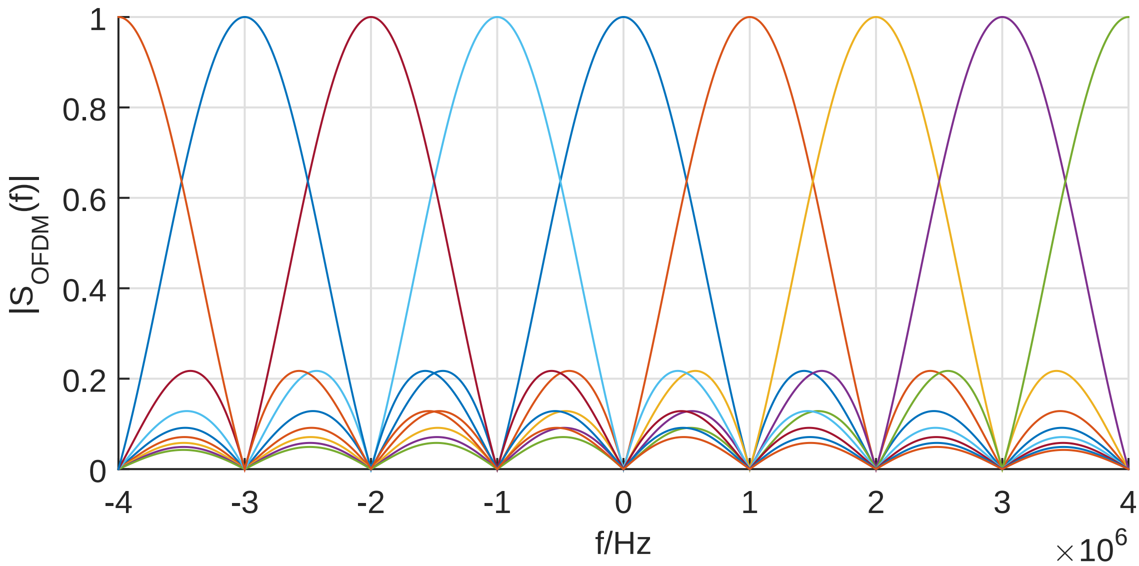


Figure 2.2.: OFDM spectrum for $N_{sc} = 8, B_{sc} = 1\,\text{MHz}$.

A proof for the orthogonality of the multicarrier signal $s(i, k)$ is shown in [Kam11, S.586].

Because of the orthogonality:

- There is no Inter Carrier Interference (ICI) if the subcarriers are sampled in the maxima, therefore, all other subcarriers are 0 in the maximum of a specific subcarrier. This means the signal fulfils the first Nyquist criteria in the frequency domain. [Kam11, S.586]

- The data can be restored by a matched-filter-receiver [Höh13, S.431]. The IFFT in the transmitter is responsible for the transmit filter impulse response, therefore, the Fast Fourier Transformation (FFT) in the receiver fulfils the matched-filter condition.

Historical Context

The first ideas for a multicarrier transmission are in the paper by Mosier and Clabaugh from 1958 [MC58]. The first paper which uses the Discrete Fourier Transformation (DFT) for a multicarrier transmission, in the form used today, was published in 1971 by Weinstein and Ebert [WE71]. This was very effective [Kam11, S.582] in combination with the invention of the FFT by Cooley and Tukey in 1965 [JWC65].

2.2. OFDM and DMT

Until now, the OFDM signal discussed in Equation 2.6 is complex-valued. For a transmission over a real channel, a real-valued signal is required. In a radio link this is automatically achieved when a quadrature modulator is used to raise the baseband signal to the desired frequency range. If OFDM shall be used for a wired connection, a quadrature modulator could be used nonetheless, to raise the spectrum out of the baseband to a bandpass signal just over the Direct Current (DC) range.

With a close look at the necessary conditions for a real-valued signal,

$$x(t) \in \mathbb{R} \;\; \text{for} \;\; X(f) = X^*(-f)$$

while $X^*(f)$ describes the complex conjugate of $X(f)$. The following conditions can be derived [Pro03, S.741] with $N_{DMT} = 2 \cdot N_{sc}$:

$$X_0, X_{N_{DMT}/2} \in \mathbb{R}$$

$$X_n = X^*_{N_{DMT}-n} \;\; \text{for} \;\; n = 1, ..., N_{DMT}/2 - 1$$

If the conjugate complex value of each symbol is inserted in the IFFT additionaly, a real-valued output signal is achieved. The disadvantage is the doubling of the IFFT length, Figure 2.3 shows this variant. This second variant is called Discrete Multi Tone (DMT)[2].

[2]The definition of DMT is not consistent in the literature. This thesis uses the variant from [Höh13, S.439].

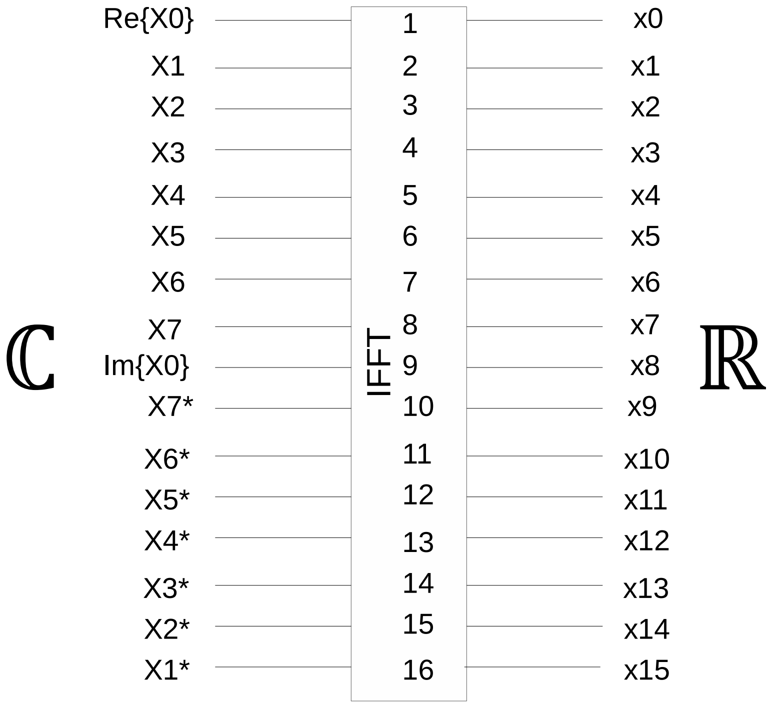


Figure 2.3.: The symbol insertion with 8 subcarriers for DMT. Notice that the IFFT length is doubled.

2.3. Guard interval

Under the influence of an frequency selective channel the orthogonality is broken, because the OFDM symbols need a certain amount of time to reach a steady state which causes ICI. Furthermore, the symbols need a certain amount of time to settle again after a symbol inducing ISI. This problems can be handled using a GI [Kam11, S.588-593].

The GI extends the multicarrier signal $s(i,k)$ with N_{sc} samples by N_{GI} additional samples. The resulting signal on the channel is $N_{chn} = N_{sc} + N_{GI}$ samples long. For the signal $s(0), s(1), ..., s(N_{sc}-1)$ the GI is $s(N_{sc}-N_{GI}), s(N_{sc}-N_{GI}+1), ..., s(N_{sc}-1)$. These additional samples will be inserted in front of the original signal[Pro08, S.751]. The final signal with GI shows Equation 2.7.

$$\underbrace{s(N_{sc}-N_{GI}), ..., s(N_{sc}-1)}_{GI}, \underbrace{s(0), ..., s(N_{sc}-1)}_{signal} \tag{2.7}$$

With GI, the duration of the OFDM symbol on the channel is extended to $T_{chn} = T_{sym} + T_{GI}$. Although, the subcarrier spacing is still like in Equation 2.1.

The bandwidth of the single subcarriers is getting smaller from $\frac{1}{T_{sym}}$ to $\frac{1}{T_{sym}+T_{GI}}$, therefore, the other subcarriers are not zero anymore in the sampling instant of a given subcarrier. This breaks the orthogonality, but with the removal of the GI, in the receiver, the orthogonality is restored.

ISI and ICI using a GI

To prevent ISI and ICI the time T_{GI} has to be longer then the significant parts of the channel impulse response τ_{max}, this will be specified in Section 3.2. Otherwise, the latest OFDM symbol will not settle until the next symbol starts and the symbol will not reach a steady state before the evaluation of the symbol starts, respectively.

If the transmit filter and the receive filter are rectangular functions, the matched filter property is fulfilled. The convolution of these is the subchannel impulse response and in shape of a triangle. In the peak of the triangle is the perfect sample instant which is ISI and ICI free. If a GI is used the rectangular of the transmit filter is longer then the receive filter. This results in a plateau in the peak of the triangle which is a longer interval for sampling where the transmission is ISI and ICI free [Pro03, S.739].

Bandwidth efficiency

During the guard interval no information is transmitted, therefore, the bandwidth efficiency β is decreased, see Equation 2.8. Hence, the GI should be as small as possible.

$$\beta = \frac{1}{1 + T_{GI}/T_{Sym}} \tag{2.8}$$

Normally, the bandwidth efficiency describes the transmitted bits per second per hertz, in this thesis the definition in Equation 2.8 is used which is adopted from [Kam11, S.590].

2.4. Channel Model and Estimation

The channel model to measure the BER is described by the AWGN $n_a(t)$ in Figure 2.1. To test the equalizer, an additional model with the channel impulse response $h(k)$ is implemented. The length of the significant parts of the impulse response is called τ_{max}. The channel is not frequency selective if the the coherence bandwidth $b_c = 1/\tau_{max}$ [Kam11, S.84-92] is higher than the subcarrier width. Under this condition the channel is considered *flat*. The subcarrier spacing in OFDM should be small enough, so that the channel is flat within each single subcarrier.

In a transmission with blocks of L OFDM symbols while the channel is not changing rapidly, therefore, it can be assumed that the channel stays the same while transmitting a single block, the channel can be estimated once at the beginning of a block. The first P symbols are set to a known *pilot* symbol. Due to the changes of the known pilot symbols influenced by the channel, the frequency response $H(n)$ can be estimated. An average over multiple symbols should be used to reduce noise influence [Kam11, S.603-604]. Figure 2.4 shows an example with $N_{sc} = 8, L = 8, P = 2$. The black dots describe the pilot symbols and the white dots the payload symbols.

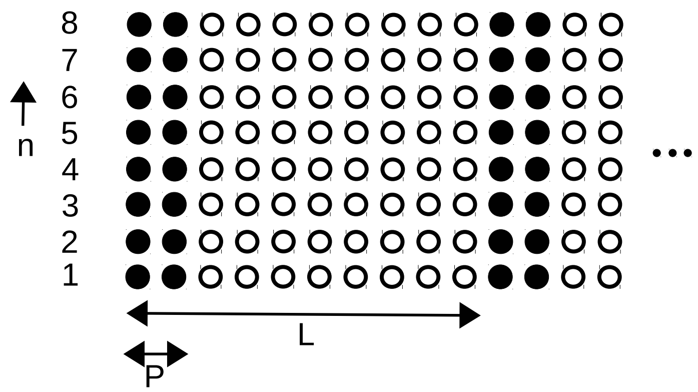


Figure 2.4.: The use of pilot symbols at the beginning of a transmission block.

2.5. Equalization

If a cyclic guard interval is used, which is suppressed at the receiver, the convolution of the transmit signal with the channel impulse response can be described as an circular convolution [Kam11, S.593-594], see Equation 2.9. This represents the overlap-save-method [Cha08, S.299-302] from digital signal processing. A necessary condition is that the GI is longer than the channel impulse response τ_{max}. The symbols in the equations can be found in Figure 2.1 on page 16.

$$r(i,k) = s_{chn}(i,k) *_{circ}^{(k)} h(k) \tag{2.9}$$

With the convolution theorem [Pap11, S.330] this can be written as in Equation 2.10

$$\underbrace{DFT_{N_{sc}}^{(k)}\left\{s_{chn}(i,k) *_{circ}^{(k)} h(k)\right\}}_{x_n(i)} = \underbrace{DFT_{N_{sc}}^{(k)}\left\{s_{chn}(i,k)\right\}}_{d_n(i)} \cdot \underbrace{DFT_{N_{sc}}^{(k)}\left\{h(k)\right\}}_{H(n)} \tag{2.10}$$

The equalization is reduced to a simple point wise division of the received signal $x_n(i)$ with the sampled channel frequency response $H(n)$. The received signal is then $y_n(i)$, see Equation 2.11.

$$y_n(i) = \frac{x_n(i)}{H(n)} \tag{2.11}$$

This one-tap equalizer works only if the channel can be considered flat and represents the zero-forcing solution. Because this is done for every subchannel separately the Minimum Mean Square Error (MMSE) solution isn't better [Kam11, S.593-594], since there is no noise amplification like in single carrier transmissions with zero-forcing equalizers. This is possible, because the channel is flat and there are no points where the frequency response matches zero. Nonetheless, it can improve the performance of the equalizer if the noise is considered while determining the coefficients. One possibility is described in [Pro03, S.745].
If a channel impulse response is too long, it can be shortened with a pre-equalizer, which is examined in [Kam11, S.596].

2.6. Peak to Average Power Ratio

A problem in OFDM is the high Peak to Average Power Ratio (PAPR), which results from the significant variation in the amplitude distribution of the OFDM symbols induced from the superposition of the subcarriers. The amplitudes in an OFDM system are approximately Gaussian distributed.

Nonlinearities in the amplifier have a big impact. If the amplifier is in saturation [Shi10, S.13,S.16], the orthogonality is broken.

The use of methods of digital signal processing [Kam11, S.617-620] to reduce the PAPR is normally better then to oversize the power amplifier and use it in a non-optimal operating point.

The methods can be distinguished between:

- Methods which are applied after the IFFT, these increase the distortion and can result in bit errors.
- Methods which change the coding or modulation process to prevent the peaks before creation, these have a higher computational effort but cause no distortion.

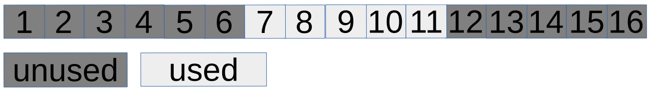
Several techniques are compared in [Far98]. The recommended technique for most applications is the *Pulse Superposition* [Far98, S.78] which is a good compromise between calculation effort and easy implementation. In the paper [FLPS02] these technique is further described. The additional computational effort is small, but additional subcarriers are needed and the technique increases the delay.

In [Kam11, S.620-621] the method *Adaptive Subcarrier Selection* is introduced and in [SK98] further described. The subcarriers which are most affected by fading are not used to transmit any data. Instead they are used to alter the signal to prevent peaks. The algorithm needs a lot of calculations, but can reduce the peaks entirely under a defined threshold.

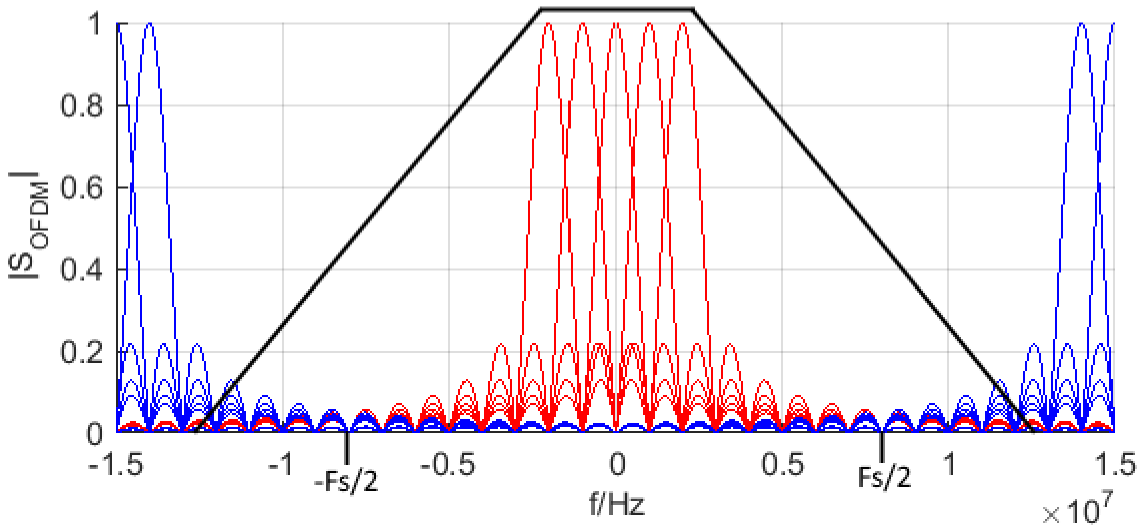
2.7. Impulse Shaping

The subcarriers are set in equal distances next to each other directly until the Nyquist frequency. Therefore, the mirror spectra of the original spectrum are right next to it, which results in challenging requirements for the analogue filters. When the mirror spectra of the transmission can be set farther apart, then the specification for the analogue low pass filters can be reduced [Kam11, S.613-614,637]. This can be achieved if the signal is oversampled.

If the subcarriers close to the edges of the frequency band are not used, it works similar to oversampling and can be accomplished by increasing the FFT size, refer to Figure 2.5. Subcarrier one lies on the Nyquist frequency $-F_s/2$ and therefore $F_s/2$



(a) The subcarriers on the edges are not used.



(b) The spectrum without using the subcarriers on the edges. The black trapeze represents the interpolation low-pass filter with the increased transition region. The blue lobes are the mirror spectra.

Figure 2.5.: Oversampling by not using subcarriers on the edges of the frequency band.

too. Thats why on the left side one more subcarrier is unused compared to the right side.

The rectangular functions of the transmit and receive filters have a si-function shape in the frequency domain [Kam11, S.613-615,621-625]. The si-function is defined as:

$$si(x) = \frac{sin(x)}{x}$$

The side lobes of this si-functions are decreasing poorly. This results in high out-of-band radiation. To handle this issue, the filters can be replaced with raised-cosine filters in the time-domain with a small roll-off-factor[Kam11, S.614].

2.8. Water Filling

The optimal distribution of the transmit power dependant on the state of the channel is called *water filling method* [Höh13, S.92-93,S.435]. The goal is to find the optimal solution which maximizes the channel capacity [Sha48, S.3].

If the channel is perfectly flat over the full bandwidth water filling is not necessary. But if the channel is frequency selective, then the noise power is different for every subcarrier. Figure 2.6 shows an example distribution, furthermore, the name water filling can be derived. It looks as if water was *poured* over the amplitude spectrum.

Subcarrier one has assigned most power because it has the least noise, while subcarrier 8 will not even be used. Hence, subcarriers with much noise should be modulated with Binary Phase Shift Keying (BPSK) while subcarriers with less noise should be modulated with higher order Quadrature Amplitude Modulation (QAM) variants. The two different symbols of BPSK are less vulnerable to noise than, for example, the 16 different symbols of 16-QAM.

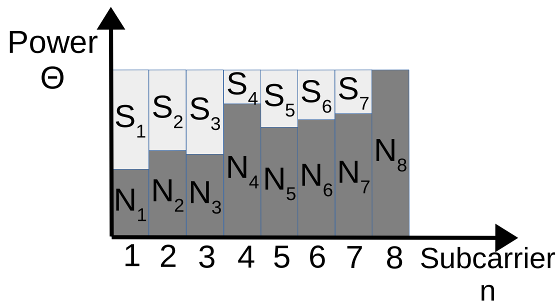


Figure 2.6.: An example of the water filling method [Höh13, S.435].

2.9. OFDM Compared to a Single-Carrier Transmission

This section shows some typical characteristics of an OFDM system compared to a single-carrier transmission:

- OFDM systems are more sensitive to timing jitter then single-carrier systems. Although, the GI is decreasing this sensitivity [Pro03, S.740].

- OFDM systems have problems with the high PAPR. In single-carrier transmissions with high roll-off-factors of the transmit filter, the modulation scheme can be chosen such the envelope is nearly constant. With low roll-off-factors [Kam11, S.599], to achieve spectral properties like in OFDM, the single-carrier transmission suffers similar problems with the PAPR.
- One of the biggest benefits of an multicarrier system is the possibility to use the water-filling method for bit- and power loading. It can be used to address the problems of PAPR and bad local channel conditions [Shi10, S.16].
- Single-carrier systems are transmitting the information over the full bandwidth. The data in an OFDM system can be addressed to specific subcarriers. Therefore, the total bandwidth is scalable. [Shi10, S.14].

3. Specification and Requirements

Based on the task description in section 1.1 this chapter describes the settings for all adjustable parameters of the system and how they are chosen.
This chapter provides information on system parts which are not included in the implementation. These description shall help the future designer who continues the work on the design.

The order in which the parameters are discussed is not necessary the order in which they have been calculated, but rather is optimized for understanding.

Another part of this chapter is the introduction to the requirements induced by the pre-created hardware platform and the used software.

3.1. Target Channel

As described before, the long-term goal of the overall project is to create a wireless transmission. It is possible to substitute the wireless link with a wired connection to achieve a more ideal channel. While writing this thesis another thesis was created in parallel to examine the behaviour of a wired channel with stubs [Ras16]. This is the reason why the created OFDM baseband controller shall be configurable to either OFDM or DMT, therefore, for wireless and wired channels, respectively. The implementation can be changed depending on the target channel.

For the design of the OFDM system it is crucial to know the length of the significant parts of the channel impulse response τ_{max}. In [Ras16] a value of several hundred nanoseconds was determined for an exemplary topology in a car. To describe a wireless channel for the Global System for Mobile Communications (GSM), there are several channels models defined by the workgroup Cost 207 [COS89]. These define impulse response durations of several microseconds.

3.2. OFDM Parameters

Guard Interval

The duration of the GI has to fulfil the condition $T_{GI} > \tau_{max}$, to make sure that the current symbol reaches a steady state while the previous symbol settles again before evaluation [Kam11, S.589]. For the basic version implemented during this thesis it is set to $T_{GI} = 224\,\text{ns}$, therefore, the assumed impulse response has to be $\tau_{max} < 224\,\text{ns}$. This assumption is relatively short, but extending T_{GI} as soon as a final value is determined is no problem. At this moment it helps to keep the amount of necessary subcarriers small. The needed changes to adapt the system to different values will be mentioned.

The GI introduces the problem of generating additional samples which have to be transmitted too. With the bandwidth efficiency set to $\beta = 0.8$, $20\,\%$ of the samples transmitted on the channel are introduced by the GI, therefore, are redundancy. This is a common value which is used in 802.11a/Wireless Local Area Network (WLAN) too [IEE, S.1643]. The data stream on the input of the system has to be stopped until the additional samples are transmitted.

Data Rate

The payload bit rate of the system shall be $R_{bit} = 50\,Mbit/s$, as mentioned in Chapter 1. This results in a bit period of $T_{bit} = \frac{1}{R_{bit}} = \frac{1}{50Mbit/s} = 20\,\text{ns}$. To accommodate the additional samples of the GI, while still achieving $R_{bit} = 50\,Mbit/s$, the system will be designed for a transmitted data rate on the channel of $R_{chn} = 62.5\,Mbit/s$, hence, the other parameters are calculated with bit rate $T_{calc} = T_{bit} \cdot \beta = 20\,\text{ns} \cdot 0.8 = 16\,\text{ns}$. The input bit stream has not to be stopped with this modification.

If in a later stage of development a channel coding shall be implemented, R_{bit} has to be reduced depending on the code rate or the other parameters are changed accordingly.

Number of Symbols

In the final product the water filling method, using the following modulation schemes, shall be applied:

- BPSK
- 4-QAM
- 16-QAM
- 64-QAM

Using only the rectangular K-QAM versions, with $M = ld(K)$ is even, allows the use of gray code for mapping of the symbols [Matb]. With this constraint every symbol error is only one bit error. BPSK to 64-QAM are typical values for modulation schemes in Radio Frequency (RF) applications, for example in 802.11a/WLAN [IEE, S.1602].

In the current stage of development 16-QAM without water filling is used.

Symbol Duration

Resulting from the definition of β and T_{GI}, the symbol duration T_{sym} can be calculated with an alteration of Equation 2.8.

$$T_{sym} = \frac{T_{GI}}{\frac{1}{\beta} - 1} = \frac{224\,\mathrm{ns}}{\frac{1}{0.8} - 1} = 896\,\mathrm{ns}$$

Used Subcarrier Number

The number of subcarriers necessary to handle the desired bit rate can be calculated with Equation 3.1 and is set to $N_{used} = 14$ [Kam11, S.591].

$$N_{used} = \frac{T_{sym}}{T_{calc} \cdot ld(K)} = \frac{896\,\mathrm{ns}}{16\,\mathrm{ns} \cdot 4} = 14 \tag{3.1}$$

Unused Subcarrier Number

There are three constraints to consider while distributing the data to the subcarriers:

1. To make the design for the analogue filters less challenging on both sides of the spectrum some carriers should not be used. This corresponds to a oversampling.
2. The carrier in the middle of the spectrum should not be used because it is DC in the baseband [Kam11, S.630].

3. The left-most carrier is the carrier at the Nyquist frequency it should not be used either.

The current state of the project considers only constraints two and three. The distribution is shown in Figure 3.1. With the use of $N_{used} = 14$ and $N_{unused} = 2$ the total amount of subcarriers is $N_{sc} = 16$.

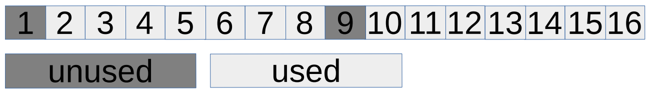


Figure 3.1.: The distribution of the subcarriers.

While using an analogue front-end more unused subcarriers should be inserted on the sides of the spectrum to meet constraint one as well.

FFT Length

The length of the FFT for the OFDM system is $N_{OFDM} = N_{sc} = 16$. Since the DMT variant has to achieve a real-valued output signal, the FFT length is $N_{DMT} = 2 \cdot N_{sc} = 32$, refer to section 2.2.

Signal Bandwidth

The subcarrier spacing can be calculated with Equation 2.1 to

$$B_{sc} = 1/896\,\mathrm{ns} = 1.116\,\mathrm{MHz}$$

which results in a total bandwidth for OFDM in the complex baseband of :

$$B_{OFDM} = \frac{N_{OFDM}}{2} \cdot B_{sc} = \frac{16}{2} \cdot 1.116\,\mathrm{MHz} = 8.929\,\mathrm{MHz}$$

The DMT variant needs twice the bandwidth of OFDM because the complete information has to be transmitted in the real-part of the signal:

$$B_{DMT} = \frac{N_{DMT}}{2} \cdot B_{sc} = \frac{32}{2} \cdot 1.116\,\mathrm{MHz} = 17.857\,\mathrm{MHz}$$

Consider that after applying a quadrature modulator to raise the complex baseband OFDM signal to the desired frequency the signal is real-valued too and needs the same bandwidth like the DMT signal.

3.3. Parameters of the AWGN Channel Model

To measure the performance of the system the BER using an AWGN channel model shall be determined. This section shows how the AWGN channel model is parametrized and which BER is expected.

Average Symbol Power

The orthogonality of the subcarriers allows to calculate each subcarrier on his own and add them together afterwards. Therefore, each subcarrier can be handled as an K-QAM modulation like in a single carrier transmission.

The average power can be calculated according to Figure 3.2 in combination with Equation 3.2 [Vol]. The values a_i represent the symbols in the complex plane and each consists of an real- and imaginary-part, like $1 + i1$. The probability of each symbol is the same. $2d$ is the distance of one square in the figure and is set to $d = 1$. For 16-QAM K is set to 16 and since a power is calculated a resistance of 1Ω is assumed.

$$\overline{P}_{sym} = \frac{d^2}{K} \sum_{i=1}^{K} |a_i|^2 \tag{3.2}$$

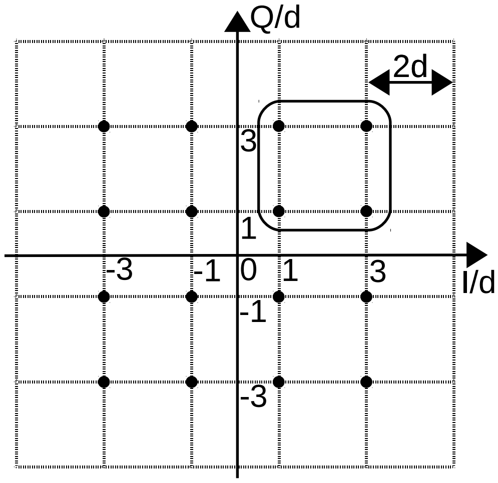


Figure 3.2.: The constellation for 16-QAM.

The calculation can be simplified if only the symbols in the square in the figure are calculated and the result is multiplied by the factor four. This is possible because the constellation is symmetric.

$$\{\overline{P}_{sym}\} = 4 \cdot \frac{\{d\}^2}{K} \sum_{i=1}^{K/4} |a_i|^2 = 4 \cdot \frac{1^2}{16}((1^2+1^2)+(2\cdot(1^2+3^2))+(3^2+3^2)) = \frac{1}{4}\cdot 40 = 10$$

Hence, the average power for 16-QAM is 10 W.

BER for 16-QAM

Equation 3.3[Kam11, S.387] shows an approximation of the bit error probability for general M-QAM on an AWGN channel. Due to the gray coding it is assumed that each symbol error is only one bit error, furthermore, the probability that a symbol gets detected as a symbol which is not a direct neighbour in the constellation diagram is neglected. E_b describes the average energy per bit and N_0 the Power Spectral Density (PSD) of the white Gaussian noise.

$$P_{b,M-QAM} = \frac{2}{ld(M)}\left(1-\frac{1}{\sqrt{M}}\right) erfc\left(\sqrt{\frac{3\cdot ld(M)}{2(M-1)}\frac{E_b}{N_0}}\right) \tag{3.3}$$

The term $erfc()$ is the complementary error function defined by [Wol]:

$$erfc(z) = 1 - erf(z) = \frac{2}{\sqrt{\pi}}\int_z^\infty e^{-t^2}dt$$

For 16-QAM this results in Formula 3.4.

$$P_{b,16-QAM} = \frac{3}{8} erfc\left(\sqrt{\frac{2}{5}\frac{E_b}{N_0}}\right) \tag{3.4}$$

For easier handling the term $\frac{E_b}{N_0}$ shall be replaced with the Signal to Noise Ratio (SNR). First, $\frac{E_b}{N_0}$ is replaced with $\frac{E_s}{N_0}$. The term E_s describes the average energy per symbol. The relation between these is $4 \cdot E_b = E_s$, because one symbol consists of 4 bits in 16-QAM.

Due to the fact that the receiver fulfils the matched filter principle [Höh13, S.431], the conversion for $\frac{E_s}{N_0}$ and SNR can be described as in Equation 3.5 for the complex baseband channel. For an only real-valued modulation, Formula 3.6 has to be used [Kam11, S.255,778]. The difference results from the fact that the complex baseband channel consists of a separate channel for the real-valued part and another one for the imaginary-valued part of the signal. Therefore, the noise has to be accounted for twice.

$$SNR_{comp} = \frac{E_s}{N_0} \tag{3.5}$$

$$SNR_{real} = \frac{E_s}{N_0/2} \tag{3.6}$$

It has to be taken account for that the GI slightly mismatches the matched filter. This can be considered as an reduction of SNR, for $\beta = 0.8$ this results in $10 \cdot log(0.8) = -0.969dB$. Furthermore, the fact that $N_{unused} = 2$ subcarriers are not used, can be described as an reduction of the SNR too. With 14 of 16 subcarriers used this results in $\delta = \frac{14}{16} = 0.875$, or $-0.58dB$. In a linear scale these factors have to be multiplied, see Equation 3.7.

$$\gamma = \delta \cdot \beta = 0.875 \cdot 0.8 = 0.7 \tag{3.7}$$

Using 3.5 and 3.7 the final Formula 3.8 is achieved.

$$P_{b,16-QAM} = \frac{3}{8} erfc\left(\sqrt{\frac{1}{10}(SNR \cdot 0.7)}\right) \tag{3.8}$$

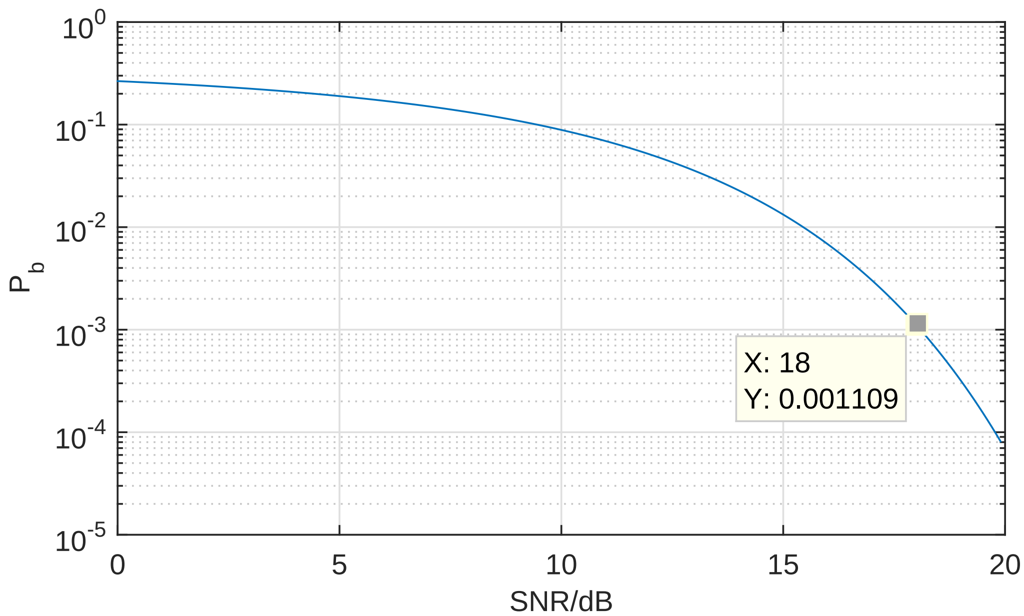


Figure 3.3.: The bit error probability for 16-QAM.

Since the SNR of the final channel is not known yet, the value is assumed to $18\,dB$.

This is a relatively low value which should be higher while using the design in a car, therefore, a worse case is described. Figure 3.3 shows that with the assumed SNR of $18\,dB$ a bit error probability of $1.109 \cdot 10^{-3}$ is expected.

Signal Power for AWGN

The AWGN channel is modelled with the Simulink AWGN block [Mata]. The block has the parameters SNR and input signal power P_s, referenced to 1Ω. Equation 3.9 shows the calculation for OFDM.

$$\overline{P}_s = \overline{P}_{sym} \cdot N_{OFDM} \cdot \frac{N_{used}}{N_{sub}} \cdot \frac{N_{OFDM}}{N_{OFDM} + N_{GI,OFDM}} \cdot 2 = 10\,\mathrm{W} \cdot 16 \cdot \frac{14}{16} \cdot \frac{16}{16+4} \cdot 2 = 224\,\mathrm{W} \tag{3.9}$$

The term $\overline{P}_{sym}$ describes the average symbol power defined in the frequency domain. Because the IFFT is calculated of the signal, using the IDFT, this value has to be multiplied with N_{OFDM}. This results of Parseval's theorem which indicates that the power is multiplied with $\frac{1}{N_{OFDM}}$ while performing the transform [Wik]:

$$\sum_{l=0}^{N_{OFDM}-1} |x[l]|^2 = \frac{1}{N_{OFDM}} \sum_{m=0}^{N_{OFDM}-1} |X[m]|^2$$

The term $\frac{N_{used}}{N_{sub}}$ takes into account that not all subcarriers are used and therefore are not increasing the signal power. $\frac{N_{OFDM}}{N_{OFDM}+N_{GI,OFDM}}$ describes the reduction of the signal power in respect to the GI. The multiplication by 2 is implementation specific for the channel model.

Formula 3.10 shows the calculation for DMT.

$$\overline{P}_s = \overline{P}_{sym} \cdot N_{DMT} \cdot \frac{N_{used}}{N_{sub}} \cdot \frac{N_{DMT}}{N_{DMT} + N_{GI,DMT}} \cdot 2 = 10\,\mathrm{W} \cdot 32 \cdot \frac{14}{16} \cdot \frac{32}{32+8} \cdot 2 = 448\,\mathrm{W} \tag{3.10}$$

3.4. ADC/DAC Converter Specification

Sampling Frequency

With the bandwidth B_{OFDM} the needed sampling frequency of the DAC in the transmitter and the ADC in the receiver can be calculated as:

$$f_{s,OFDM} = 2 \cdot B_{OFDM} = 17.857\,\mathrm{MHz}$$

The converters need two channels, one for the real-part, and one for the imaginary-part of the signal. The sampling period is $T_{s,OFDM} = 1/f_{s,OFDM} = 1/17.857\,\mathrm{MHz} = 56\,\mathrm{ns}$.

The DMT signal needs only a single DAC/ADC channel because it is already a real-valued signal. However, the sampling frequency has to be twice as high:

$$f_{s,DMT} = 2 \cdot B_{DMT} = 35.714\,\mathrm{MHz}$$

This results in a sampling period of $T_{s,DMT} = 1/f_{s,DMT} = 1/35.714\,\mathrm{MHz} = 28\,\mathrm{ns}$.

Number of Bits

The 16-QAM modulation needs at least 3 integer bits to represent the highest values $-3; +3$ including the sign bit. The use of the FFT in radix-2 configuration increases the necessary bits in the worst case accordingly to Equation 3.11 [Xil15, S.28]:

$$N_{int} = N_{input} + ld(N_{FFT}) + 1 \tag{3.11}$$

For the OFDM mode this results in

$$N_{int} = N_{input} + ld(N_{OFDM}) + 1 = 3 + ld(16) + 1 \mathrel{\widehat{=}} 8\,\mathrm{bit}$$

while the DMT mode needs:

$$N_{int} = N_{input} + ld(N_{DMT}) + 1 = 3 + ld(32) + 1 \mathrel{\widehat{=}} 9\,\mathrm{bit}$$

As soon as a method to reduce the PAPR is implemented, the integer bit number can be reduced in favour of the fractional bit number.

The number of fractional bits is set to $N_{frac} = 3$. This allows to approximate the quantization noise to $3 \cdot 6.02\,dB = 18.06\,dB$[Her14, S.483].

ADC/DAC Resolution

With the derivation of the number of integer- and fractional bits the total converter resolution can be determined. With $N_{conv} = N_{int} + N_{frac} = 9 + 3 = 12$ the converter needs 12 bits for DMT and 11 bits for OFDM. Reducing N_{int} can be considered since it is a worst case value. The value is set to $N_{int} = 6$ in the model, this can increase slightly the BER because the clipping can result in bit errors. That can be considered as a very simple method to handle PAPR. The multiply and add operations in the equalizer and the channel models are increasing the word width. After these operations the word width is cast to their input width this can result in bit errors. The increase of the BER due to quantization is further investigated in section 6.5. Expressed in the Q fixed-point number format the precision is set to s5Q3 [Rei11, S.82-84]:

- **s** one sign bit
- **5** integer bits
- **3** fractional bits

3.5. Hardware Platform

The system is implemented on the MircoZed development board which is based on the Xilinx Zynq-7000 All Programmable SoC[Zed]. The MicroZed is plugged into a carrier board used as communication controller in the X-by-wire(less) project, refer to Figure A.1 in the appendix on page 84. This board was designed in the scope of the thesis [Adl15].

In this thesis the carrier board is used as power supply for the MicroZed and routing of the pins from the FPGA to the *Connection Extension Board*. The Connection Extension Board has several SMB connectors which allow to measure the signals from the FPGA pins.

MicroZed

There are MicroZed versions with two different FPGAs available. The used MicroZed uses the bigger version with the XC7Z7020 FPGA.

Part Number	XC7Z020
Look-Up Tables (LUTs)	53200
Flip-Flops	106400
Total Block RAM (# 36Kb Blocks)	4.9Mb (140)
Programmable DSP Slices (18x25 MACCs)	220

Table 3.1.: The resources of the PL of the MicroZed [Xil].

Table 3.1 gives an overview of the available resources of the Programmable Logic (PL). Furthermore, every Zynq-7000 device contains an hard core Dual ARM© Cortex™-A9 MPCore™, called Processing System (PS), which can be programmed and interact with the PL. This project uses mainly the PL part and only the oscillator of the PS as a clock source for the PL. The block diagram 3.4 gives an overview of the capabilitys of the MicroZed.

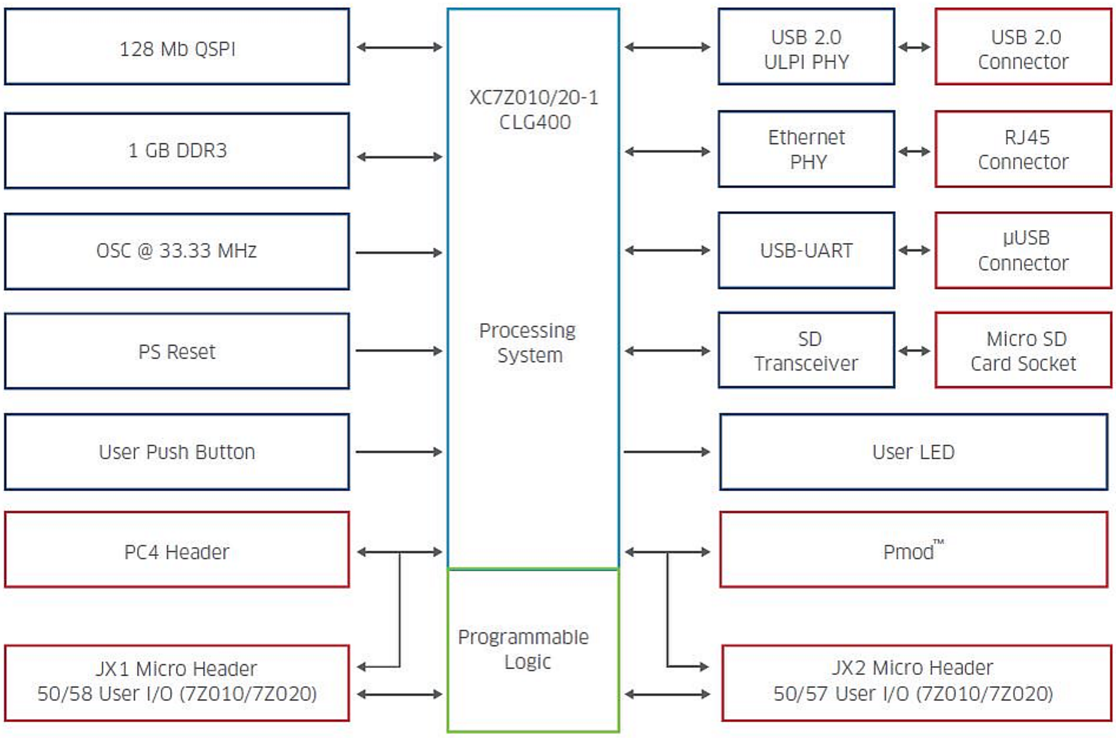


Figure 3.4.: The function block diagram of the MicroZed [AVNa].

Figure A.2 on page 85 shows the MicroZed board.

Connection Extension Board

The Connection Extension Board is plugged onto the carrier board like the MicroZed. It allows for example to connect an oscilloscope and check if the implemented system on the FPGA works as expected, refer to Figure A.3 on page 85. In a later stage of development the Connection Extension Board can be removed and replaced with another board holding the physical layer for a wired or wireless connection.

3.6. Software Requirements

The design is implemented and tested with the software in the following list. Fur using and further development it is recommended to use the same versions.

- MATLAB 2014b with Simulink and the following toolboxes
- Communications System Toolbox
- DSP System Toolbox
- Fixed-Point Designer
- Simulink Coder
- Vivado 2014.4 System Edition Design Suite. With System Generator for DSP

4. Implementation

This chapter describes the implementation with System Generator in Simulink. That includes more general implementation specific design decisions and the actual implementation. All source codes can be found on the data medium, refer to section A.2.

4.1. Development Paradigms

The creation of the FPGA netlist through System Generator blocks is a different approach in respect to a Very High Speed Integrated Circuit Hardware Description Language (VHDL) model which shall shorten the development time.

Model Architecture

The System Generator building blocks have more parameters than Simulink blocks which have to be configured. For many necessary operations a complete Simulink block already exists, while a corresponding System Generator block has to be created out of multiple basic blocks, for example the 16-QAM.

Given this circumstance, it is useful to implement the model first in Simulink and in parallel with System Generator blocks. The additional Simulink blocks do not increase the implementation effort very much, but allow to create the expected behaviour with the easier configurable Simulink blocks and to compare the results with the outputs of the System Generator blocks. This can easily be achieved by subtracting the result of a given Simulink block, from the result of the equivalent System Generator block to get the difference signal.

The whole model consists of equivalent Simulink and System Generator blocks which are built in parallel while the difference signal is created after each function block. This procedure allows to find errors after changes in one part of the model by giving the opportunity of checking whether the difference signal did not change.

Keeping the Word Width

The word width is increased by some blocks within the model. Usually, the output word width of these blocks can be cast to their input word width without significantly increasing the BER. This is possible, because the specified number of integer bits $N_{int} = 6$ is high enough to make overflows unlikely.

All blocks which need to quantize a value are configured to *convergent rounding* instead of *truncation* this prevents an induces bias. The blocks which suffer an overflow are configured to *saturate* in place of *overflow* to prevent wrong calculations [Matc].

Both measures increase the required hardware resources which is accepted because the resource utilization causes no problems.

4.2. System Clock Frequency

A model in Simulink needs a fundamental sample period $T_{clk} = \frac{1}{f_{clk}}$ of which all other sample periods in the system are multiples. The fundamental sample period defines the clock frequency which the block needs to be supplied with to run on the FPGA. All other clock frequencies within the system are created thorough the use of clock enable signals which run at multiples of T_{clk}. For each required clock, a separate enable signal is used while all blocks are supplied with the fundamental clock. [Xil14a, S.43].

The fundamental sample period T_{clk} has to fulfil some requirements.

- It has to be an integer divider of the transmitter input and receiver output sample period of 20 ns.
- It has to be an integer divider of the transmitter output and receiver input sample period of 28 ns for DMT and 56 ns for OFDM.
- The clock signal shall be provided by the PL clock output of the PS of the MicroZed. The maximum providable sample period is $4ns$.

The greatest common divisor is $gcd(20ns, 28ns) = 4\,\text{ns}$ and $gcd(20\,\text{ns}, 56\,\text{ns}) = 4\,\text{ns}$. Hence, $T_{clk} = 4ns$ is chosen.

4.3. Data Handling

At first glance it appears to be the easiest way to process the values with the frequency they shall appear at the output. Therefore, the data is handled as an continuous stream. When using a GI, the GI creates additional samples while not changing the sample rate which makes it necessary to stop the data stream until the additional samples have been sent.

A First In First Out (FIFO) directly at the input and output of the transmitter and receiver can handle this issue. Data values can be collected in the input FIFO until a complete block is present and afterwards popped out to be processed at an arbitrary rate. The output FIFO can be used to generate the expected sample period at the output.

Integer Multiple of the Fundamental Period

Another challenge is the necessity that all sample periods in the system are multiples of $T_{clk} = 4ns$. Therefore, any sample period T_a has to fulfil the condition in Equation 4.1.

$$\frac{T_a}{T_{clk}} \in \mathbb{N} \tag{4.1}$$

With the number of used subcarriers $N_{used} = 14$ a multiple of 2^n is not achieved in all stages of the design. Equation 4.2 shows the calculation for the sample period at the IFFT while the condition is not achieved.

$$\frac{T_a}{T_{clk}} = \frac{T_b \cdot M \cdot N_{used}/N_{sc}}{T_{clk}} = \frac{20ns \cdot 4 \cdot 14/16}{4ns} = \frac{35}{2} \notin \mathbb{N} \tag{4.2}$$

One possibility to handle that issue is to insert another FIFO for each of the parallel data streams before they are converted back to serial transmission which allows to output them at a faster rate which fulfils the condition in Formula 4.1.

A simpler way to do this is to process the data as if N_{used} is a multiple of 2^n and not evaluate the additional values. Then $N_{unused} = 2$ additional 16-QAM modulators are necessary which are fed with artificial data.

4.4. Configuration of the Model

To allow easy experimenting with the parameters of the model a Graphical User Interface (GUI) is implemented. With a double click on the block `Edit Model Parameters` in the model `Basic_DMT.slx` the configuration can be changed. The block opens a small GUI, see Figure 4.1.

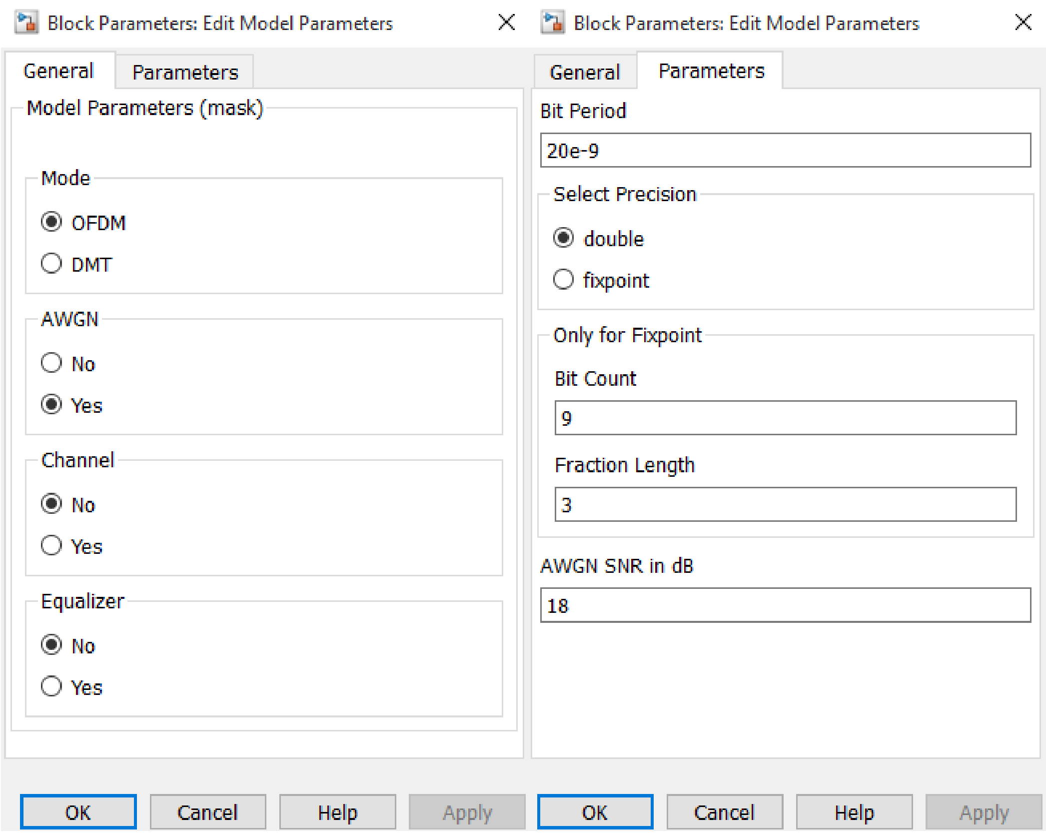


Figure 4.1.: The configuration GUI of the model.

The following options are accessible:

- Selection of OFDM- or DMT-mode.
- Activation of the AWGN channel model.
- Activation of the FIR-filter based channel model.
- Activation of the equalizer.
- Selection of the bit period of the data source.

- Selection of the SNR of the AWGN channel model.

Furthermore, it has to be selected if the model shall use double- or the fixed-point-precision. This selection changes the configuration only for the Simulink blocks, the System Generator Model always uses fixed-point arithmetic. The total number of bits and fractional bit number can be set too, these fields are always evaluated for the System Generator blocks while they are only used for the Simulink blocks if fixed-point arithmetic is selected.

After clicking the 'Ok' button in the GUI the script `initScript.m` is run which changes the parameters of the model depending on the values in the GUI. Additionally, after opening the model the script is run to initialize it.

4.5. Data Source

For testing of the system a data source has to be created which generates a stimulus. It generates random bits with an equal probability for `zeros` and `ones`. For the regular operation in Simulink the `Bernoulli Binary Generator` block is used which generates Bernoulli distributed values, therefore, only two different values are produced which fulfils the requirements for generating random bits.

Synthesizable Data Source

For testing the system on the FPGA a synthesizable random bit generator is necessary. To achieve this, an additional random bit generator with System Generator blocks is created. The generator uses an Linear Feedback Shift Register (LFSR) with 15 registers in feedback to generate a Maximum Length Sequence (MLS). With the generator polynomial

$$h(x) = x^{15} \oplus x^{1} \oplus 1$$

a sequence with the length $2^{15} - 1 = 32767$ is achieved before the sequence repeats [Jon03], refer to Figure 4.2.

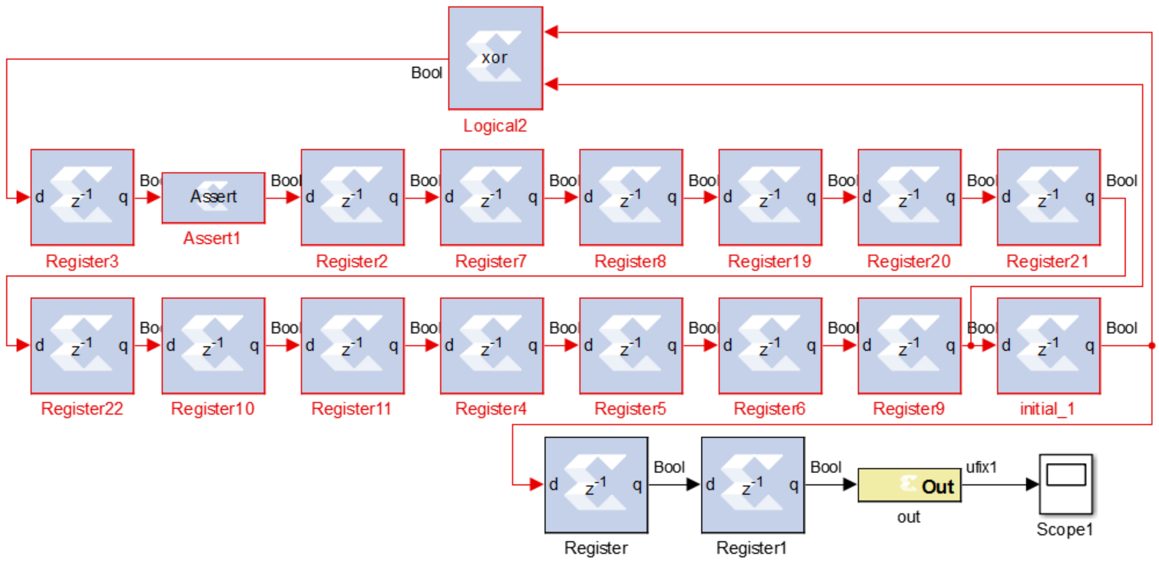


Figure 4.2.: The LFSR for the random bit generator.

Debugging

For debugging purposes it can be useful to not use random data, but rather transmit constant `zeros` or constant `ones`. This allows to predict the expected signal and debug the model. Therefore, in the Simulink environment an additional source is implemented which generates a sequence containing only `ones`.

4.6. Transmitter

Serial to Parallel

This section describes the implementation of the block `S/P` of the transmitter in Figure 2.1 on page 16.

The incoming bits through the block `inp_trans` with the period $T_b = 20\,\text{ns}$ are stored in a FIFO, refer to Figure 4.3. This allows to process the data at the system rate $T_{clk} = 4\,\text{ns}$, because they can be output at an arbitrary rate as long as enough samples are inserted. To do this, the bits are stored in the FIFO until a block of 56 bits is completed. This number results from the $N_{used} = 14$ used subcarriers and the $M = 4$ bit per symbol of the 16-QAM.

The FIFO block needs the same sample rate at every input, the zero-order hold

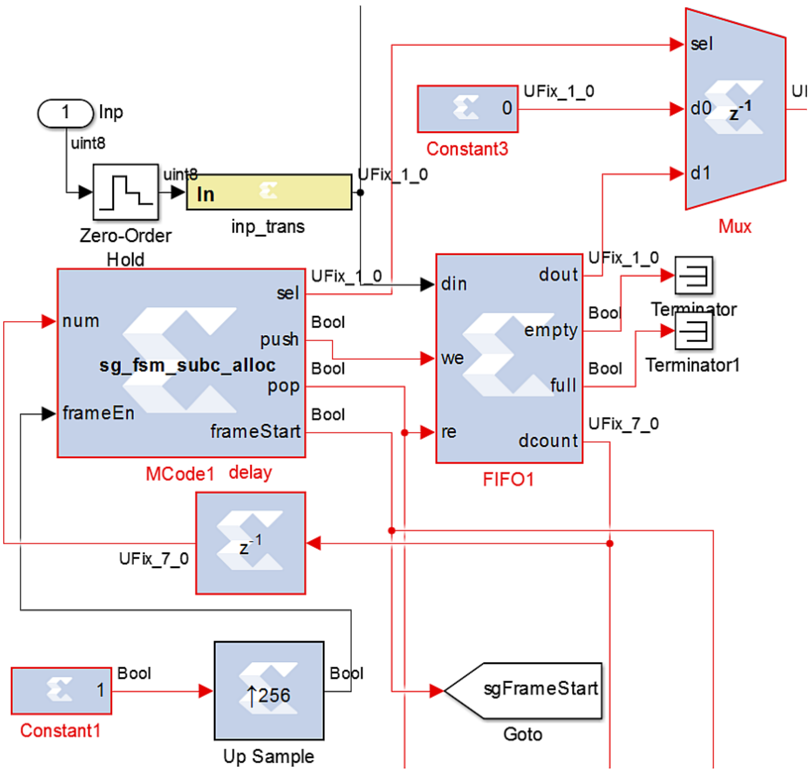


Figure 4.3.: The transmitter input FIFO with FSM.

block is used to resample the input signal to $T_{clk} = 4\,\text{ns}$, because the other inputs use this period. Hence, each bit is sampled 5 times.

Distribution of the Bits

The bits are distributed between the N_{used} subcarriers and fed to the modulators. The first $M = 4$ are assigned to the first used subcarrier, the second M bits are assigned to the second subcarrier used. This pattern is continued until all bits are distributed.

The push and pop signals for the FIFO are generated by an Finite State Machine (FSM) in the `MCode1` block, see Figure 4.3. The multiplexer is used to ensure that the 16-QAM modulators can be fed with artificial data for the unused subcarriers, in the model this is done by transmitting `zeros`. Because the FIFO operates with the sample period of $T_{clk} = 4\,\text{ns}$ and new bits arrive with $T_b = 20\,\text{ns}$, the FIFO outputs values only in 20% of the time, refer to figure 4.4.

The blue rectangular signal describes the possible start points for popping out bits. When no complete data block is available at the beginning of a frame the multiplexer

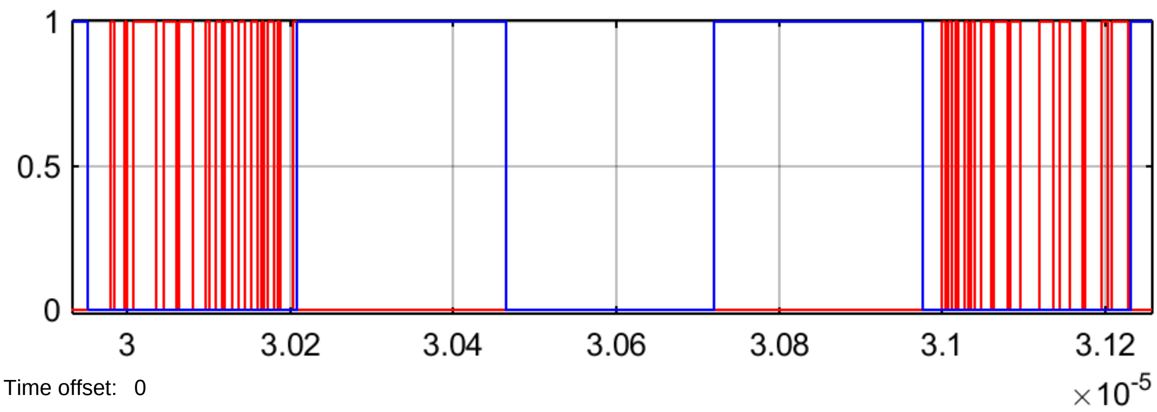


Figure 4.4.: The output signal of the FIFO.

is used to feed artificial, not evaluated data to the modulators. This frame based processing is necessary because the system has to make sure that a complete block of data, representing all subcarriers, reaches the IFFT together. The select signal for the multiplexer is also generated by the FSM.

FSM

The FSM consists of four states to control the FIFO and multiplexer. Table 4.1 shows the logic table for the output signals depending on the states.

state	push	pop	sel
0	0	0	0
1	0	1	1
2	1	0	0
3	1	1	1

Table 4.1.: Possible output values of the FSM dependent of the states.

The signals `pop` and `sel` could be merged, but are separated for clarity. Each state can transit to every other state. The output signals of the FSM have a fixed sequence because the input signals are periodic. The signal `frameStart` is a Mealy signal and therefore not shown in the table.

Figure 4.5 shows a detailed Algorithmic State Machine (ASM) chart [Pon06, S.317-321, 376-381] of the FSM. The block `A` substitutes the next state logic, because the same is used for all states. The ASM chart for the next state logic is displayed in

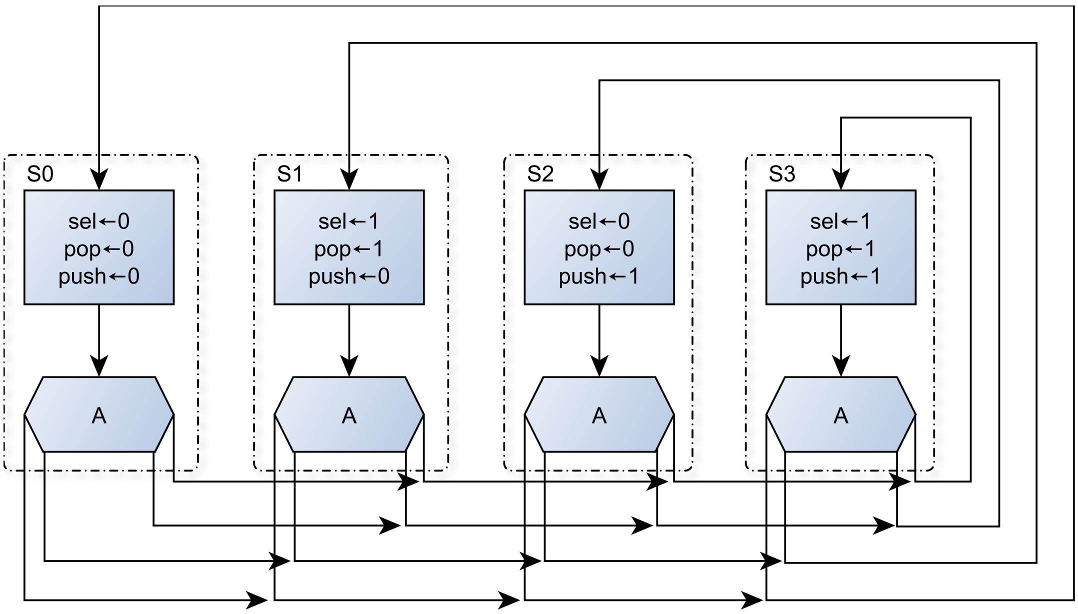


Figure 4.5.: The ASM chart of the FSM at the input of the transmitter. The block `A` is substituted by the logic in figure 4.6.

Figure 4.6. Table 4.2 shows all signals including the internal registers. All internal registers are initialized to 0.

The file `sg_fsm_subc_alloc.m` contains the source code for the FSM.

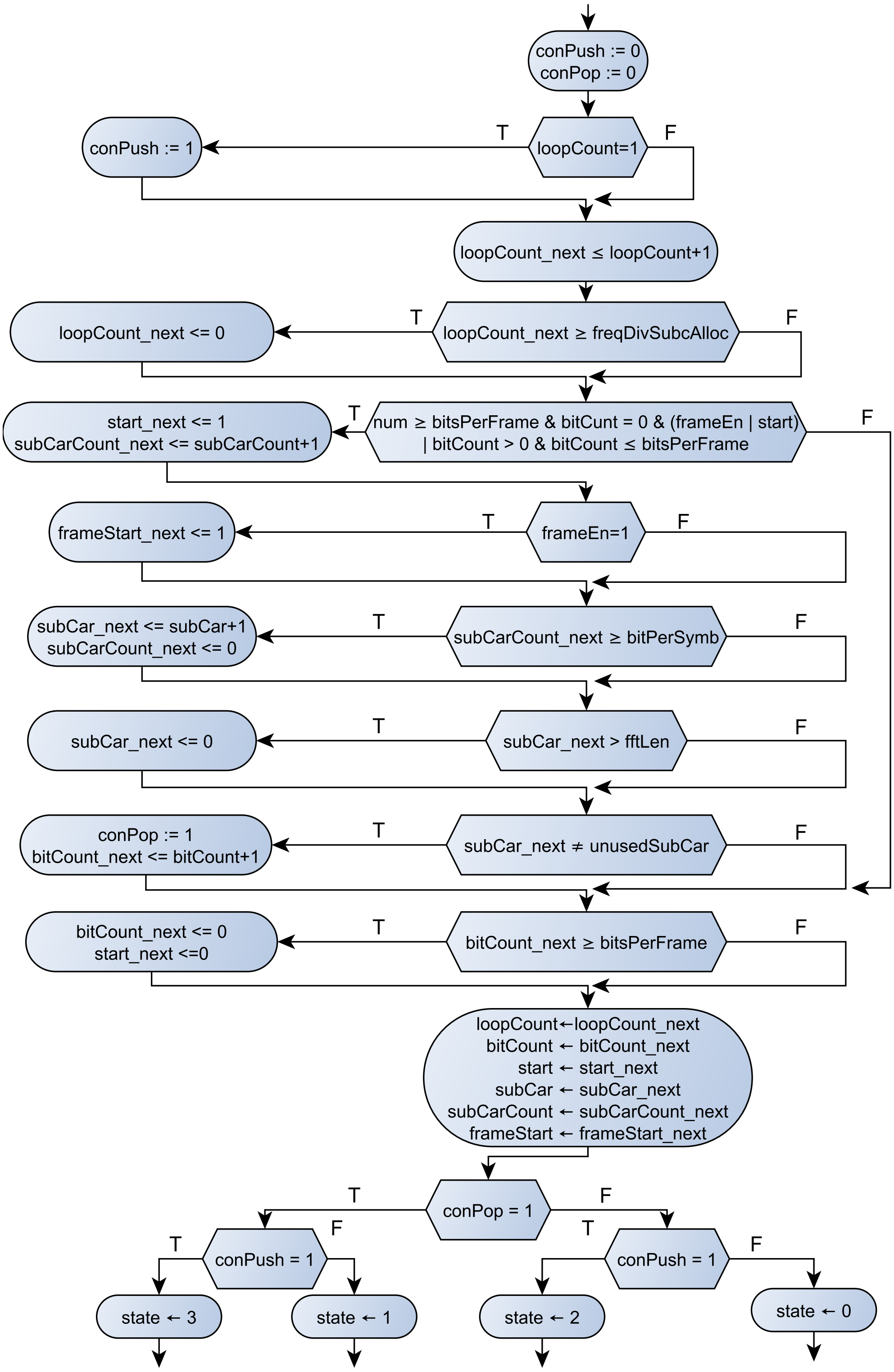


Figure 4.6.: The ASM chart of the transition logic of the FSM at the input of the transmitter. This diagram substitutes the block `A` in figure 4.5.

Signal type	**Name**	**Description**
Input	`num`	Number of bits in the FIFO
Input	`frameEn`	Clock signal which unlocks the possibility for a new frame to start
Output Moore	`sel`	Controls the multiplexer for the use of real data or arbitrary data
Output Moore	`push`	Signals the FIFO to insert a bit
Output Moore	`pop`	Signals the FIFO to output a bit
Output Mealy	`frameStart`	Signals the GI that a new frame starts
Internal register	`state`	Stores the state
Internal register	`bitCount`	Counts the already sent bits in the current frame
Internal register	`start`	Asserts that a frame is started
Internal register	`subCarCount`	Controls the number of bits that are assigned to a single subcarrier. Each subcarrier receives $M = 4$ bits for 16-QAM
Internal register	`subCar`	Controls the curent subcarrier the bits are assigned to
Internal register	`loopCount`	Controls that each bit is only inserted once into the FIFO
Constant	`fftLen`	The number of subcarriers
Constant	`bitPerSymb`	The bits per symbol for 16-QAM
Constant	`bitsPerFrame`	The product of $N_{used} = 14$ and $M = 4$
Constant	`freqDivSubcAlloc`	The bit period $T_b = 20\,\text{ns}$ divided by the system period $T_{clk} = 4\,\text{ns}$. Used to determine when a value has to be pushed into the FIFO
Identifier variable	`conPush`	Substitutes the logic to determine if in the next cycle the input value has to be pushed
Identifier variable	`conPop`	Substitutes the logic to determine if in the next cycle a value has to be poped

Table 4.2.: All signals of the transmitter input FSM.

Modulation

This section describes the implementation of the modulator blocks of the transmitter in Figure 2.1 on page 16.

After exiting the FIFO the data is divided in $N_{sc} = 16$ parallel streams with $M = 4$ bits each. Every stream is fed to a 16-QAM modulator like in Figure 4.7. The `BitBasher` block divides the 4-bit input word to the inphase- and quadrature paths. The resulting 2-bit words are used as select signals for multiplexers which choose the corresponding constants. The period of each data word is $T_p = T_{clk} \cdot N_{sub} \cdot M = 4\,\text{ns} \cdot 16 \cdot 4 = 256\,\text{ns}$ due to the parallelization.

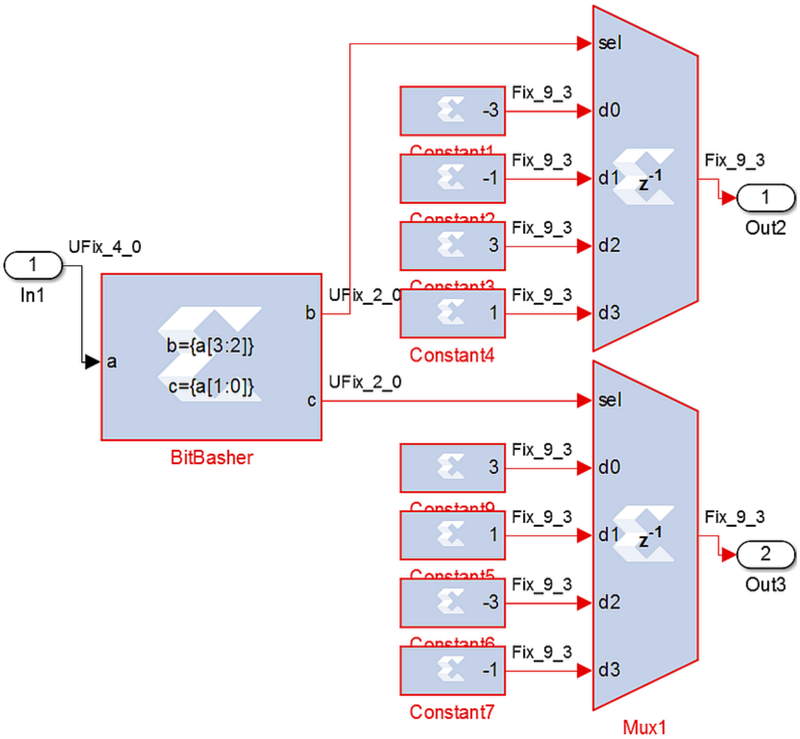


Figure 4.7.: Model for the 16-QAM modulator.

Create Frame for IFFT

This section describes the implementation of the block `P/S` of the transmitter in Figure 2.1 on page 16.

After the modulation is done, the data is converted back to serial order. Therefore, the word period is reduced again to $T_{IFFT,OFDM} = T_p/N_{sc} = 256\,\text{ns}/16 = 16\,\text{ns}$. In

DMT mode the data has to be arranged like in Figure 2.3 on page 19 which results in doubling the amount of data and the word period has to be $T_{IFFT,DMT} = T_p/N_{sc}/2 = 256\,\text{ns}/16/2 = 8\,\text{ns}$.

This is done with a multiplexer having $N_{sc} = 16$ inputs, wherein each is connected to the output of a modulator. The select input of the multiplexer is connected to a counter, with the frequency $1/T_{IFFT,OFDM}$ and $1/T_{IFFT,DMT}$ respectively. This circuit is implemented twice, once for the real- and once for the imaginary part of the signal.

Furthermore, the values of the unused subcarriers are set to zero again because the initially transmitted zero sequence is modulated to $-3 + 3i$. This step is done by a `MCode` block with the code in the file `sg_frame.m`. All subcarriers are fed into this block before they are routed to the multiplexer.

IFFT

This section describes the implementation of the block `IFFT` of the transmitter in Figure 2.1 on page 16.

The serial data stream is fed into the System Generator FFT block, refer to Figure 4.8 for the block in OFDM mode. The following list describes the inputs with the corresponding circuitry:

- `config_tdata_fdw_inv`: This input allows to configure the mode of the block. A `one` sets the block to FFT mode and a `zero` for IFFT.

- `config_tvalid`: It is possible to change the configuration during runtime. This input signals the block that the `config` signals are valid. Because it is not required to change the configuration for the designed system, the block is set to a constant `one`. The pre circuit is used to ensure that the signal is not `one` until the second clock cycle to prevent errors.

- `data_tdata_xn_im_0`: This input receives the serial data stream of the imaginary part of the signal. The block expects the signal in fixed point format without integer bits, the reinterpret block moves the decimal comma accordingly without changing the bits.

- `data_tdata_re_im_0`: This input receives the serial data stream of the real part of the signal.

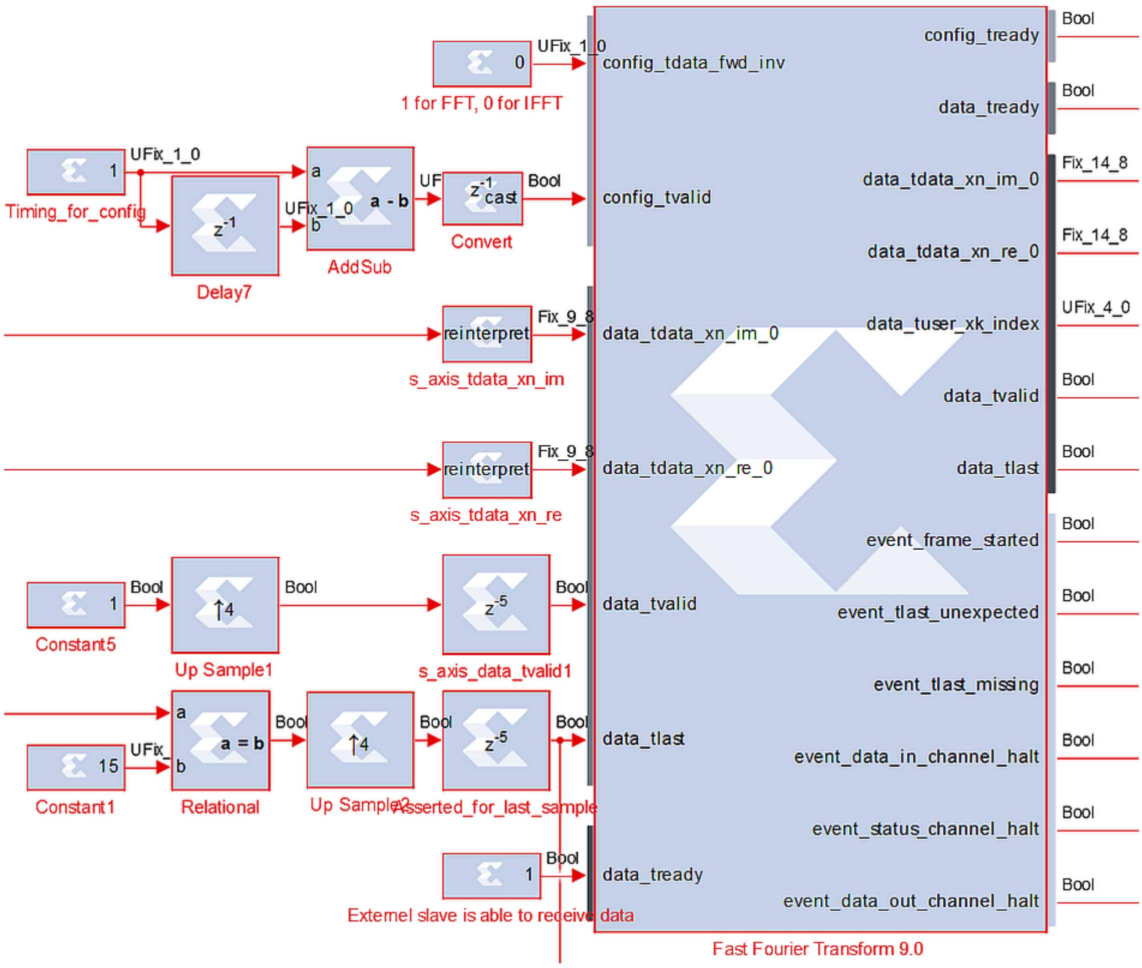


Figure 4.8.: The System Generator block for the IFFT with configuration circuits for OFDM mode.

- `data_tvalid`: This signal indicates to the block if the data is valid. The block expects this signal with the system rate $T_{clk} = 4\,\text{ns}$. It reads the values at `data_tdata_re_im_0` and `data_tdata_xn_im_0` into the block when `data_tvalid` is `one`, these signals have the period $T_{IFFT,OFDM} = 16\,\text{ns}$. The upsampling by 4 ensures that the signal has the system rate while its `one` for only 4 ns because the upsample block inserts zeros behind the signal. This prevents that the same value is processed multiple times.

- `data_tlast`: This signal indicates to the block if the current data sample is the last of a frame. The `a` input of the block `Relational` is connected to the counter which controls the multiplexer of the frame generation. It is compared to the value 15 to check if the last value is present. The upsampling block is necessary because the signal is expected to have the system rate.

- `data_tready`: This input allows the receiver of the data from the FFT to assert that it is ready to receive data. It is not used, hence tied to one.

This list describes the output signals:

- `config_tready`: This signal is asserted when the block is ready to receive configuration data. The port is not used because the configuration is not changed during runtime.
- `data_tready`: This signal is asserted when the block is ready to receive data.
- `data_tdata_xn_im_0`: Outputs the serial data stream of the imaginary part of the signal. The word width is increased according to Equation 3.11 on page 35. The signal width is cast to the input width to keep the amount of required hardware resources low, this can cause bit errors which can be neglected. The output period is $T_{clk} = 4\,\mathrm{ns}$, to regenerate the period at the input of the IFFT $T_{IFFT,OFDM}$ block a FIFO is used in which the values are written. The pop signal for the FIFO is a clock with the rate $1/T_{IFFT,OFDM}$ which is generated by a counter.
- `data_tdata_re_im_0`: Outputs the serial data stream of the real part of the signal. The data is cast and stored in a FIFO like the imaginary part.
- `data_tuser_xk_index`: This signal indicates the number of output values for the current frame.
- `data_tvalid`: Asserts that the output data is valid. This is used as push signal for the FIFO.
- `data_tvalid`: Asserts the last value of a frame.
- The block has several event outputs. For example the signals `event_tlast_unexpected` and `event_tlast_missing` stay zero all the time if the input signal `data_tlast` is applied correctly. The signal `event_frame_started` is one in a periodic manner for each started frame if the block runs correctly.

More detailed information about the block can be found in the datasheet [Xil15].

Rounding Issues

It was not possible to receive exactly the same calculation results from the Simulink and the System Generator IFFT blocks. Figure 4.9 shows in the top scope the real part of the signal from the Simulink IFFT block, while the middle scope shows the real part of the signal from the System Generator IFFT block. The trace in the bottom shows the difference signal, the results are differing in the Least Significant Bit (LSB). This is no problem for the performance of the system.

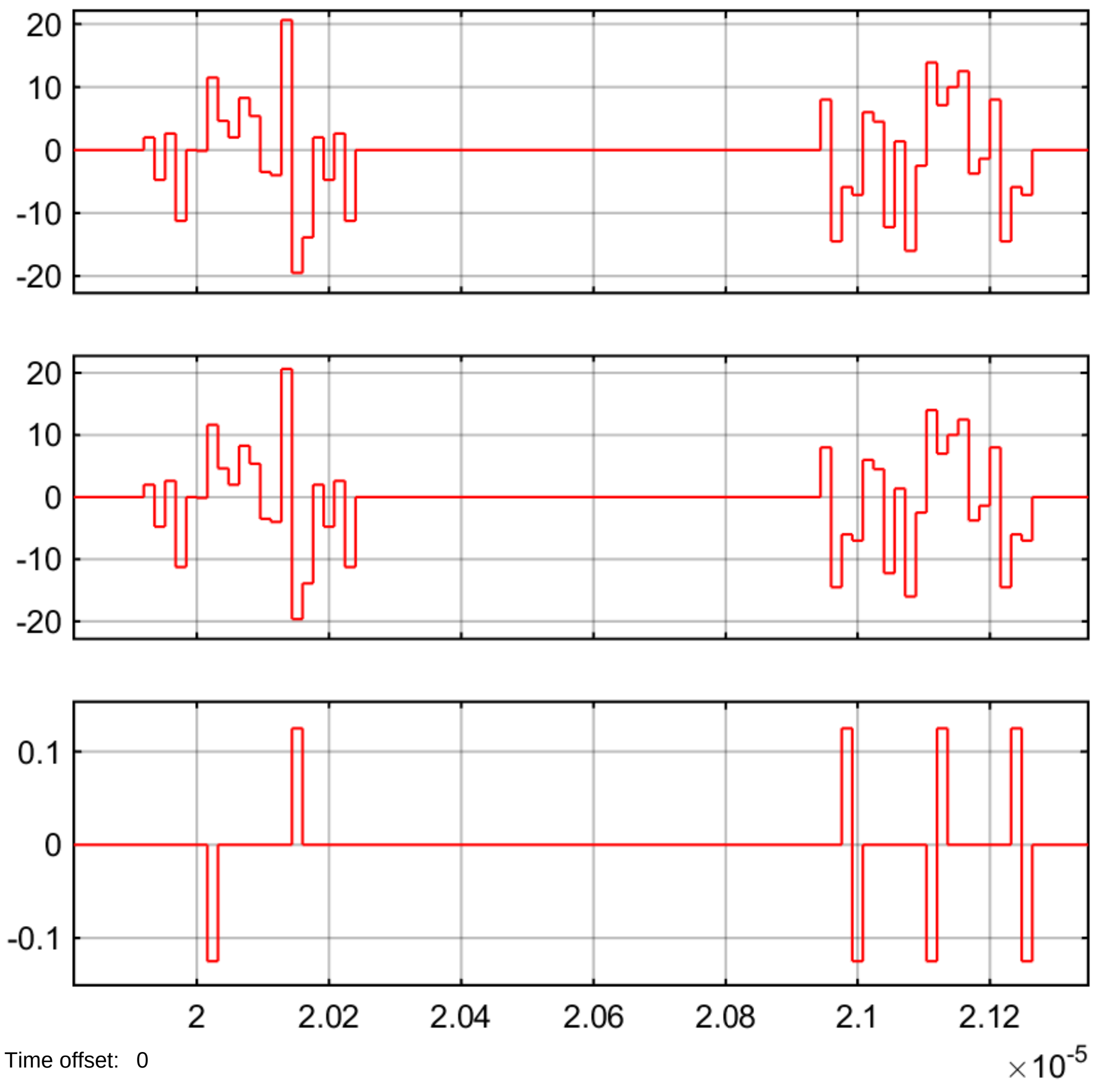


Figure 4.9.: Comparision of the Simulink and System Generator IFFT block signals.

Guard Interval

This section describes the implementation of the block `GI` of the transmitter in Figure 2.1 on page 16.

After the IFFT, the GI is applied to the serial stream which is done using a Moore FSM. The FSM reads the current frame and outputs it again with the GI inserted in front of the signal, refer to Equation 2.7 on page 19. Table 4.3 lists the input and output signals of the FSM.

Signal	Type	Description
`inp_re`	Input	Real part of input signal
`inp_im`	Input	Imaginary part of input signal
`frameEn`	Input	Starts the insertion of the GI. This is the `frameStart` signal provided by the input FSM, refer to Table 4.2
`out_re`	Output	Real part of output signal
`out_im`	Output	Imaginary part of output signal
`pop`	Output	Asserts that the FSM outputs a value

Table 4.3.: Inputs and outputs of the GI FSM.

The FSM idles in `s0` until the signal `frameEn` starts the fixed sequence of the states, refer to Figure 4.10. Expect for `s0`, each state has only one possible following state. The substitutions of the output values corresponding to the states are explained in Table 4.4.

The shown FSM is used in the OFDM mode and uses the code in `sg_fsm_GI_OFDM`. The DMT mode uses the code in `sg_fsm_GI_DMT`, all provided information are similar for DMT.

In OFDM mode $N_{chn} = N_{sc} + N_{GI} = 16 + 4 = 20$ values are created out of $N_{sc} = 16$ values using 32 states. In DMT mode, due to twice the IFFT length and to maintain the bandwidth efficiency $\beta = 0.8$, 40 values are created out of 32 using 64 states.

Identifier	Output values
A - s1	i1_re <= inp_re i1_im <= inp_im
B - s2	i2_re <= inp_re i2_im <= inp_im
C - s12	i12_re <= inp_re i12_im <= inp_im
D - s13	i13_re <= inp_re i13_im <= inp_im out_re <= inp_re out_im <= inp_im pop <= 1
E - s16	i16_re <= inp_re i16_im <= inp_im out_re <= inp_re out_im <= inp_im pop <= 1
F - s17	out_re <= i1_re out_im <= i1_im pop <= 1
G - s32	out_re <= i16_re out_im <= i16_im pop <= 1

Table 4.4.: Substitutions for the output values corresponding to the states in Figure 4.10.

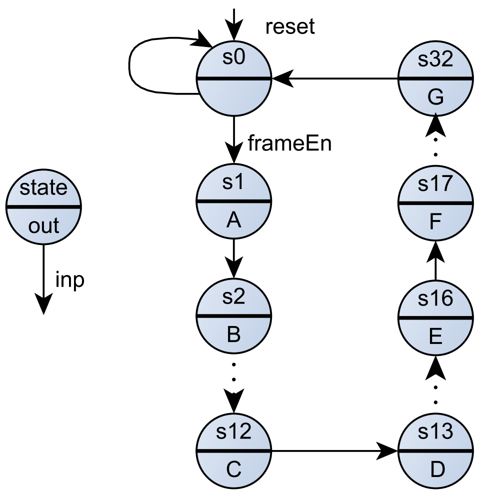


Figure 4.10.: Moore FSM for the insertion of the GI in OFDM mode. The dotted lines shall indicate that more states are in between.

Output FIFO

The data including the GI is fed into a FIFO which is controlled by a FSM, refer to Figure 4.11 for OFDM mode. This allows to output the data to the DAC with the expected rate. The upsample blocks `Up Sample1` and `Up Sample2` decrease the word period of the real- and imaginary part to $T_{clk} = 4\,\text{ns}$, doing this allows to downsample the FIFO output signals to an arbitrary period which is a multiple of T_{clk}. The downsample by 14 blocks lead to the expected rate of $4\,\text{ns} \cdot 14 = 56\,\text{ns}$ for OFDM, refer to section 4.2.

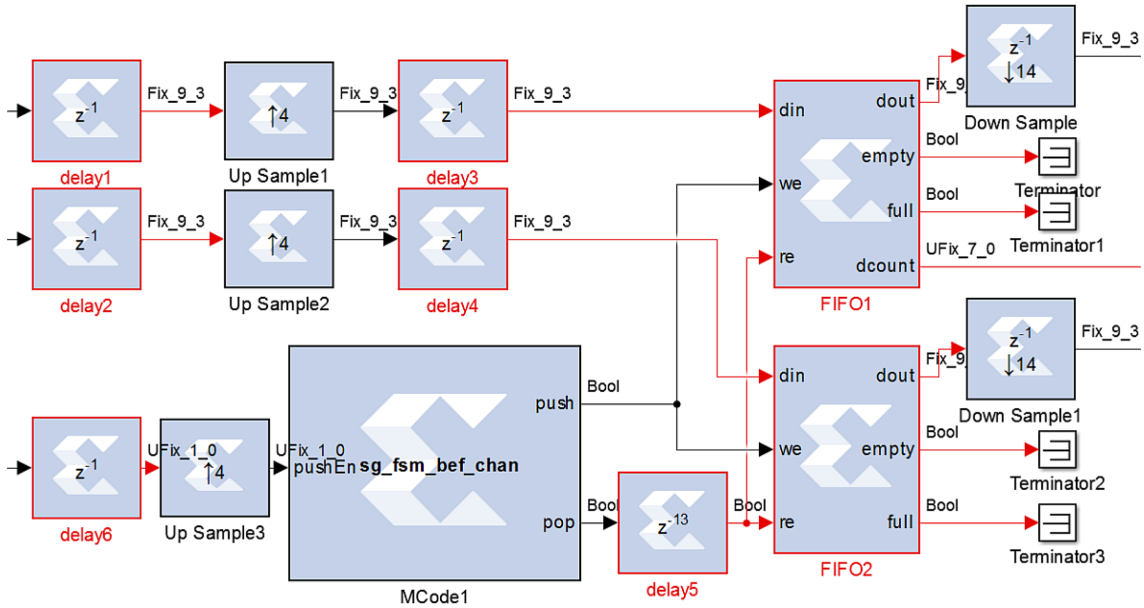


Figure 4.11.: Output FIFO for the transmitter in OFDM mode.

state	push	pop
0	1	0
1	1	1
2	0	1
3	0	0

Table 4.5.: Possible output values of the transmitter output FSM dependent of the states.

The FSM uses 4 states to control the FIFO, they are shown in Table 4.5. All signals and constants of the FSM are explained in Table 4.6. The ASM chart is shown in Figure 4.12. All internal registers are initialized to 0.

The file `sg_fsm_bef_chan.m` contains the source code for the FSM.

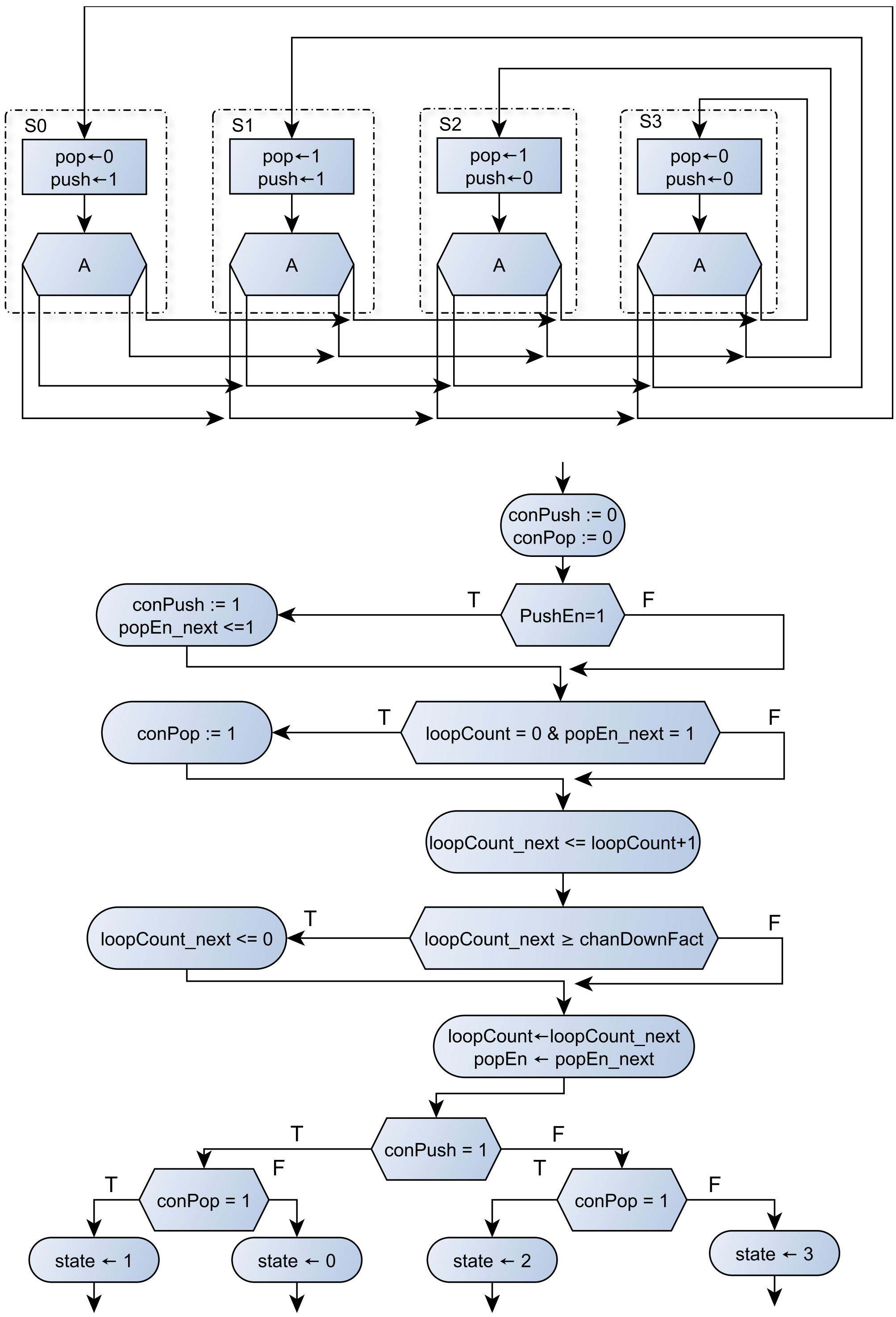


Figure 4.12.: ASM chart for the output FSM of the transmitter in OFDM mode. The lower part substitutes the block `A` in the upper part.

Signal type	Name	Description
Input	`pushEn`	Asserts that valid data is at the input port of the FIFO. This is the `pop` signal of the GI FSM
Output Moore	`pop`	Signals the FIFO to output a word
Output Moore	`push`	Signals the FIFO to insert a word
Internal register	`state`	Stores the state
Internal register	`popEn`	Used for initialization to enable the pop signals only when already a word is inserted
Internal register	`loopCount`	Counts the cycles to pop words with the expected rate
Constant	`chanDownFact`	Divisor of the expected word period of the output with the system period. Used to pop words with the expected rate
Identifier variable	`conPush`	Substitutes the logic to determine if in the next cycle the input value has to be pushed
Identifier variable	`conPop`	Substitutes the logic to determine if in the next cycle a value has to be poped

Table 4.6.: All signals of the transmitter output FSM.

4.7. Channel

This section describes the implementation of the blocks `Channel` and `sum` in Figure 2.1 on page 16.

The channel model follows after the output FIFO. It consists of an FIR-filter to test the equalizer and an AWGN channel model to measure the BER. Because these measurements are done in Simulink but not on the hardware, the channel models are only implemented in Simulink blocks and used to test both implementations.

FIR-Filter

The FIR-filter channel model uses the `Discrete FIR Filter` block, which allows to insert the filter coefficients as a row vector. The implemented filter is a lowpass with the order 2 and, therefore, a group delay of one sample. The used filter coefficients are $a0 = 0.211949085954037 03$, $a1 = 0.576101828091926$ and $a2 = a0$. It is created with the Filter Design and Analysis Tool in Simulink using the window method.

Figure 4.13 shows the magnitude response of the filter. The created magnitude response shall simulate the lowpass behaviour of a wire.

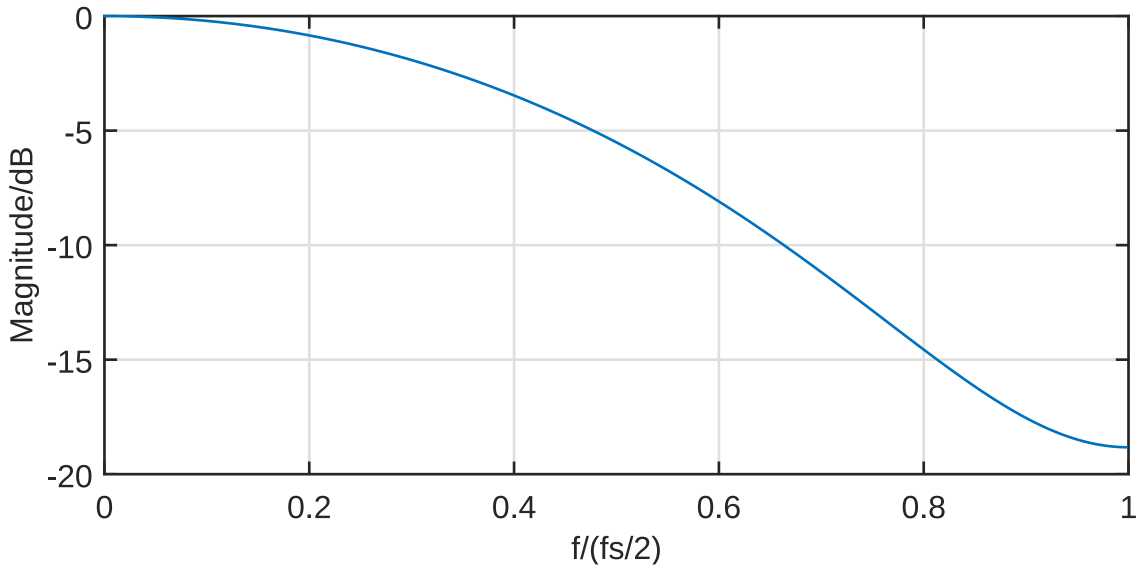


Figure 4.13.: The magnitude response of the FIR-filter channel model.

AWGN

The AWGN channel model uses the `AWGN Channel` block. The block works only with double values, therefore, the values have to be converted to double first when working with fixed-point numbers. After applying the noise, they are converted back to fixed-point format. The block expects the input signal power as a parameter which is calculated for OFDM in Equation 3.9 on page 34 and for DMT in Equation 3.10.

4.8. Receiver

As shown in Figure 2.1 on page 16, the receiver is mainly build like the transmitter but in reverse order. Hence, only the blocks differing from the ones in the transmitter will be shown here. The rest will be mentioned shortly.

Guard Interval Removal

The first stage of the receiver is a FIFO with a FSM which generates the control signals. The data is read into the FIFO first to allow the computation at an arbitrary rate. This section describes the implementation of the crossed `GI` block within the receiver in Figure 2.1 on page 16. It removes the GI by another FSM, refer to Figure 4.14.

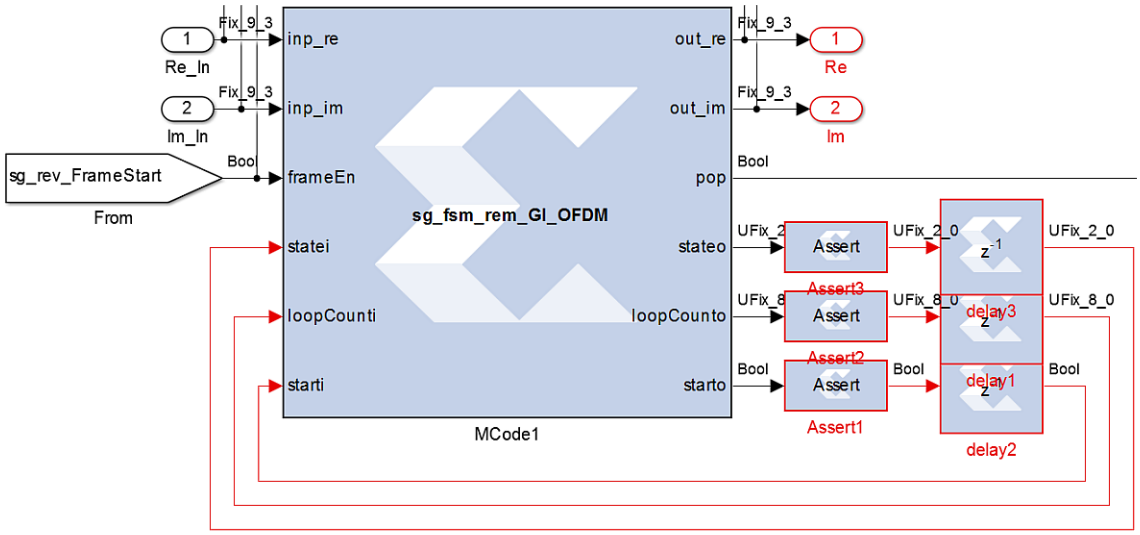


Figure 4.14.: The FSM to remove the GI.

Signal type	Name	Description
Input	`inp_re`	Real part of input signal
Input	`inp_im`	Imaginary part of input signal
Input	`frameEn`	Asserts that a new frame is starting. The Signal is generated by the FSM of the input FIFO
Output Moore	`out_re`	Real part of output signal
Output Moore	`out_im`	Imaginary part of output signal
Output Moore	`pop`	Asserts that a word is outputted
Register	`state`	Register for storing the state
Register	`loopCount`	Register for counting how many values have been outputted in the current frame
Register	`start`	Register for storing the information that actually a frame is processed
Constant	`GI_Active`	Number of samples of the GI in current mode
Constant	`fftLenActive`	Number of samples of the FFT in current mode

Table 4.7.: All signals of the FSM to remove the GI.

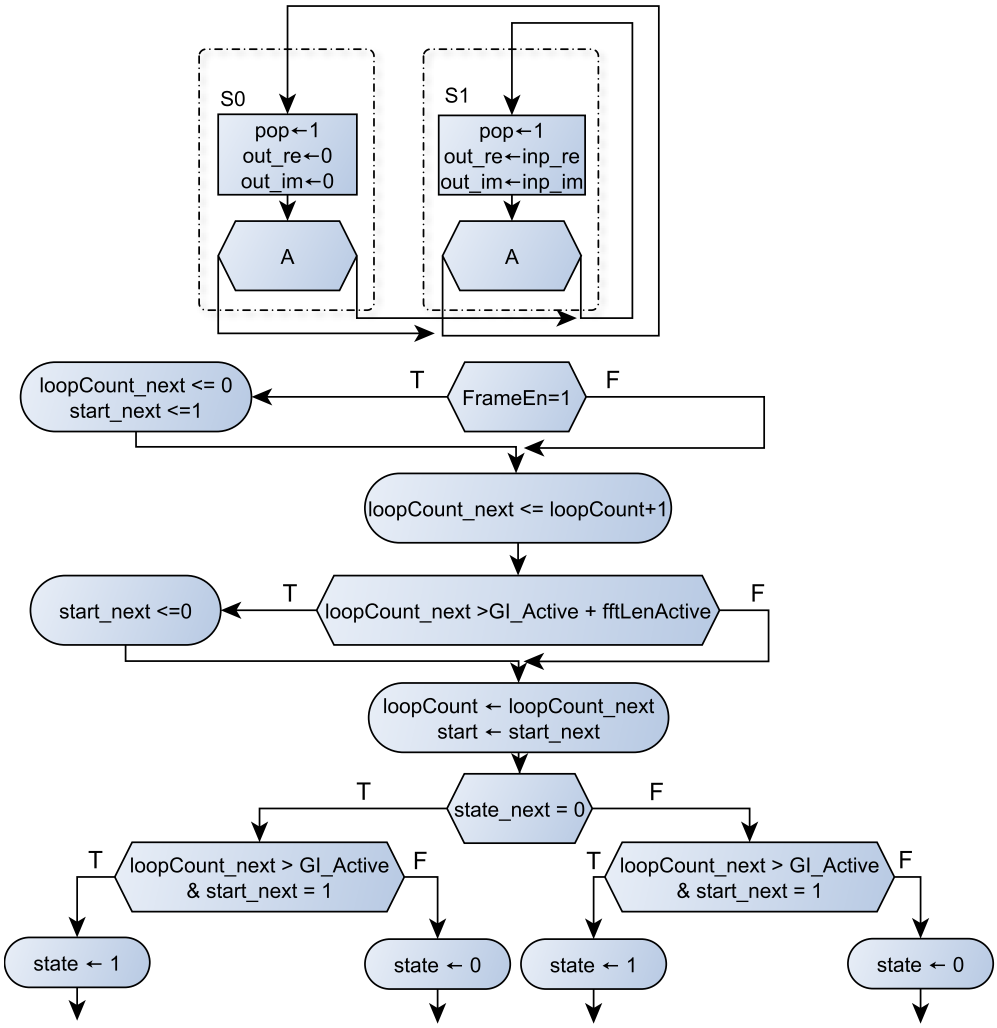


Figure 4.15.: ASM chart for the FSM which removes the GI. The lower part substitutes the block `A` in the upper part.

To remove the GI samples, the FSM consists of two states:

- Pass the input to the output
- Output zeros

The algorithm uses the signals in Table 4.7 and is shown in ASM chart 4.15. The internal registers are modelled differently than in the previous FSMs, this is done due to pipelining issues which will be described in detail in section 5.3. Each register is modelled as an input and an output with a delay for saving the value. The code for the FSM is in the file `sg_fsm_rem_GI_OFDM.m`.

Equalizer

After removing the GI the signal is run through the FFT which works analogue to the IFFT in the transmitter. The next step is to parallelize the data again to distribute it to the demodulators of the specific subcarriers. This is done with an `MCode` block and the source `Extract_Subcar_16.m` and describes the `S/P` and `Equalizer` blocks in Figure 2.1 on page 16.

The command `xl_slice()` [Xil14d, S.208-213] is used to extract the specific subcarriers. `xl_slice()` returns unsigned fixed point values which have to be reinterpreted to get the decimal comma at the expected place, this is done using `xl_force()`.

If the equalizer is activated the code is replaced with `Extract_Subcar_16_Equal.m` and in addition to extracting the subcarriers, the equalizer is applied in this step. The coefficients are prepared in the `initScript.m` script. The vector with coefficients is zero padded to the length of the FFT to achieve the expected number of coefficients, refer to Listing 4.1.

```
h=[a0; a1; a2;];

H=fft(h,fftLenActive);%ofdm16 dmt32
if dmtOfdm == 2 %DMT active
    H = H(1:16);
end

e=H;
e1=e(1);
e2=e(2);
...
e16=e(16);

%Prepare coefficients for sysgen
e1s=1/e1;
e2s=1/e2;
...
e16s=1/e16;
```

Listing 4.1: Excerpt of `initScript.m` which shows how the equalizer coefficients are calculated.

As shown in Equation 2.11 on page 22, each received value has to be divided by the coefficient. The `MCode` block allows only divisions by values which are a power of 2. Therefore, the inverse of each coefficient is precalculated within the script to allow the calculation by a multiplication in hardware. However, as soon as an channel estimation algorithm is used, this section has to be reevaluated, because the coefficients cannot be precalculated.

In the source code for the `MCode` block the coefficients have to be cast with the `xfix()` command to fixed-point values because, coming out of Matlab, they are double values. After calculating the multiplication the values are cast back to the initial `s5Q3` format, refer to Listing 4.2 for the code.

```
%Extract subcarriers
x1_re = xl_slice(x_re, bitCountAftFFTRec*16-1, bitCountAftFFTRec*15);
x2_re = xl_slice(x_re, bitCountAftFFTRec*15-1, bitCountAftFFTRec*14);
...
x16_re = xl_slice(x_re, bitCountAftFFTRec-1   , 0  );

x1_im = xl_slice(x_im, bitCountAftFFTRec*16-1, bitCountAftFFTRec*15);
x2_im = xl_slice(x_im, bitCountAftFFTRec*15-1, bitCountAftFFTRec*14);
...
x16_im = xl_slice(x_im, bitCountAftFFTRec-1   , 0  );

%Reinterpret to expected Q format
tmp_y1_re = xl_force(x1_re,xlSigned,fracLen);
tmp_y2_re = xl_force(x2_re,xlSigned,fracLen);
...
tmp_y16_re = xl_force(x16_re,xlSigned,fracLen);

tmp_y1_im = xl_force(x1_im,xlSigned,fracLen);
tmp_y2_im = xl_force(x2_im,xlSigned,fracLen);
...
tmp_y16_im = xl_force(x16_im,xlSigned,fracLen);

%Convert coefficients from double to fixed point
e1_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, e1_re);
e2_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, e2_re);
...
e16_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, e16_re);
```

```
e1_im = xfix({xlSigned, bitCountAftFFTRec,fracLen}, e1_im);
e2_im = xfix({xlSigned, bitCountAftFFTRec,fracLen}, e2_im);
...
e16_im = xfix({xlSigned,bitCountAftFFTRec,fracLen}, e16_im);

%Apply coefficients (complex multiplication)
y1_re = tmp_y1_re*e1_re- tmp_y1_im*e1_im;
y2_re = tmp_y2_re*e2_re- tmp_y2_im*e2_im;
...
y16_re = tmp_y16_re*e16_re- tmp_y16_im*e16_im;

y1_im = tmp_y1_re*e1_im+ tmp_y1_im*e1_re;
y2_im = tmp_y2_re*e2_im+ tmp_y2_im*e2_re;
...
y16_im = tmp_y16_re*e16_im+ tmp_y16_im*e16_re;

%Cast to output Q format
y1_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y1_re);
y2_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y2_re);
...
y16_re = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y16_re);

y1_im = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y1_im);
y2_im = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y2_im);
...
y16_im = xfix({xlSigned, bitCountAftFFTRec,fracLen}, y16_im);
```

Listing 4.2: Code excerpt for the extraction of subcarriers and the equalizer.

Demodulator

This section describes the crossed modulator blocks in the receiver in Figure 2.1 on page 16.

The extracted subcarriers are fed into a `MCode` block for each subcarrier, running the code in `Demod_16QAM.m`, refer to listing 4.3. The code decides the bits using if statements and concatenates them to a 4 bit output word.

```
function [y1]= Demod_16QAM(I,Q)
%Some ideas from UG958
 one  = xfix({xlUnsigned,1,0}, 1);
```

```
zero = xfix({xlUnsigned,1,0}, 0);

 if (I >= 0)
        Y0 = one;
        if (I > 2)
            Y1 = zero;
        else
            Y1 = one;
        end;
    else
        Y0 = zero;
        if (I > -2)
            Y1 = one;
        else
            Y1 = zero;
        end;
 end;
    if (Q >= 0)
        Y2 = zero;
        if (Q > 2)
            Y3 = zero;
        else
            Y3 = one;
        end;
    else
        Y2 = one;
        if (Q > -2)
            Y3 = one;
        else
            Y3 = zero;
        end;
    end;

    y1 = xl_concat(Y0, Y1, Y2, Y3);
  end
```

Listing 4.3: Demodulator for 16QAM.

After the demodulator, the parallel streams are serialized again which reduces the period to $T_{clk} = 4\,\text{ns}$. The data is fed into a FIFO, controlled by a FSM, similar to the output FIFO of the transmitter. This allows to output the data at an arbitrary rate which is used to restore the input period of $T_b = 20\,\mathrm{ns}$.

4.9. Bit Error Ratio

For measuring the performance of the system a block which calculates the BER is created. This measurements are done within Simulink, therefore, the block is only created with Simulink blocks and the code is shown in Listing 4.4.

The input signals are the received bit `recv` and the sent bit `sent`, delayed by the transmission delay. An additional reset input `rst` sets all counters to zero again. The variable `sent_r` counts the sent bits while `error_r` counts the errors. The ratio between these is calculated for each invocation and saved to `ber_r`.

```
function [sent_out,error,ber] = ber_calc (  sent, recv, rst)

global ber_r;
global sent_r;
global error_r;

sent_r=sent_r+1;

if sent ~= recv
    error_r = error_r+1;
end
ber_r = error_r/sent_r;

if rst == 1
    ber_r=0;
    sent_r=0;
    error_r=0;
end

sent_out = sent_r;
error = error_r;
ber = ber_r;
end
```

Listing 4.4: Matlab code for calculating the BER.

5. Synthesis for the FPGA

This chapter describes the synthesis and implementation of the System Generator blocks from Simulink on the FPGA. It discusses the occurring timing issues and how they are handled. Furthermore, the final synthesis and implementation result are examined and the hardware resource utilization for several configuration modes is compared.

5.1. Block Diagram

This section describes how the created design is added to a Vivado block diagram. In the `System Generator` block, inside the Simulink model, the compilation type `IP Catalog` is used. This allows to add the generated netlist as a block to the *IP Catalog* dialogue in Vivado. After creating a `Block Design` in Vivado the block can be added.

The transmitter and receiver are combined in the block `basic_dmt`. To test the design with data, the random bit generator is added as an additional IP Catalog block. The `ZYNQ7 Processing System` block is added too, used as a clock source for the needed clock signal with the period $T_{clk} = 4ns$, refer to Figure 5.1.

The output signal of the random bit generator is connected to the transmitter input port `inp_trans`, in addition it is also output to port `X12`. When the data leaves the transmitter by `out_trans_re` and `out_trans_im` it is looped back to the receiver into the ports `inp_recv_re` and `inp_recv_im`. Using a real channel, the ports have to be connected to output pins, which are connected to the transceiver hardware.

The regenerated bits inside the receiver are routed through port `out_recv` to pin `X10`. Port `diffsig` outputs the `XOR` operation of the `out_recv` and the delayed `inp_trans` signal to check for bit errors. The resulting signal is forwarded to port `X5` and can be

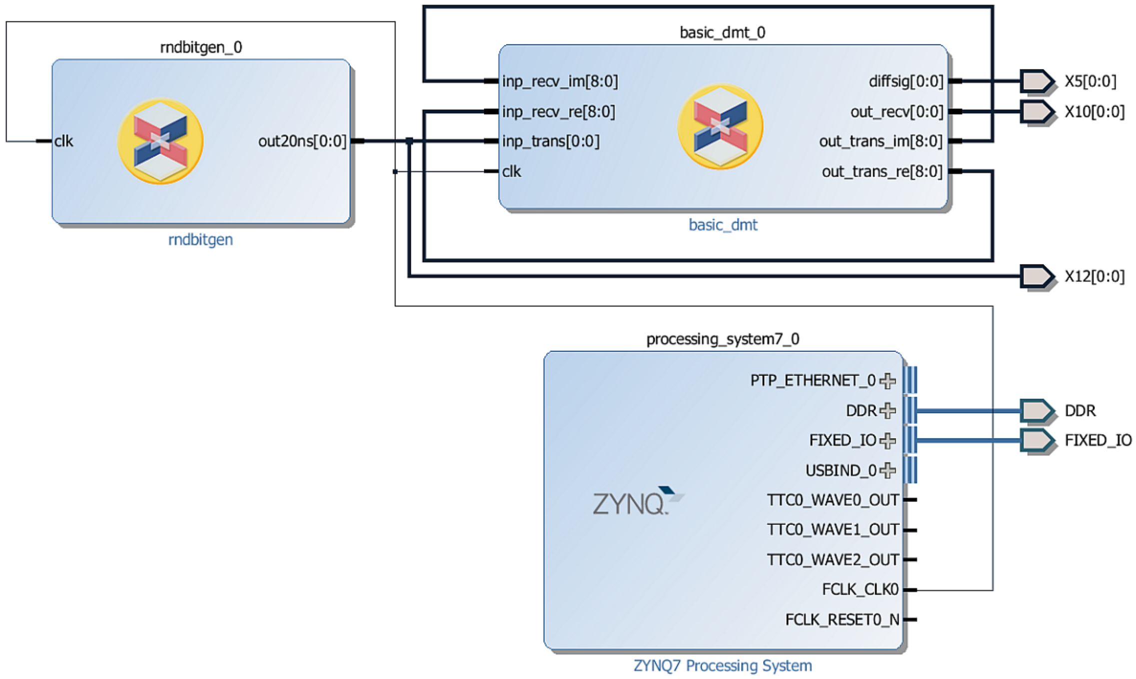


Figure 5.1.: The block diagram of the system in Vivado.

measured with an oscilloscope. Table 5.1 shows the Pins with their site on the Ball Grid Array (BGA) of the FPGA, the I/O standard of all pins is `LVCMOS33` [JED].

Signal Port	Signal Name	Site
`X5`	`diffsig`	`V15`
`X10`	`out_recv`	`L16`
`X12`	`inp_trans`	`T10`

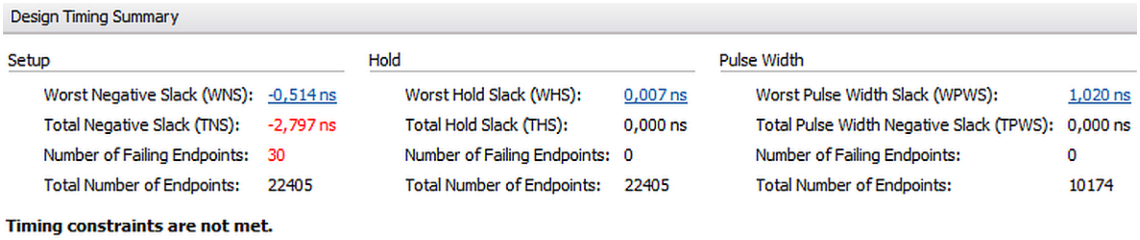
Table 5.1.: The pin list for the output signals of the FPGA

The identifiers in the column *Signal Port* are chosen corresponding to the identifiers of the SMB connectors on the Connection Extension Board, refer to Figure A.3 on page 85. The routing of the signal lines is done in pairs of two to allow the use of differential signalling. For single ended use this causes crosstalk, therefore, the connectors with the maximum distance to each other are chosen.

For proper operation of the processing system the `DDR` and `FIXED_IO` pins have to be connected. This can be done by the connection automation wizard.

5.2. Static Timing Analysis

The fact that some parts of the system are running with the system clock period of $T_{clk} = 4ns$ induces rough timing requirements. First tries to synthesize and implement the model showed that the timing failed. Even trying all the possible permutations of the integrated synthesis and implementation strategies of Vivado could not fix the problems. Figure 5.2 shows an example for 30 failing endpoints. Therefore, the timing is failing for 30 registers.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0,514 ns	Worst Hold Slack (WHS):	0,007 ns	Worst Pulse Width Slack (WPWS):	1,020 ns
Total Negative Slack (TNS):	-2,797 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	30	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	22405	Total Number of Endpoints:	22405	Total Number of Endpoints:	10174

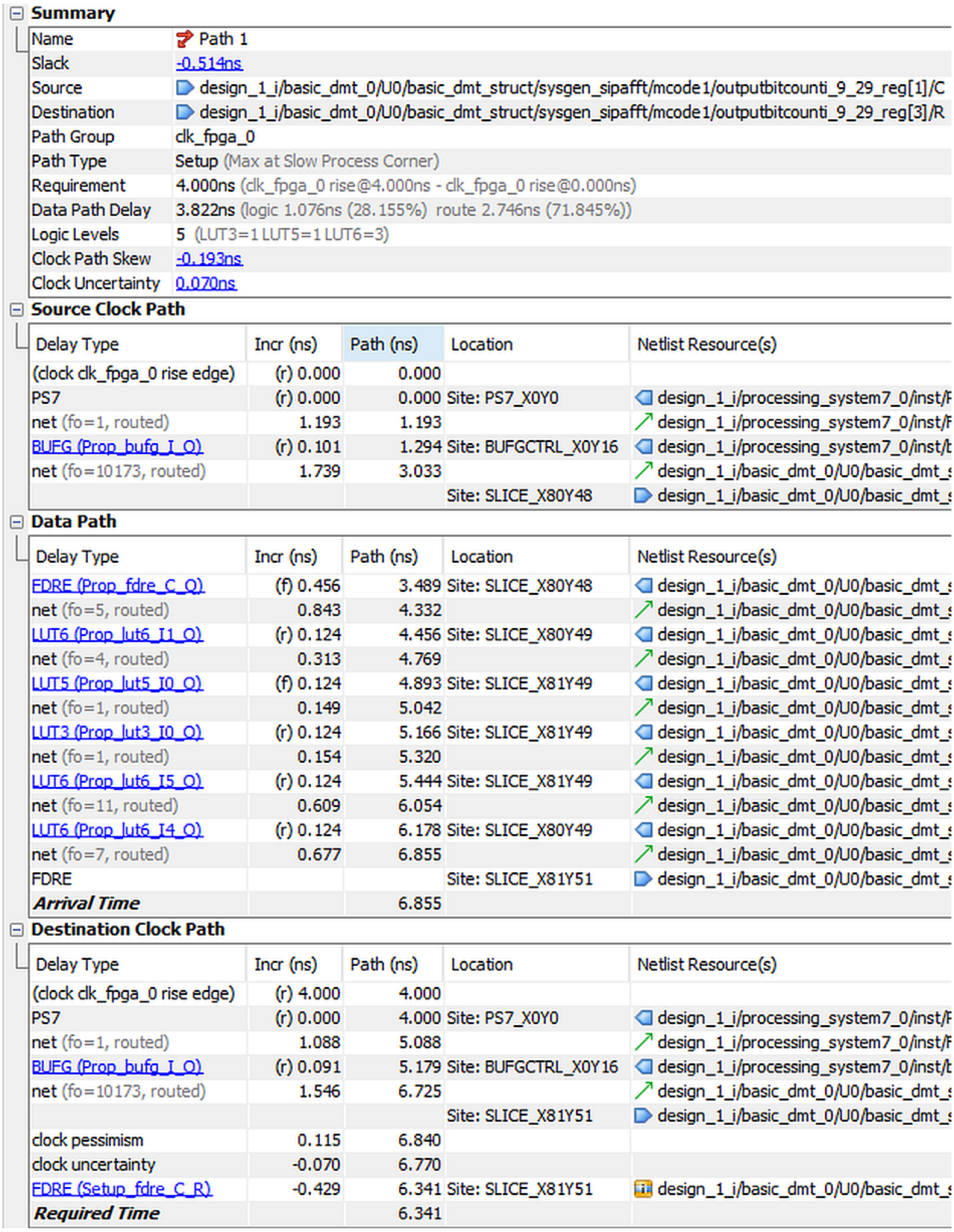
Timing constraints are not met.

Figure 5.2.: An example with 30 failing endpoints.

The picture shows that in all cases the setup time is violated while the hold time is fine. The setup time describes the time interval meanwhile the signal at the input of an register is not allowed to change *before* the clock edge [Pon06, S.216]. The hold time is the time interval meanwhile the signal at the input of the register is not allowed to change *after* the clock edge. If one of these times is violated, the register enters a metastable state where it can randomly output `zero` or `one`.

Figure 5.3 shows the details for one of the failing endpoints, they are described in the following list:

- **Slack**: The path is failing between the source and destination register because the data path takes 0.514 ns to long, this is called slack.
- **Logic Levels**: Describes the number of Look Up Tables (LUTs) which are between both registers, in this case 5.
- **Clock Path Skew**: The clock skew describes the difference of the arrival time of the clock at both registers. This time has to be subtracted from the overall time budget [Pon06, S.606-610].
- **Source Clock Path**: This paragraph describes the time needed by the clock to reach the first register. The sum of the delays is $T_{scp} = 3.033$ ns.



Summary

Name	Path 1
Slack	-0.514ns
Source	design_1_i/basic_dmt_0/U0/basic_dmt_struct/sysgen_sipafft/mcode1/outputbitcounti_9_29_reg[1]/C
Destination	design_1_i/basic_dmt_0/U0/basic_dmt_struct/sysgen_sipafft/mcode1/outputbitcounti_9_29_reg[3]/R
Path Group	clk_fpga_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	4.000ns (clk_fpga_0 rise@4.000ns - clk_fpga_0 rise@0.000ns)
Data Path Delay	3.822ns (logic 1.076ns (28.155%) route 2.746ns (71.845%))
Logic Levels	5 (LUT3=1 LUT5=1 LUT6=3)
Clock Path Skew	-0.193ns
Clock Uncertainty	0.070ns

Source Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_fpga_0 rise edge)	(r) 0.000	0.000		
PS7	(r) 0.000	0.000	Site: PS7_X0Y0	design_1_i/processing_system7_0/inst/F
net (fo=1, routed)	1.193	1.193		design_1_i/processing_system7_0/inst/F
BUFG (Prop_bufg_I_O)	(r) 0.101	1.294	Site: BUFGCTRL_X0Y16	design_1_i/processing_system7_0/inst/t
net (fo=10173, routed)	1.739	3.033		design_1_i/basic_dmt_0/U0/basic_dmt_s
			Site: SLICE_X80Y48	design_1_i/basic_dmt_0/U0/basic_dmt_s

Data Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
FDRE (Prop_fdre_C_Q)	(f) 0.456	3.489	Site: SLICE_X80Y48	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=5, routed)	0.843	4.332		design_1_i/basic_dmt_0/U0/basic_dmt_s
LUT6 (Prop_lut6_I1_O)	(r) 0.124	4.456	Site: SLICE_X80Y49	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=4, routed)	0.313	4.769		design_1_i/basic_dmt_0/U0/basic_dmt_s
LUT5 (Prop_lut5_I0_O)	(f) 0.124	4.893	Site: SLICE_X81Y49	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=1, routed)	0.149	5.042		design_1_i/basic_dmt_0/U0/basic_dmt_s
LUT3 (Prop_lut3_I0_O)	(r) 0.124	5.166	Site: SLICE_X81Y49	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=1, routed)	0.154	5.320		design_1_i/basic_dmt_0/U0/basic_dmt_s
LUT6 (Prop_lut6_I5_O)	(r) 0.124	5.444	Site: SLICE_X81Y49	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=11, routed)	0.609	6.054		design_1_i/basic_dmt_0/U0/basic_dmt_s
LUT6 (Prop_lut6_I4_O)	(r) 0.124	6.178	Site: SLICE_X80Y49	design_1_i/basic_dmt_0/U0/basic_dmt_s
net (fo=7, routed)	0.677	6.855		design_1_i/basic_dmt_0/U0/basic_dmt_s
FDRE			Site: SLICE_X81Y51	design_1_i/basic_dmt_0/U0/basic_dmt_s
Arrival Time		6.855		

Destination Clock Path

Delay Type	Incr (ns)	Path (ns)	Location	Netlist Resource(s)
(clock clk_fpga_0 rise edge)	(r) 4.000	4.000		
PS7	(r) 0.000	4.000	Site: PS7_X0Y0	design_1_i/processing_system7_0/inst/F
net (fo=1, routed)	1.088	5.088		design_1_i/processing_system7_0/inst/F
BUFG (Prop_bufg_I_O)	(r) 0.091	5.179	Site: BUFGCTRL_X0Y16	design_1_i/processing_system7_0/inst/t
net (fo=10173, routed)	1.546	6.725		design_1_i/basic_dmt_0/U0/basic_dmt_s
			Site: SLICE_X81Y51	design_1_i/basic_dmt_0/U0/basic_dmt_s
clock pessimism	0.115	6.840		
clock uncertainty	-0.070	6.770		
FDRE (Setup_fdre_C_R)	-0.429	6.341	Site: SLICE_X81Y51	design_1_i/basic_dmt_0/U0/basic_dmt_s
Required Time		6.341		

Figure 5.3.: An example of the detailed information of an failing endpoint.

- **Data Path**: After the clock reaches the register the output value is delayed by the 5 LUTs and reaches the second register after $T_d = T_{scp} + T_{dp} = 3.033\,\text{ns} + 3.822\,\text{ns} = 6.855\,\text{ns}$.

- **Destination Clock Path**: The following clock edge starts with a delay of 4 ns, related to the first edge and reaches the second register at $T_{dcp} = 6.341\,\text{ns}$.

Because T_d is higher then T_{dcp} the slack s is negative and the setup time is violated $s = T_{dcp} - T_d = 6.341\,\text{ns} - 6.855\,\text{ns} = -0.514\,\text{ns}$. This cannot be fixed by increasing $T_{dcp} = 6.341\,\text{ns}$, for example through additional wire delay, this could cause a hold time violation [VLS].

The row **Data Path Delay** lists that 28.55% are caused by the logic while 71.845% are induced through the routing. However, since the Vivado algorithms are highly optimized the easiest way to fix the violations is due pipelining.

More detailed information about the timing analysis can be found in [Xil14c, S.99-107]

5.3. Pipelining

Pipelining is the insertion of additional registers within the data path to reduce the data path delay between the registers. The way to do this, is to look at the path which causes the biggest problems, therefore, the highest negative slack. The **source** and **destination** rows show these registers, refer to Figure 5.3. The corresponding block has to be identified in the Simulink System Generator model and an additional delay has to be added. Often multiple violations are caused by the same block, hence, they are fixed with a single additional delay. This procedure is repeated for all failing endpoints.

Afterwards the model is synthesized and implemented with all strategy permutations again, detailed information about the strategies can be found in [Xil14b, S.151-154]. If new violations show up or old ones are still not fixed the procedure has to be repeated.

The problematic paths are the FSMs, the FFT blocks and the demodulator blocks.

Figure 5.4 shows an excerpt of the Vivado design runs window which shows all the

Name	WNS	TNS	WHS	THS	Strategy
impl_12	-0.52	-5.79	0.02	0.00	Performance_ExploreSLLs (Vivado Implementation 2014)
impl_13	-0.31	-2.44	0.02	0.00	Performance_Retiming (Vivado Implementation 2014)
impl_14	0.13	0.00	0.02	0.00	Area_Explore (Vivado Implementation 2014)
impl_15	-0.35	-3.58	0.02	0.00	Power_DefaultOpt (Vivado Implementation 2014)
impl_16	-0.22	-1.54	0.02	0.00	Flow_RunPhysOpt (Vivado Implementation 2014)
impl_17	-0.16	-1.07	0.02	0.00	Flow_RunPostRoutePhysOpt (Vivado Implementation 2014)
impl_18	-0.69	-17.39	0.02	0.00	Flow_RuntimeOptimized (Vivado Implementation 2014)
impl_19	-6.78	-10397.12	-0.16	-1.67	Flow_Quick (Vivado Implementation 2014)
impl_20	-0.44	-3.14	0.02	0.00	Congestion_SpreadLogic_high (Vivado Implementation 2014)
impl_21	-0.40	-24.03	0.02	0.00	Congestion_SpreadLogic_medium (Vivado Implementation 2014)
impl_22	-0.40	-2.79	0.02	0.00	Congestion_SpreadLogic_low (Vivado Implementation 2014)
impl_23	-0.44	-2.20	0.05	0.00	Congestion_SpreadLogicSLLs (Vivado Implementation 2014)
impl_24	-0.52	-5.79	0.02	0.00	Congestion_BalanceSLLs (Vivado Implementation 2014)
impl_25	-0.20	-1.94	0.02	0.00	Congestion_BalanceSLRs (Vivado Implementation 2014)
impl_26	-0.20	-1.94	0.02	0.00	Congestion_CompressSLRs (Vivado Implementation 2014)
synth_2					Vivado Synthesis Defaults (Vivado Synthesis 2014)
impl_27	0.16	0.00	0.03	0.00	Vivado Implementation Defaults (Vivado Implementation 2014)
impl_28	0.18	0.00	0.03	0.00	Performance_Explore (Vivado Implementation 2014)
impl_29	0.18	0.00	0.03	0.00	Performance_ExplorePostRoutePhysOpt (Vivado Implementation 2014)
impl_30	0.09	0.00	0.03	0.00	Performance_RefinePlacement (Vivado Implementation 2014)
impl_31	0.27	0.00	0.03	0.00	Performance_WLBlockPlacement (Vivado Implementation 2014)
impl_32	0.22	0.00	0.03	0.00	Performance_WLBlockPlacementFanoutOpt (Vivado Implementation 2014)
impl_33	0.18	0.00	0.03	0.00	Performance_LateBlockPlacement (Vivado Implementation 2014)

Figure 5.4.: An excerpt of the design runs window including successful and failing implementations.

synthesis and implementation permutations and is further described in the following list:

- **Name**: Shows the name of the run. Each synthesis is implemented by multiple implementation strategies.
- **WNS**: Contains the worst negative slack of the implementation. A negative value implies a failed setup time violation, therefore, the best implementation is the one with the highest positive value.
- **TNS**: Stands for the total negative slack of the implementation. It is the sum of all paths with a negative slack. This value implies how far an implementation is away from meeting the timing constraints.
- **WHS**: Contains the worst hold slack of the implementation. A negative value implies a hold time violation.
- **THS**: Stands for the total hold slack of the implementation. It is the sum of all paths with a negative hold slack.

The best variant to chose from the picture is `impl_31` because it has the highest positive WNS.

5.4. Resource Utilization

With the selection of an implemented design, the FPGA resource utilization can be viewed. Table 5.2 compares the utilization for OFDM- and DMT mode. Furthermore, it is checked how activating the equalizer affects the resource utilization.

Configuration	LUTs	Register	DSP
OFDM	2082	4980	22
DMT	3104	7789	32
OFDM + equalizer	3058	4980	22
DMT + equalizer	4168	7789	32

Table 5.2.: FPGA resource utilization for several configurations.

The table shows that the equalizer increases the amount of needed LUTs by about 30%, the number of Digital Signal Processing (DSP) slices and registers is unaffected. Choosing DMT instead of OFDM increases the necessary amount of LUTs, DSP slices and registers by about 30%. This difference originates from the additional FFT stage in the receiver and transmitter. Figure A.4 on page 86 shows the graphical representation of the implementation of the DMT mode with equalizer.

6. Results

This chapter describes how the function of the design is verified. It is shown that the design has the expected signal properties of an OFDM system. Furthermore, it is checked that the system is working as expected while running on the hardware.

6.1. OFDM Spectrum

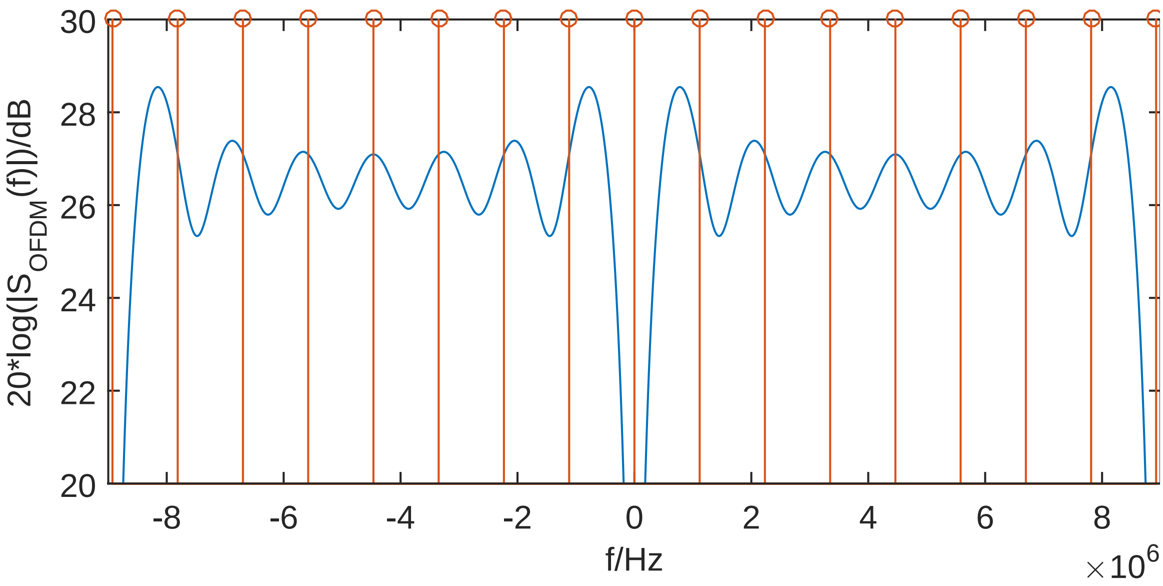
The use of the GI destroys the orthogonality of the OFDM signal. To show this circumstance, Figure 6.1a shows the spectrum measured on the output of the transmitter without GI and Figure 6.1b shows it including the GI. The pictures are taken zoomed in. The red lines show the positions of the subcarriers. To achieves this pictures, no random data was sent, instead a series of `ones` is transmitted. Therefore, each subcarrier transmits the same symbol and should have the same value respectively.

Consider that without GI, the intersection point of the spectrum with the red lines is at the same value, hence, each subcarrier has the same value. While the GI is included, the orthogonality is broken and the intersection points vary. After removing the GI the orthogonality is restored.

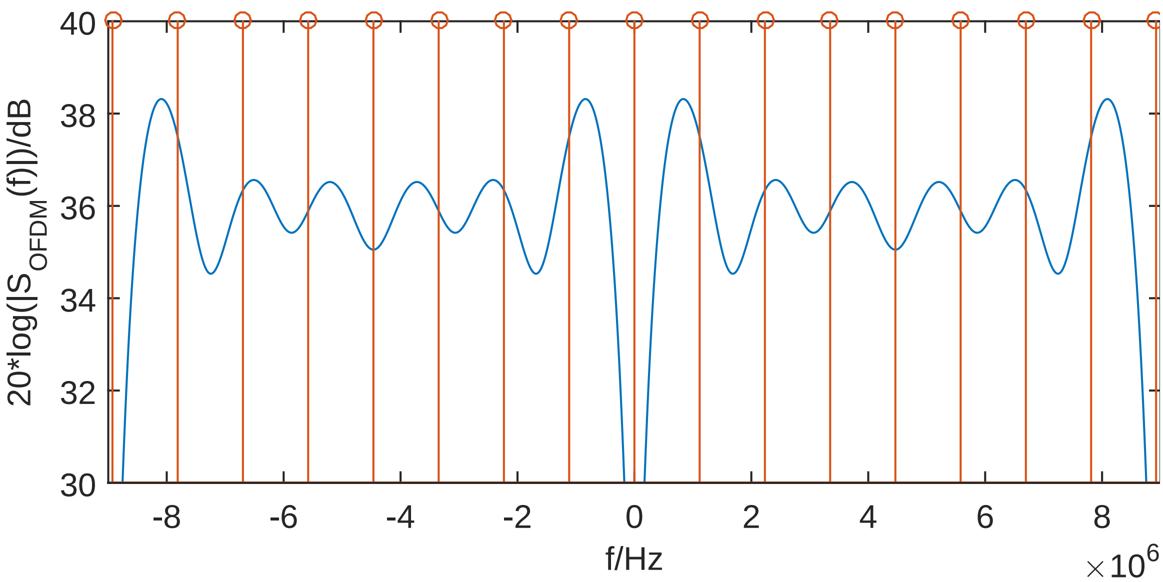
6.2. Equalizer

The equalizer can be tested using the FIR-channel model. Figure 6.2a shows the constellation diagram without the equalizer and Figure 6.2b shows it while active. Without the equalizer, the symbols are spread over a large area of the complex plane. As long as the equalizer is active, the original signal is restored.

The values in the centre at $0 + i0$ are not evaluated, they are produced in the idle times while no valid data is processed. The AWGN channel is deactivated during this measurements and double precision is used.

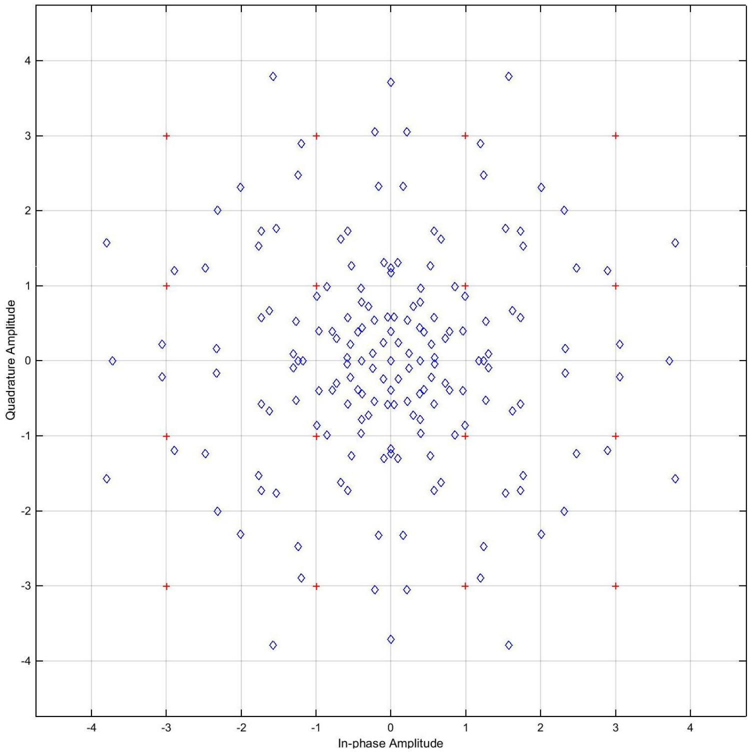


(a) Spectrum without GI.

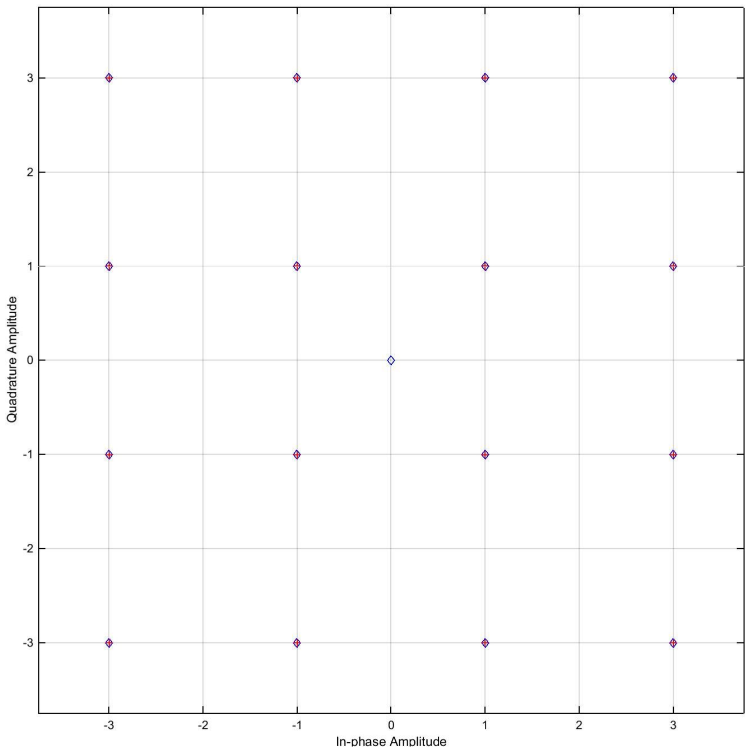


(b) Spectrum including GI.

Figure 6.1.: Comparision of the spectrum on the channel.



(a) Constellation diagram without equalizer.



(b) Constellation diagram with equalizer.

Figure 6.2.: The performance of the equalizer.

6.3. Functional Test

It has to be verified that the system is working as expected. This shall be checked for the Simulink blocks and the System Generator blocks. Therefore, it has to be checked if the bits sent into the transmitter are received at the end of the receiver and if there are differences between the signals of both variants.

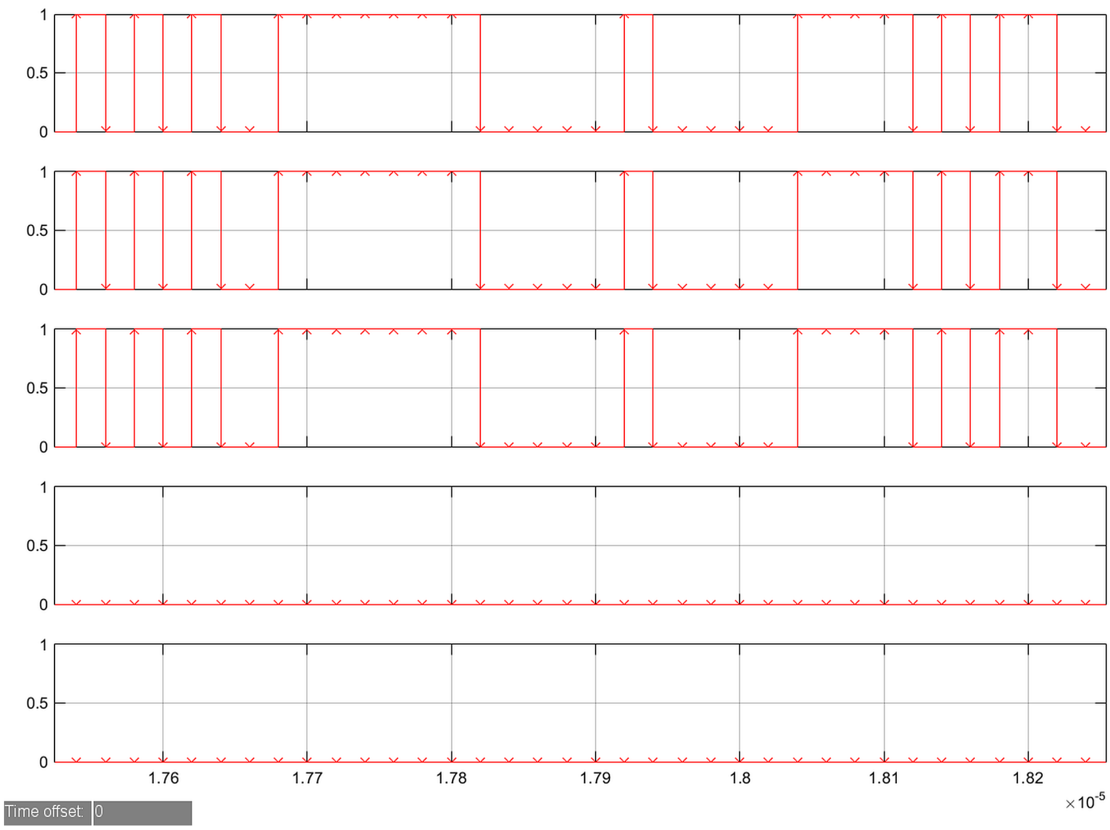


Figure 6.3.: Comparison of the received signals with the sent signal.

The sent bit stream is delayed respectively to the transmission delay and fed into a `XOR` block together with the received bits. The output is `one` each time the signals are different. Figure 6.3 shows:

- Row1: The delayed sent signal.
- Row2: The received signal of the Simulink blocks.
- Row3: The received signal of the System Generator blocks.

- Row4: The output of the `XOR` block for the Simulink blocks.
- Row5: The output of the `XOR` block for the System Generator blocks.

Hence, the system is working as expected and behaves the same for Simulink and System Generator.

Because of the differences in the rounding in the FFT and IFFT blocks, sometimes the Simulink variant has a bit error while the System Generator variant has none. This happens in both directions, hence, in average the BER is the same.

6.4. Testbench

Inside the `System Generator` block, it is possible to activate the generation of a testbench while compiling the netlist for synthesis in Vivado. Using this option, the input stimuli at the `Gateway In` blocks are saved. The calculated values leaving through the `Gateway Out` blocks are saved too. The simulation inside Vivado allows to compare if the generated netlist behaves like the simulation in Simulink. The simulator uses the saved stimuli and asserts an error if the calculated output values differ from the saved values of Simulink.

Unfortunately, the used license for Vivado does not include the permission for the `AXI BFM`. This license is necessary to simulate the processing system. However, the relevant block `basic_dmt` can be simulated on its own. The `Behavioural Simulation` , `Post-Synthesis Behavioural Simulation` and `Post-Synthesis Timing Simulation` can be skipped as long as the `Post-Implementation Behavioural Simulation` and `Post-Implementation Timing Simulation` are successful. Only if the post-implementation simulations are failing, the error should be searched in the other simulation.

The simulations have no errors and behave like the simulation in Simulink.

6.5. Bit Error Ratio Measurements

Table 6.1 shows the comparison of the BER for double precision and several Q-formats. The measurement duration is $4.2ms$ with 209460 transmitted bits and is repeated for

OFDM and DMT mode using 18 dB and 16 dB SNR in the AWGN channel model.

Measuring with 16 bit in the `s8Q7` format behaves similar to the double precision measurement. This format has enough bits in the integer-part to minimize the clipping of the amplitudes, hence, it can be considered not including errors due to PAPR. In addition, it has enough bits in the fractional part to not induce errors by quantization.

Measuring with the assumed `s5Q3` format shall show that the value is a good trade-off between hardware resource requirements and performance, the value includes errors induces by quantization in the fractional-part and due to PAPR.

The measurement with 16 bit and the `s12Q3` format shall show the influence of the quantization in the fractional-part without errors due to PAPR.

The result of the calculation with double precision is largely the value from Equation 3.8, the difference is around 5% and results from the assumed fixed signal power while it should be measured for each symbol. The measurements with `s8Q7` shows that the BER is similar to double precision.

A comparison of the measurements of `s5Q3`, `s12Q3` with `s8Q7` in OFDM mode shows that the additional BER is induced through the quantization in the fractional part and not by the integer bits.

This measurement in DMT mode shows that 6 integer bits are not enough because the BER is rising by one power of ten. Therefore, the DMT mode should be run with at least the `s6Q3` format.

Precision	**SNR/dB**	**Expected BER**	**OFDM BER**	**DMT BER**
double	18	$1.1 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$
s8Q7	18	$1.1 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$1.07 \cdot 10^{-3}$
s5Q3	18	$1.1 \cdot 10^{-3}$	$1.58 \cdot 10^{-3}$	$1.21 \cdot 10^{-2}$
s12Q3	18	$1.1 \cdot 10^{-3}$	$1.57 \cdot 10^{-3}$	$1.65 \cdot 10^{-3}$
double	16	$6.8 \cdot 10^{-3}$	$6.48 \cdot 10^{-3}$	$6.59 \cdot 10^{-3}$
s8Q7	16	$6.8 \cdot 10^{-3}$	$6.49 \cdot 10^{-3}$	$6.6 \cdot 10^{-3}$
s5Q3	16	$6.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$2.31 \cdot 10^{-2}$
s12Q3	16	$6.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$7.71 \cdot 10^{-3}$

Table 6.1.: BER for the different precisions.

6.6. Verification on the FPGA

Figure 6.4 shows three output signals on an Tektronix TDS 2042C oscilloscope, which is connected to the SMB connectors of the Connection Extension Board, refer to Figure A.5 on page 87.

The blue signal shows the transmitted bits without delay while the purple signal shows the received bits. The yellow signal allows to check for bit errors because it outputs the `XOR` operation of the received- and delayed transmitted bits.

The picture shows that the system is working like the simulation and behaves like expected.

Furthermore, the picture shows that the samples with the period of 20 ns are distorted. On a future analogue front-end, the period is increased to 28 ns for DMT and 56 ns for OFDM. Nonetheless, especially for DMT it should be considered to optimize the transmission lines.

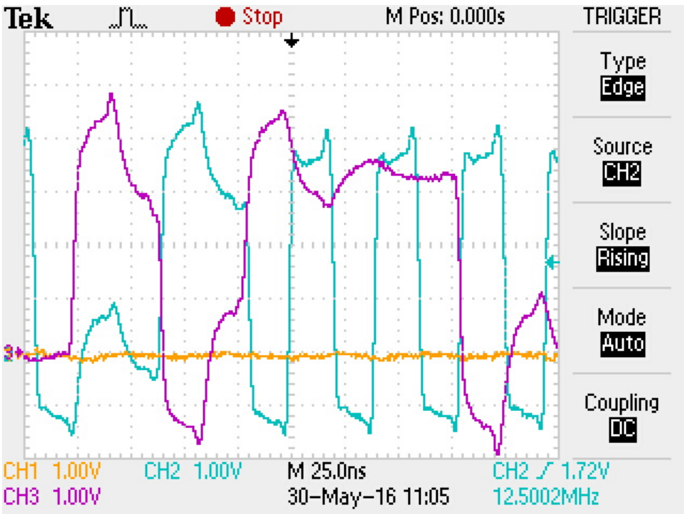


Figure 6.4.: The output signals on the oscilloscope.

7. Conclusion

It has been successfully implemented a system which can perform OFDM and DMT on an Zynq-7020 FPGA. Data rates up to 50 MBit/s are possible while using 16 subcarriers. The System performs 16-QAM modulation and inserts a GI.

To show the performance of the system, the Simulink AWGN channel model is used and the BER is measured. An equalizer using the zero forcing solution is added to improve the performance of the system. A channel model based on an FIR-filter is implemented to test the equalizer.

The design has been successful implemented on the FPGA and the correct results have been measured to verify the operation of the system. The BER measurements show that in OFDM mode at least the `s5Q3` format should be used and in the DMT mode the `s6Q3`.

After describing the fundamentals of the OFDM system for the implemented blocks, some of the advanced blocks are described too, to give a starting point for a future continuation. While describing the design process and setting of all system parameters, they are closely examined and described in detail. Furthermore, the relevant parameters of the system boundaries have been developed to allow an easier creation of the adjacent system parts.

7.1. Future Work

- With the fixed system parameters, an analogue transceiver board for wired communication, replacing the connection extension board, can be developed which implements the DAC/ADC hardware and the line drivers. This allows to create real transmissions outside of the FPGA between multiple nodes using the DMT mode.

- When the design is configured for OFDM transmission, an analogue transceiver board for wireless communication can be created. This has to implement the DAC/ADC hardware and an quadrature modulator to lift the signal in a given frequency range. With the creation of the amplifier, antenna and recovery hardware, the transmission can be done wireless.

- The design can be extended with several improvements. An channel estimation algorithm can be applied to allow the equalizer to adapt automatically to a time dependent channel. With the dynamic measurement of the channel, an dynamic modulation control can be implemented which varies the use of different modulation alphabets. This can be used to utilize the water filling theorem and make the system less prone to bad channel conditions.

- To make the system even less vulnerable to bit errors, a channel coding algorithm should implemented. Another benefit of the channel coding is the achievement of a diversity gain, because the bits are spread over multiple subcarriers[Kam11, S.600]. It has to be considered that this would reduce the usable bit rate. Using a convolutional coding, the hard-decision of the bit demodulation process can be replaced by a soft-decision algorithm to reduce the bit error probability even further.

- It can be implemented an algorithm for PAPR reduction, this would reduce the necessary bits in the non-fractional part of the data words.

- The use of multiple nodes makes it necessary to implement a sampling frequency synchronisation. Creating wireless transmission, a synchronisation circuit for the carrier frequency has to be implemented on the hardware.

- Another interesting possibility is the creation of a custom Simulink and System Generator FFT block. This should allow to prevent the differences while rounding.

A. Appendix

A.1. Figures

This section shows additional figures which are referenced in this thesis.

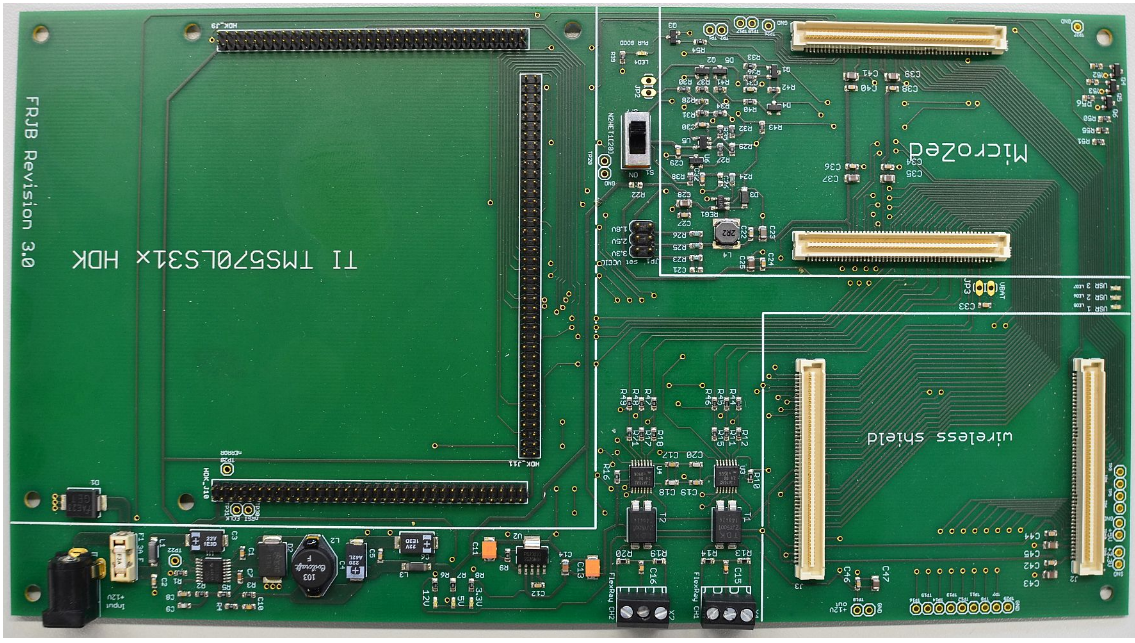


Figure A.1.: The carrier board for the MicroZed and the connection extension.

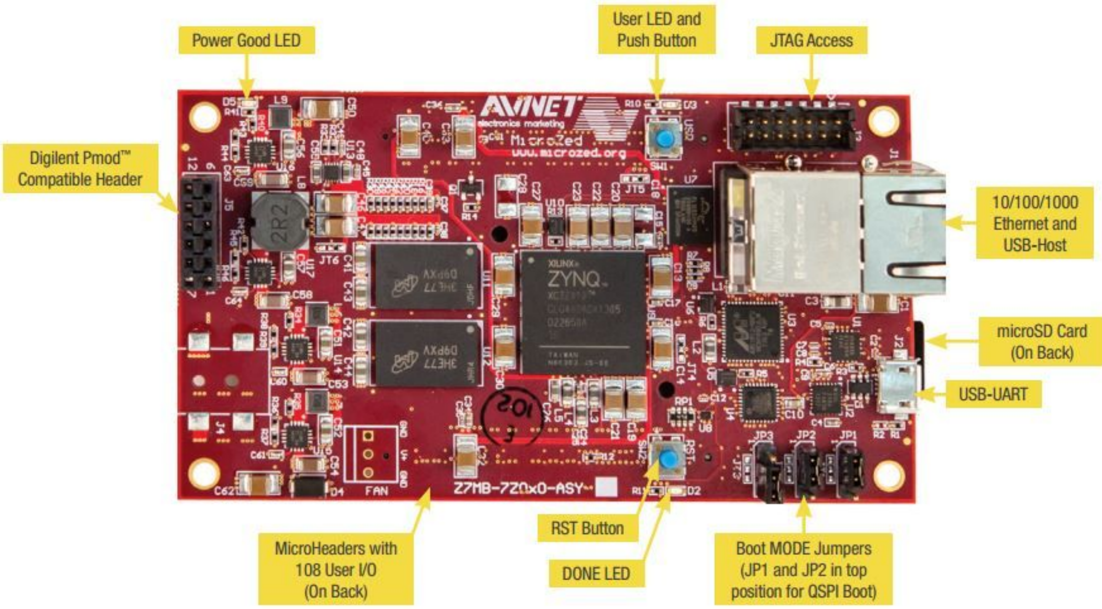


Figure A.2.: The MicroZed board [AVNb].

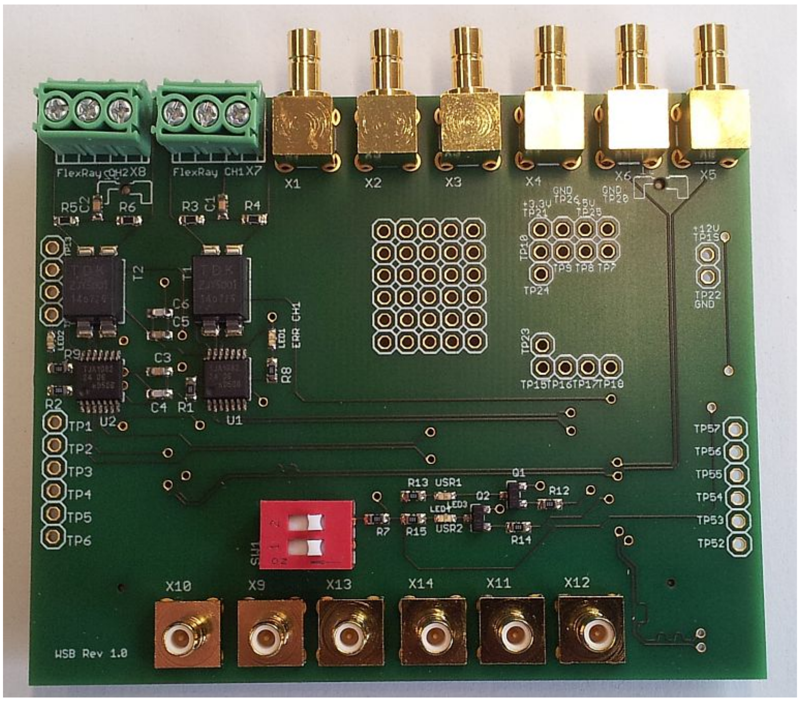


Figure A.3.: The Connection Extension Board which allows to connect measurement devices.

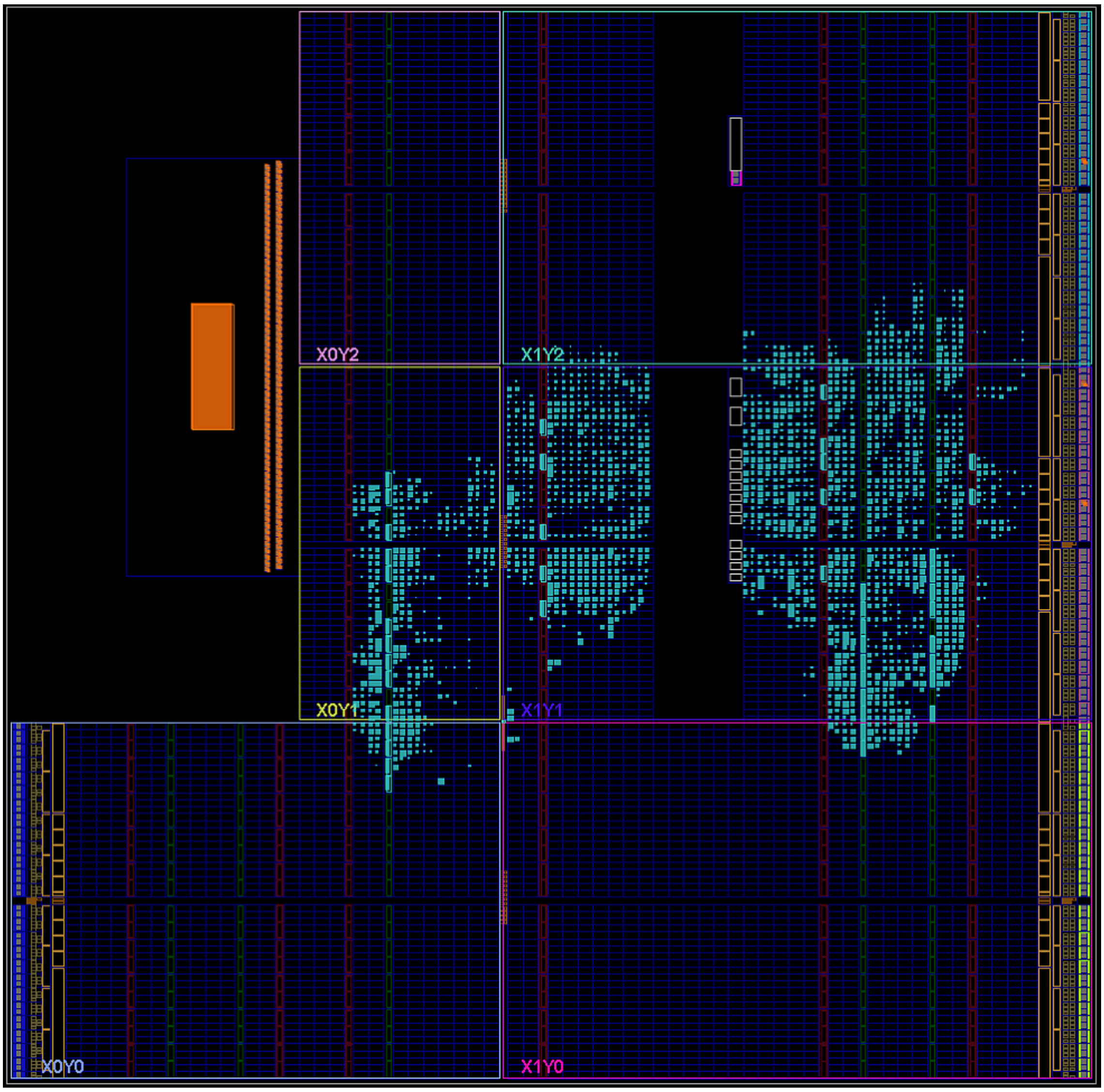


Figure A.4.: Graphical view of the resource utilization.

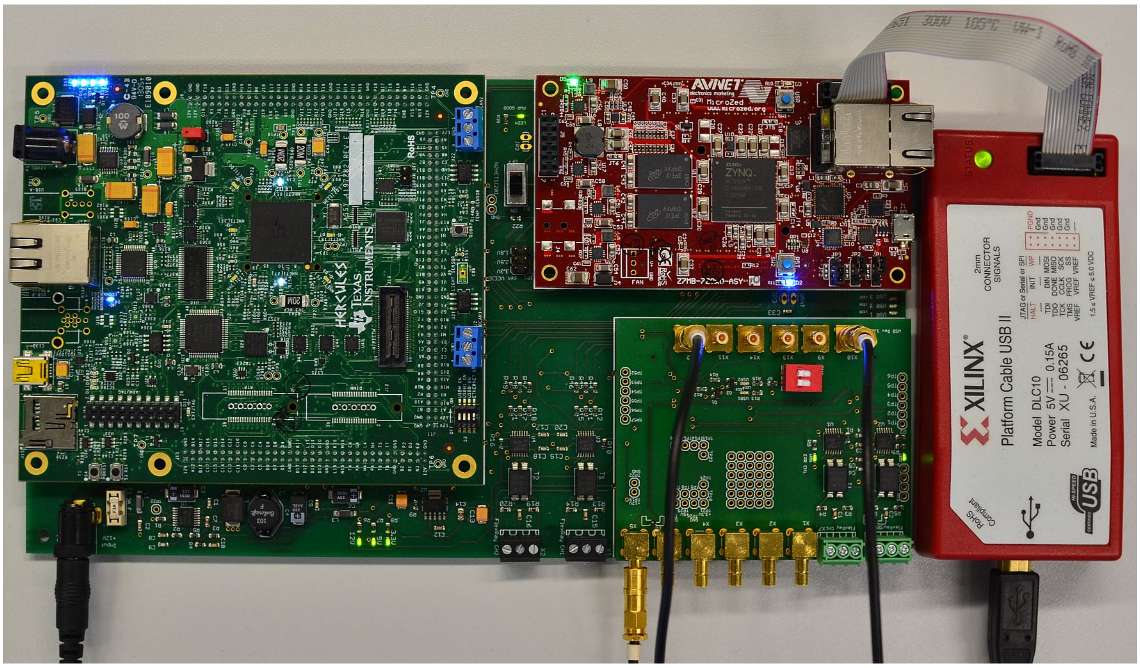


Figure A.5.: The assembled measurement hardware.

A.2. Files on the Data Medium

This chapter gives an overview of the files on the data medium. The data medium can be reviewed at the supervising examiner and the second examiner. The following folders can be found:

- `1_PDF` - Containing this document.

- `2_Baseband_Controller` - Containing `Basic_DMT.slx` the main model which includes the described Simulink and System Generator blocks. After opening, `initScript.m` is run, which is in this folder too. It also includes all source codes for the `MCode` blocks.

- `3_Random_Bit_Generator` - Containing `RNDBitGen.slx` which implements the random bit generator using the LFSR, refer to Section 4.5. This folder contains an `initScript.m` too. This is a different one then for `Basic_DMT.slx`.

- `4_Vivado_Project` - This folder contains an example project with a finished synthesis and implementation of the system in DMT mode with deactivated equalizer. Section A.3 explains how it can be run on the hardware.

A.3. Example Vivado Project

This section explains how to run the example from the data medium on the hardware platform:

1. After opening the project, the successful implemented design has to be opened within the `Implementation->Implemented Design->Design Runs` section.

2. Clicking `Program and Debug->Generate Bitstream` prepares the bit stream for the FPGA.

3. Clicking `File->Export->Export Hardware` and selecting `Include bitstream` has to be done next.

4. The using of the PS as clock source makes it necessary to initialize the PS in addition to the PL. Launching the design within the Software Development Kit (SDK) is the easiest way to do it because the initialization is automated while doing this. Therefore, clicking `File->Launch SDK` is performed next.

5. Inside the SDK a C program has to be selected to run on the processing system. A simple *Hello World* project template is used, because the project is only used for initialization. Right clicking on `hel_wld` and selecting `Run as->Run configurations..` opens a dialogue.

6. The slide `Target Setup` includes a drop down menu where `Reset Entire System` has to be selected. The check boxes `Program FPGA`, `Run ps7_init` and `Run ps7_post_config` have to be checked.

7. Clicking the button `Run` starts the programming and the signals can be measured on an oscilloscope.

Bibliography

[Adl15] Frederic Alexander Hermann Adler. Entwicklung einer Evaluierungsplattform für drahtlose und drahtgebundene FlexRay-Datenkommunikation im Automobil, 2015.

[AVNa] AVNET. MicroZed Evaluation Kit. `http://zedboard.org/sites/default/files/product_briefs/pb-microzed-eval-v2a.pdf` [Online; accessed 12-05-2016].

[AVNb] AVNET. Quick Start Instructions MicroZed Evaluation Kit. `http://zedboard.org/sites/default/files/documentations/QSC-Z7MB-7Z010-G-v1d_0.pdf` [Online; accessed 12-05-2016].

[Cha08] Chassaing, Rulph/ Reay, Donald. *Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK (Topics in Digital Signal Processing, Band 1).* John Wiley & Sons, 2. auflage edition, 5 2008.

[COS89] COST. *COST 207 Digital Land Mobile Radio Communications.* European Commission, 1989.

[Far98] D. Farnese. Techniques for Peak Power Reduction in OFDM Systems, 1998.

[FLPS02] D. Farnese, A. Leva, G. Paltenghi, and A. Spalvieri. Pulse superposition: a technique for peak-to-average power ratio reduction in OFDM modulation. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 3, pages 1682–1685 vol.3, 2002.

[HAW] HAW Hamburg. Zukunftsprogramm der Fakultät TI - Neue Lehrangebote. `https://www.haw-hamburg.de/fakultaeten-und-departments/ti/unsere-fakultaet/zukunftsprogramm-der-fakultaet-ti/gefoerderte-projekte.html` [Online; accessed 14-04-2016].

[Her14] Hering, Ekbert/ Bressler, Klaus/ Gutekunst, Jürgen. *Elektronik für Ingenieure und Naturwissenschaftler (Springer-Lehrbuch)*. Springer Vieweg, 6., vollst. aktualisierte u. erw. aufl. 2014 edition, 5 2014.

[Höh13] P.A. Höher. *Grundlagen der digitalen Informationsübertragung: Von der Theorie zu Mobilfunkanwendungen*. Springer Fachmedien Wiesbaden, 2013.

[IEE] IEEE Standards Association. IEEE 802®: Local And Metropolitan Area Network Standards. `http://standards.ieee.org/getieee802/download/802.11-2012.pdf` [Online; accessed 28-04-2016].

[JED] JEDEC SOLID STATE TECHNOLOGY ASSOCIATION. Interface Standard for Nominal 3 V/3.3 V Supply Digital Integrated Circuits . `http://www.jedec.org/sites/default/files/docs/jesd8c-01.pdf` [Online; accessed 24-05-2016].

[Jon03] Jonathan Kemp. Generating an MLS sequence, 2003. `http://www.kempacoustics.com/thesis/node83.html` [Online; accessed 12-05-2016].

[JWC65] John W. Tukey James W. Cooley. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[Kam11] K.-D. Kammeyer. *Nachrichtenübertragung*. Vieweg+Teubner Verlag, 8 2011.

[Mata] MathWorks. Mathworks documentation: Awgn channel. `http://de.mathworks.com/help/comm/ref/awgnchannel.html` [Online; accessed 28-05-2016].

[Matb] MathWorks. Mathworks documentation: Rectangular qam modulator baseband. `http://de.mathworks.com/help/comm/ref/rectangularqammodulatorbaseband.html` [Online; accessed 28-04-2016].

[Matc] MathWorks. Precision and range. `http://de.mathworks.com/help/fixedpoint/ug/precision-and-range.html#f6481` [Online; accessed 15-06-2016].

[MC58] R. R. Mosier and R. G. Clabaugh. Kineplex, a bandwidth-efficient binary transmission system. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 76(6):723–728, Jan 1958.

[Pap11] Lothar Papula. *Mathematische Formelsammlung: für Ingenieure und Naturwissenschaftler (German Edition)*. Vieweg+Teubner Verlag, 10. aufl. 2009 edition, 11 2011.

[Pon06] Pong P. Chu. *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*. Wiley-IEEE Press, 1 edition, 4 2006.

[Pro03] John G. Proakis, editor. *Wiley encyclopedia of telecommunications*, volume 2. Wiley, Hoboken, NJ, 2003.

[Pro08] John G. Proakis. *Digital Communications*. McGraw-Hill, 2008.

[Ras16] Tobias Frank Rastetter. Modellierung von passiven Bustopologien im Automobil für hohe Datenraten, 2016.

[Rei10] Konrad Reif, editor. *Batterien, Bordnetze und Vernetzung (Bosch Fachinformation Automobil) (German Edition)*. Vieweg+Teubner Verlag, 2010 edition, 6 2010.

[Rei11] Jürgen Reichardt. *Lehrbuch Digitaltechnik: Eine Einführung mit VHDL*. Oldenbourg Wissenschaftsverlag, 2 edition, 5 2011.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423, 623–656, July, Oct. 1948.

[Shi10] William Shieh. *OFDM for optical communications*. Academic, London, 2010.

[SK98] H. Schmidt and K. D. Kammeyer. Reducing the peak to average power ratio of multicarrier signals by adaptive subcarrier selection. In *Universal Personal Communications, 1998. ICUPC '98. IEEE 1998 International Conference on*, volume 2, pages 933–937 vol.2, Oct 1998.

[Vec] Vector Informatik GmbH. Vector E-Learning - FlexRay - Brake-by-Wire. `https://elearning.vector.com/index.php?seite=vl_flexray_introduction_de&root=376493&wbt_ls_kapitel_id=454126&wbt_ls_seite_id=454133&d=yes` [Online; accessed 14-04-2016].

[VLS] VLSI Concepts. "Examples Of Setup and Hold time" : Static Timing Analysis (STA) basic (Part 3c). `http://www.vlsi-expert.com/2011/05/`

example-of-setup-and-hold-time-static.html [Online; accessed 25-05-2016].

[Vol] Jürgen Vollmer. Script - Digitale Modulationsverfahren - Lineare Modulation.

[WE71] S. Weinstein and P. Ebert. Data transmission by frequency-division multiplexing using the discrete fourier transform. *Communication Technology, IEEE Transactions on*, 19(5):628–634, October 1971.

[Wik] Wikipedia. Parseval's theorem. https://en.wikipedia.org/wiki/Parseval%27s_theorem [Online; accessed 05-06-2016].

[Wol] Wolfram MathWorld. Wolfram MathWorld - Erfc. http://mathworld.wolfram.com/Erfc.html [Online; accessed 05-06-2016].

[Xil] Xilinx. Zynq-7000 All Programmable SoCs Product Tables and Product Selection Guide. http://www.xilinx.com/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf [Online; accessed 12-05-2016].

[Xil14a] Xilinx. UG897 - Vivado Design Suite User Guide - Model-Based DSP Design using System Generator, 2014.

[Xil14b] Xilinx. UG904 - Vivado Design Suite User Guide - Implementation, 2014.

[Xil14c] Xilinx. UG906 - Vivado Design Suite User Guide - Design Analysis and Closure Techniques, 2014.

[Xil14d] Xilinx. UG958 - Vivado Design Suite Reference Guide - Model-Based DSP Design using System Generator, 2014.

[Xil15] Xilinx. Fast Fourier Transform v9.0 - LogiCORE IP Product Guide - PG109, 2015.

[Zed] ZedBoard.org. MicroZed. http://zedboard.org/product/microzed [Online; accessed 12-05-2016].

Symbols

Notation	Description
a	Symbols in the complex plane for the calculation of the average symbol power.
B	Overall bandwidth.
b_c	Coherence bandwidth.
B_{DMT}	DMT bandwidth.
β	Bandwidth efficiency.
B_{OFDM}	OFDM bandwidth.
B_{sc}	Subcarrier bandwidth.
d	Normalizes the distance for calculation of the average symbol power.
δ	Factor which takes account for the unused subcarriers in reduction of SNR.
$d_n(i)$	The i-th symbol of the n-th subcarrier.
$\hat{d}_n(i)$	Decided signal before the demodulator of the n-th subcarrier for the i-th OFDM symbol.
E_b	Average energy per bit.
E_s	Average energy per symbol.
f_{clk}	Fundamental sample rate.
f_n	Frequency of the n-th subcarrier.

Notation	Description
f_s	Sampling frequency.
$f_{s,DMT}$	Sampling frequency ADC/DAC using DMT.
$f_{s,OFDM}$	Sampling frequency ADC/DAC using OFDM.
γ	Summarizes the factors which reduce the SNR.
$g_s(t)$	Transmit filter impulse response.
$h(k)$	Discrete FIR-filter channel model impulse response.
$H(n)$	Channel model transfer function.
$h(t)$	FIR-filter channel model impulse response.
K	Number of symbols in the modulation aplhabet.
k	Normalizes index k'.
k'	Index for the samples within one OFDM symbol.
L	Number of OFDM symbols in a block.
M	Number of bits for K-QAM.
n	Index for the subcarriers.
N_0	PSD of the white Gaussian noise.
$n_a(t)$	AWGN noise of the AWGN channel model.
N_{chn}	Number of samples on the channel containing the OFDM symbol and the GI
N_{conv}	Total number of converter bits.
N_{DMT}	FFT length for DMT.
N_{frac}	Number of fractional bits.
N_{GI}	Number of samples for the GI.
$N_{GI,DMT}$	Number of samples for the GI for DMT.

Notation	**Description**
$N_{GI,OFDM}$	Number of samples for the GI for OFDM.
N_{input}	Number of input bits for the IFFT.
N_{int}	Number of integer bits.
N_{OFDM}	FFT length for OFDM.
N_{sc}	Number of subcarriers.
N_{unused}	Number of unused subcarriers.
N_{used}	Number of used subcarriers.
P	Number of pilot symbols.
$P_{b,16-QAM}$	Approximation of the BER for 16-QAM.
$P_{b,M-QAM}$	Approximation of the BER for M-QAM.
P_s	Input signal power of the AWGN channel model.
$\overline{P}_{sym}$	Average symbol power for 16 QAM.
R_{bit}	Payload bit rate.
R_{chn}	Bit rate on the channel.
$r_{chn}(i,k)$	Signal before the GI removal at the receiver.
$r(i,k)$	Signal after the GI removal at the receiver.
s	Slack between source and destination register.
$s_{chn}(i,k)$	Discrete multicarrier signal including GI.
$s(i,k)$	Discrete multicarrier signal before GI.
s_{MC}	General multicarrier signal.
s_{MC_disc}	Discrete multicarrier signal.
s_{MC_simp}	Simplified multicarrier signal.
SNR_{comp}	SNR for a complex baseband channel.
SNR_{real}	SNR for a real-valued channel.
T_{sym}	OFDM symbol duration.

Notation	**Description**
T_a	Specific sample period to show the necessity of FIFOs.
τ_{max}	Duration of the significant parts of the channel impulse response.
T_{bit}	Payload bit period.
T_{calc}	Calculation period corresponding to 62.5 MBit/s.
T_{chn}	Period on the channel including GI- and symbol duration.
T_{clk}	Fundamental sample period.
T_d	Sum of T_{dp} and T_{scp}. Duration until the data reaches the destination register
T_{dcp}	Duration until the clock reaches the destination register.
T_{dp}	Duration of the data path.
T_{GI}	Guard interval duration.
$T_{IFFT,DMT}$	Word period at the IFFT for DMT.
$T_{IFFT,OFDM}$	Word period at the IFFT for OFDM.
T_p	Word period after parallelization.
T_s	Sampling period.
T_{scp}	Duration of the source clock path.
$X(f)$	Used to show the condition for a real-valued signal.
$x_n(i)$	Signal at the receiver after the FFT of the n-th subcarrier for the i-th OFDM symbol
$x(t)$	Used to show the condition for a real-valued signal.
$y_n(i)$	Signal at the receiver after the equalizer of the n-th subcarrier for the i-th OFDM symbol

Acronyms

Notation	Description
ABS	Anti-lock Breaking System.
ADC	Analog Digital Converter.
ASM	Algorithmic State Machine.
AWGN	Additive White Gaussian Noise.
BER	Bit Error Ratio.
BGA	Ball Grid Array.
BPSK	Binary Phase Shift Keying.
CAN	Controller Area Network.
DAC	Digital Analog Converter.
DC	Direct Current.
DFT	Discrete Fourier Transformation.
DMT	Discrete Multi Tone.
DSP	Digital Signal Processing.
ESP	Electronic Stability Program.
FFT	Fast Fourier Transformation.
FIFO	First In First Out.
FIR	Finite Impulse Response.
FPGA	Field Programmable Gate Array.

Notation	**Description**
FSM	Finite State Machine.
GI	Guard Interval.
GSM	Global System for Mobile Communications.
GUI	Graphical User Interface.
HAW	Hamburg University of Applied Sciences.
ICI	Inter Carrier Interference.
IDFT	Inverse Discrete Fourier Transformation.
IFFT	Inverse Fast Fourier Transformation.
ISI	Inter Symbol Interference.
LFSR	Linear Feedback Shift Register.
LIN	Local Interconnect Network.
LSB	Least Significant Bit.
LUT	Look Up Table.
MLS	Maximum Length Sequence.
MMSE	Minimum Mean Square Error.
OFDM	Orthogonal Frequency Division Multiplexing.
PAPR	Peak to Average Power Ratio.
PL	Programmable Logic.
PS	Processing System.
PSD	Power Spectral Density.
QAM	Quadrature Amplitude Modulation.

Notation	Description
RF	Radio Frequency.
SDK	Software Development Kit.
SNR	Signal to Noise Ratio.
SoC	System on Chip.
VHDL	Very High Speed Integrated Circuit Hardware Description Language.
WLAN	Wireless Local Area Network.

Declaration

I declare within the meaning of §16(5) APSO-TI-BM of the Examination and Study Regulations of the Master of Science degree programme Information and Communication Engineering that: this Master thesis has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Neu Wulmstorf, June 13th 2016	
City, Date	Signature