Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2021

All-digital FPGA receiver

on Intel Stratix 10 TX

Kim Persson

Master of Science Thesis in Electrical Engineering

Master thesis report

Kim Persson

LiTH-ISY-EX--21/5430--SE

Supervisor:

Ted Johansson

ISY, Linköping University

Examiner

Mark Vesterbacka

ISY, Linköping University

Abstract

Ordinary digital receivers use analog mixers and conventional analog-to-digital converters. This thesis explores the possibility to build a receiver using an Intel Stratix 10 TX FPGA to sample a signal directly at its RF frequency. By band-limiting the RF signal, the transceivers can be used as PWM based analog-to-digital converters and sample the RF signals as a bit-stream. Filters and mixers could then be implemented in the FPGA to downconvert the RF signal to a complex digital baseband signal. This method was proven to work in theory but had drawbacks. The sampling method added a lot of distortion and the bandwidth was limited. The receiver could not be implemented on the FPGA since the only connected transceivers could not be configured to receive analog signals. The other type of transceiver could probably be used but would have required another model of the board where these were connected.

Sammanfattning

Vanligtvis använder digitala mottagare analoga mixrar och konventionella analog-till-digital-omvandlare. Det här examensarbetet undersöker om det är möjligt att bygga en mottagare av en Intel Stratix 10 TX FPGA genom att använda dess sändtagare till att sampla en signal direkt vid dess radiofrekvens. Genom att begränsa bandbredden kan sändtagarna användas som analog-till-digital-omvandlare baserade på PWM och sampla radiofrekvenssignalen som en bitström. Filter och mixrar kan då implementeras i FPGAn och radiofrekvenssignalen kan där konverteras ned till en komplex digital basbandssignal. Den här metoden kan fungera teoretiskt men den hade nackdelar som att samplingsmetoden förvrängde signalen och bandbredden var begränsad. Mottagaren gick inte att implementera på FPGAn då den enda typen av sändtagare som var kopplade på kortet inte kunde konfigureras till att ta emot analoga signaler. Den andra typen av sändtagare hade troligen gått att använda men hade krävt en annan modell av kortet där dessa var inkopplade.

Table of contents

List of figures

List of Tables

Notation

Abbreviation/ Acronym	Meaning	Explanation	Context
ADC	Analog to Digital Converter	Device used to convert a time-continuous analog signal to a time-discrete digital signal.	Used to sample the RF signal.
DAC	Digital to Analog Converter	Device used to convert a time-discrete digital signal to a time-continuous analog signal.	See chapter 2.1.
DDC	Digital Down Converter	Moves an RF signal to a lower frequency at a lower sampling rate.	Used in the receiver.
DSM	Delta-Sigma Modulation	Method for encoding an analog signal using only two levels (1 and 0).	See chapter 2.1.
EVM	Error Vector Magnitude	A measure used for the performance of digital radio receivers and transmitters.	Used in simulations to calculate expected receiver performance.
FIR	Finite Impulse Response	A type of digital filter.	Used as lowpass filters for test signals and in the receiver.
FPGA	Field Programmable Gate Array	Integrated circuit that can be configured after manufacturing.	Used to implement the receiver.
I	In-phase	The “real” part of a complex signal used in QAM.	See chapter 2.3.
ISY	Institutionen för SYstemteknik	One of the departments of the Linköping institute of technology. Department of electrical engineering in English.	This thesis was made at ISY.
LNA	Low-Noise Amplifier	Amplifier used to amplify weak signals with a minimum of added noise.	Used to amplify a signal before the ADC.

NRZ	Non-Return-to-Zero	A modulation method where the information is encoded into pulses with either a positive or negative voltage.	Used by digital transceivers.
PAM4	Four-level Pulse Amplitude Modulation	A modulation method where the information is encoded into pulses with four voltage levels.	Used by digital transceivers.
PCM	Pulse Code Modulation	Method to represent a sampled analog signal.	The output from the receiver is in this format.
PFM	Pulse-Frequency Modulation	Method for encoding an analog signal using only two levels (1 and 0).	See chapter 2.1.
PLL	Phase Locked Loop	A system where the phases of the output signal is locked to the phase of the input signal.	Used to create clock signals.
PWM	Pulse Width Modulation	A method to control power by using a switch that is either fully on or fully off.	See chapter 2.1 and 2.5.
RF	Radio Frequency	An oscillation in the frequency range from around 20 kHz to around 300 GHz.	The receiver will receive a RF signal at 2.4 GHz.
SDR	Software Defined Radio	A radio communication system implemented in software.	The receiver is based on the same principle.
SNR	Signal to Noise Ratio	The ratio between the signal power and the noise power.	Used as a measurement.
VGA	Variable Gain Amplifier	Amplifiers where the gain depends on a control signal.	Used to change the gain before the ADC.
VHDL (VHSIC-HDL)	Very High Speed Integrated Circuit Hardware Description Language	Hardware language to describe digital circuits.	The receiver is implemented in VHDL.

Q	Quadrature	The “imaginary” part of a complex signal used in QAM.	See chapter 2.3.
QAM	Quadrature Amplitude Modulation	A method to modulate two bit streams with two out-of-phase carriers.	Common modulation method used in digital communication.

1 Introduction

1.1 Motivation

The use of wireless communications has increased greatly over the last decades. Several standards of cellular networks and wireless LAN are now used with more and more connected devices. Traditionally, components like mixers, filters, modulators/demodulators have been implemented in hardware. With Software Defined Radio, SDR, see Figure 1.1, which is a communication system where hardware components have been replaced by software, the whole spectrum can be sampled and processed in the digital domain [1]. There are analog-to-digital converters, ADCs, capable of sampling frequencies up to a few gigahertz with high precision, typical 14 bits, but they are expensive and have high energy consumption. Another way is to band-limit the signal with an analog bandpass filter and then sample with a 1-bit ADC. FPGAs with high speed transceivers can be used to process the sampled signal since they easily can be reconfigured to handle different frequencies and modulation methods. By using 1-bit ADCs the cost and the power consumption would be much less but at the price of limited bandwidth and more noise. Intel Stratix 10 is a family of FPGAs with such high-speed transceivers that could be possible to use.

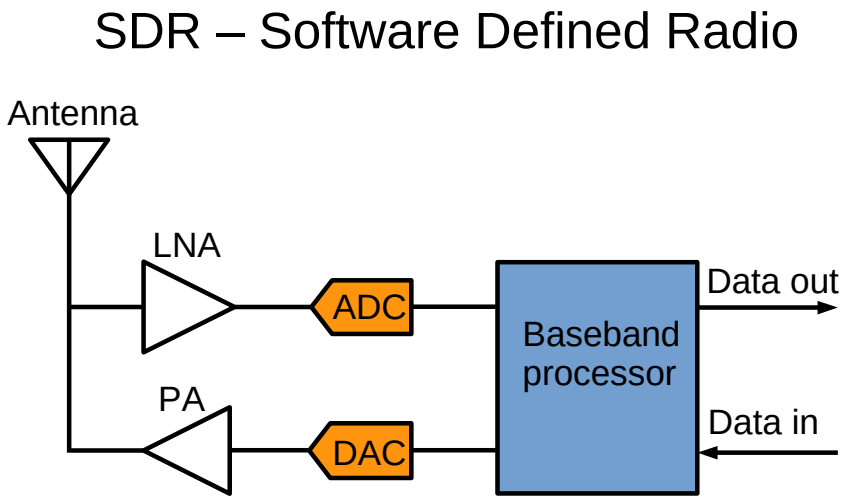


Figure 1.1: *Ideal Software Defined Radio where the antenna is directly connected to the ADC and DAC.*

1.2 Purpose

The main purpose of this thesis is to design and implement a digital receiver on an FPGA and replace traditional analog components with digital counterparts implemented in VHDL. With digital circuits some of the disadvantages of analog circuits can be avoided and by using an FPGA,

changes can easily be done without hardware modifications. This receiver may, for example, receive a 64-QAM signal by sampling it directly at its RF frequency and then demodulate it into a digital complex baseband signal with much lower sampling frequency.

The hardware used is an Intel Stratix 10 TX development kit. It was chosen because of its fast transceivers which are necessary for carrier frequencies above 1 GHz. In addition to implementing the receiver on an FPGA, the receiver will be simulated in MATLAB with different parameters. By testing different sampling frequencies, carrier frequencies, bandwidths and the presence of other signals, the limit of the receiver will be examined.

1.3 Problem statements

While designing the receiver, there are problems that need to be solved to progress. The following questions are to most interesting ones to find answers to during this thesis.

- Which method for the 1-bit ADC would be best to implement on the FPGA?
- How well would a 64-QAM signal with 20 MHz bandwidth be received?
- How can a simplified version of the receiver be implemented in VHDL?
- What problems/disadvantages can be found with this type of receiver?
- Can the available Intel Stratix 10 TX be used?

1.4 Limitations

The Intel Stratix 10 TX development kit has several types of connections to the outside world, like USB, Ethernet, MXP, QSFPDD, FMC+ and 2.4 mm RF connectors. Because of limited time and available hardware in the lab, only the RF connectors will be used.

A receiver is a complex system and things like phase and frequency offsets will affect the performance if they are not compensated. In this thesis it is assumed that these errors do not exist and no compensation is needed.

The system implemented in VHDL will only be able to receive a signal using a carrier frequency of 2.40 GHz to simplify the design. Other carrier frequencies will only be simulated.

No other RF signals than the test signal will be present when the tests on the FPGA are performed. An analog bandpass filter is therefore not necessary.

2 Theory

2.1 RF 1-bit ADC

In this thesis the RF signal will be sampled by a 1-bit ADC. With conventional ADCs, where the samples consist of several bits, all frequencies up to half the sampling frequency can be included. With just one bit the whole spectrum would be drowned in quantization noise. If instead just the frequencies of interest were filtered out by a bandpass filter the signal could be sampled using only one bit if the sampling frequency is much higher than the bandwidth of the signal. Since the goal with this thesis is to do as much as possible in the digital domain the signal has to be sampled at its RF frequency to avoid the use of analog mixers. Here three possible methods are discussed.

The first possible method for doing 1-bit ADC is Delta-Sigma modulation (DSM) [2]. Figure 2.1 shows how a delta-sigma modulator can be built. The modulator consists of a difference amplifier, an integrator, a comparator that acts as ADC and a DAC. Delta-sigma modulators work by letting a difference amplifier measure the difference between the analog input signal and the digital output signal passing through a 1-bit DAC. The output from the difference amplifier is integrated and the integral can go in both positive and negative direction. The comparator compares the integrator output voltage with a reference voltage and whenever the voltage passes the reference voltage, the output of the comparator will switch. The output from the comparator goes to a flip-flop where a sample clock controls when it updates its output. This method would require a whole delta-sigma modulator connected to the transceiver input of the FPGA. For the speed required in this thesis the modulator would have to be built as an integrated circuit.

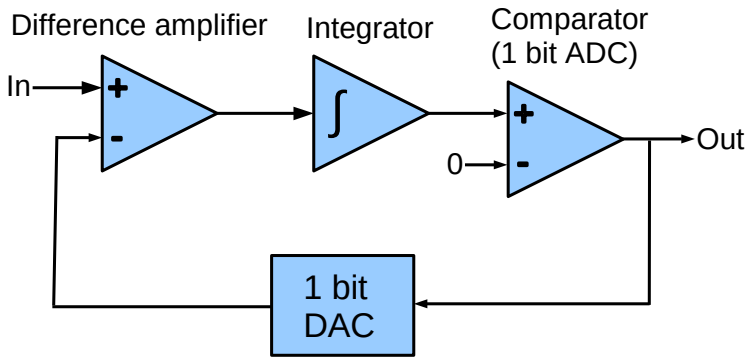


Figure 2.1: *Delta-sigma modulator.*

Another type of 1-bit ADC is based on pulse-frequency modulation (PFM) [3] [4]. PFM uses pulses with fixed width and modulate the signal by varying the frequency of these pulses. These types of ADCs are based on voltage-controlled oscillators (VCO) that is controlled by the input signal. Figure 2.2 shows how the ADC can be built. The VCO is controlled by x(t), with a DC offset, and outputs a square wave, w(t). This square wave is sampled every T_s second in $w_s[n]$. To get T_s seconds wide pulses, an XOR gate has to be added where one input is $w_s[n]$ and the other $w_s[n]$ delayed one clock cycle. The resulting $y_s[n]$ will then contain pulses for each rising and falling edges of w(t). This method would require a VCO but the sampling could be done with the with FPGA directly for low frequencies and with help of the transceivers for high frequencies. The disadvantage with this method, in addition to have to use a VCO, is that VCOs have a nonlinear behavior. This can be improved but would add complexity.

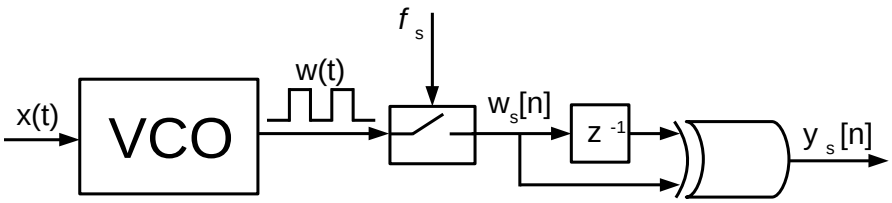


Figure 2.2: *1-bit VCO-ADC.*

A third type of 1-bit ADC is based on pulse-width modulation (PWM) [5]. This ADC consists of a high-speed comparator that compares the RF signal with a reference signal, see Figure 2.3. If the amplitude of the RF signal is higher than the amplitude of the reference signal, the output will be high and vice versa. This output is then sampled. This type of ADC is possible to implement by using almost only the digital transceivers on an FPGA [6]. The transceivers consist of comparators with differential inputs followed by registers that sample the signal and by configuring them in the right way they could work as 1-bit ADCs. A transceiver can also be used to create the reference signal by transmitting a PWM representation of the reference signal filtered by a lowpass filter.

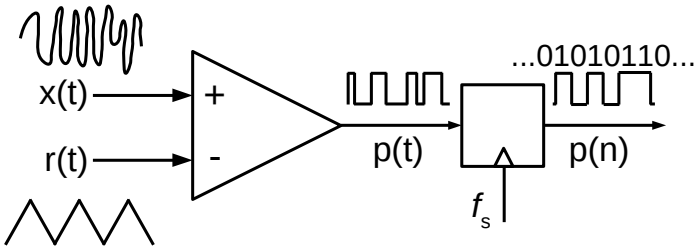


Figure 2.3: *PWM based ADC.*

Using PWM is the simplest way to implement an ADC that could sample the RF signal since only one lowpass filter is needed. Therefore, this method is the one that will be used in this thesis.

2.2 All-digital receiver architecture

Figure 2.4 shows the proposed architecture from [6]. The signal x(t) is the RF signal received by the antenna after being filtered by a bandpass filter and amplified by a low noise amplifier (LNA) with variable gain amplification (VGA). This signal is one of the inputs to a comparator. The other one is r(t), which is a reference signal. This reference signal, which is usually a triangle wave, is created by letting a transceiver transmitting a 1-bit representation of a triangle wave. A lowpass filter then removes the high frequency components. The comparator, which is a part of the transceiver, compares the two signals and if the amplitude of x(t) is higher than the amplitude of r(t) the output will be high, otherwise low. This output is sampled at a rate controlled by the transceiver clock. The transceiver clock is N times higher than the FPGA clock so the serial bit stream has to be deserialized and be transferred from the transceiver N bits at a time.

The polyphase DDC, Digital Down Converter, downconverts, lowpass filters and downsamples the sampled RF signal in N parallel branches. The RF signal is downconverted by using N local oscillators (LO) for the I part and N for the Q part. If the incoming bit is one, then the output from the mixers is equal to the value from the local oscillator otherwise it is zero. The signal is in this step also converted from a 1-bit PWM signal to a pulse code modulated (PCM) signal. The downconverted signal is then filtered through a polyphase lowpass filter and downsampled by a factor N. The resulting signal is the complex baseband signal with a sample rate equal to the FPGA clock.

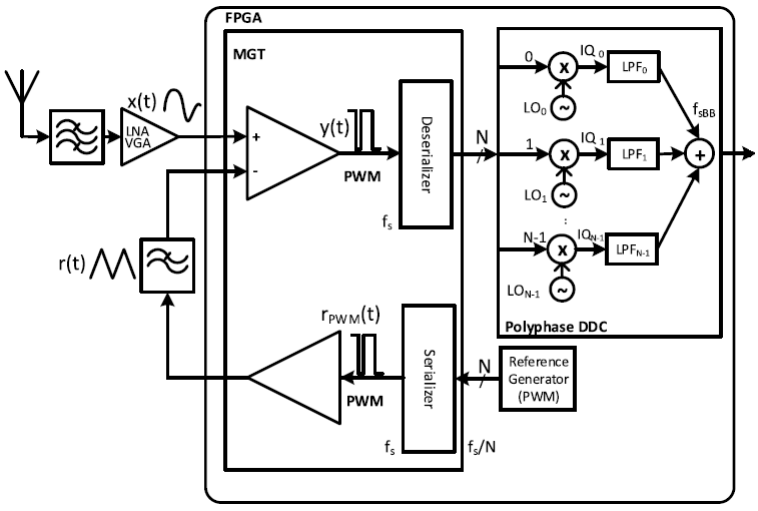


Figure 2.4: *Proposed receiver architecture [6].*

2.3 Quadrature Amplitude Modulation, QAM

QAM is modulation method used for both analog and digital signals that modulates the amplitude of two carrier waves [7]. These carrier waves have the same frequency but are out of phase by 90 degrees and therefore orthogonal. They can therefore be added together without interfering with each other. By using this method, twice the information can be sent by using the same bandwidth compared to if only one carrier was used.

To create a digital QAM signal we have to start with a modulation scheme, which maps a bit stream to symbols. These symbols are complex values that are placed on a square lattice where the real will modulate one of the two carriers and imaginary part the other carrier. Usually the number of symbols is a power of 4. 16-QAM with 16 points, is a scheme where each symbol can represent 4 bits, see Figure 2.5. The first 2 bits can be represented by the real part of the symbol and the last 2 bits by the imaginary part. By using Gray coding, only one bit will differ between neighbors so if noise will cause a symbol error at the receiver side, most likely just one bit will be wrong. If we want to send the text “Hello World!” using 16-QAM, we can start by using ASCII to encode the characters into bits and then use the scheme to map the 96 bits to 24 symbols, see Table 2.1.

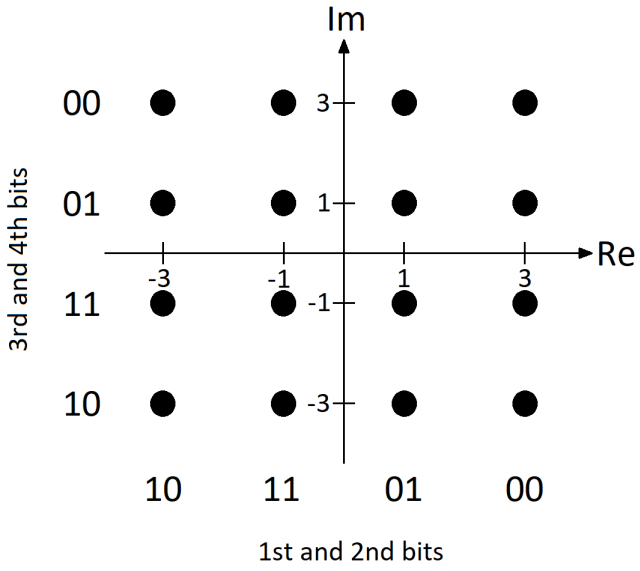


Figure 2.5: *16-QAM signal space diagram.*

Character	Hexadecimal	Binary	Symbols
H	48	0100 1000	1+3i, -3+3i
e	65	0110 0101	1-3i, 1+1i
l	6C	0110 1100	1-3i, -1+3i
l	6C	0110 1100	1-3i, -1+3i
o	6F	0110 1111	1-3i, -1-1i
space	20	0010 0000	3-3i, 3+3i
W	57	0101 0111	1+1i, 1-1i
o	6F	0110 1111	-1-3i, -1-1i
r	72	0111 0010	1-1i, 3-3i
l	6C	0110 1100	1-3i, -1+3i
d	64	0110 0100	1-3i, 1+3i
!	21	0010 0001	3-3i, 3+1i

Table 2.1: *Text to binary.*

This series of symbols is the digital complex baseband signal. Before being converted into two analog signals, the baseband signal has to be upsampled and filtered with a lowpass filter called PAM filter, see section 2.4. The digital and analog versions of the baseband signal are shown in Figure 2.6. As can be seen, the signal filtered by the PAM filter does not pass through the sample points here. This is because the PAM filter will be used again at the receiver as a matching filter. The bandwidth is decided by how many symbols per seconds that are sent. For every symbol per second, 1 Hz of bandwidth is required, e.g. 1 Msymbols/s require 1 MHz of bandwidth. The PAM filter will also increase the used bandwidth.

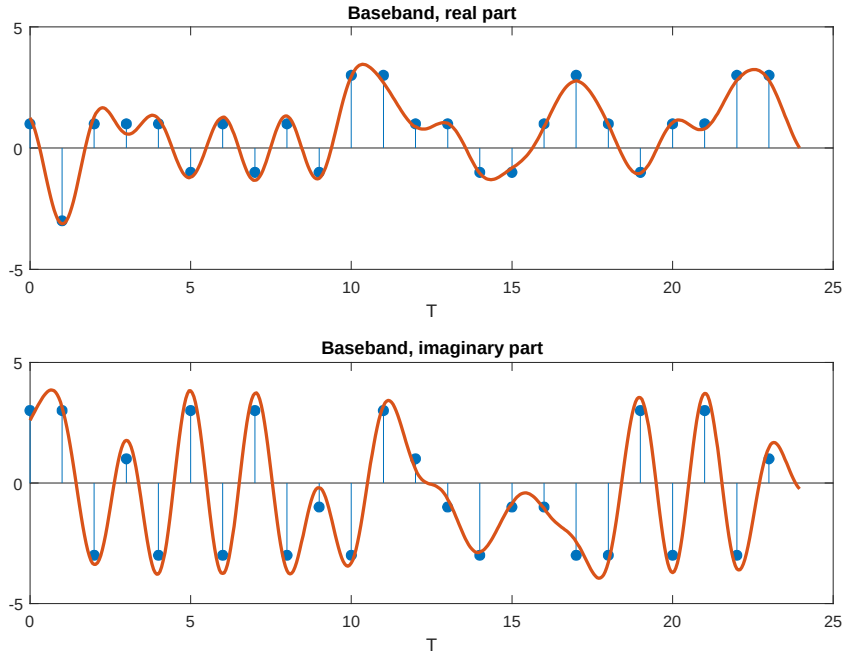


Figure 2.6: *Baseband signal before (blue) and after PAM filter (red).*

The passband signal, $s_c(t)$, is created by multiplying the baseband signal with two carriers and added together.

$$s_c(t) = \cos(2\pi f_c) I(t) - \sin(2\pi f_c) Q(t) \tag{1}$$

The amplitude modulation of the carrier that lags the other by 90 degrees is called the in-phase component, I(t), and is the real part of the baseband. The other one is called the quadrature component, Q(t), and is the imaginary part of the baseband. This works because a sine and a cosine with the same frequency are orthogonal, which can be seen as their inner product is 0.

$$\int_0^T \cos(2\pi f_c t) \sin(2\pi f_c t)\, dt = \frac{1}{2} \int_0^T \sin(4\pi f_c t)\, dt = 0 \tag{2}$$

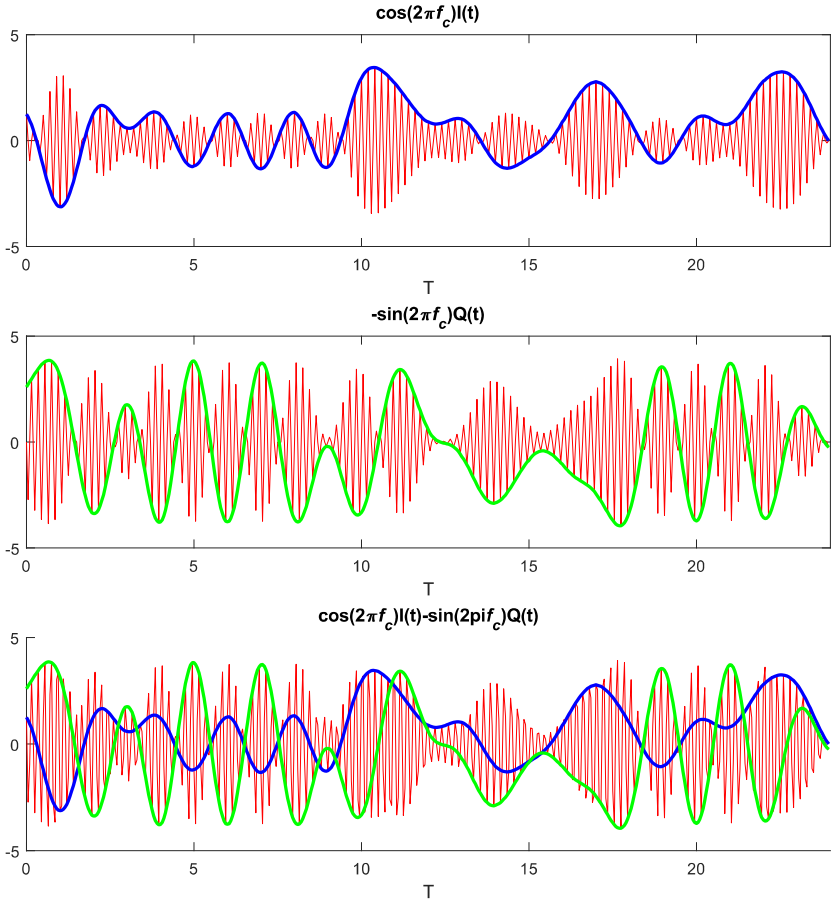


***Figure** 2.7: Carriers modulated by I(t) and Q(t) and the resulting passband signal.*

Another way to look at it is if we have digital versions of the carriers with a frequency that is half the sampling frequency. Then one period of a cosine would be [1 0 -1 0] and one period of a -sine [0 -1 0 1]. Since the none-zero samples do not overlap, they could be multiplied with any number and be added together without affecting each other. Figure 2.7 shows how the carriers are modulated by I(t) and Q(t) separately and the resulting passband signal when they are added together. To get I(t) and Q(t) back at the receiver side, $s_c(t)$ is multiplied separately by a cosine and a sine.

$$\begin{aligned} r_I(t) &= s_c(t)\cos(2\pi f_c t) = I(t)\cos(2\pi f_c t)\cos(2\pi f_c t) - Q(t)\sin(2\pi f_c t)\cos(2\pi f_c t) \\ r_Q(t) &= -s_c(t)\sin(2\pi f_c t) = -I(t)\cos(2\pi f_c t)\sin(2\pi f_c t) + Q(t)\sin(2\pi f_c t)\sin(2\pi f_c t) \end{aligned} \quad (3)$$

This can be rewritten as:

$$
\begin{aligned}
r_I(t) &= \frac{1}{2} I(t)[1+\cos(4\pi f_c t)] - \frac{1}{2} Q(t)\sin(4\pi f_c t) \\
&= \frac{1}{2} I(t) + \frac{1}{2}[I(t)\cos(4\pi f_c t) - Q(t)\sin(4\pi f_c t)] \\
r_Q(t) &= \frac{1}{2} Q(t)[1-\cos(4\pi f_c t)] - \frac{1}{2} I(t)\sin(4\pi f_c t) \\
&= \frac{1}{2} Q(t) - \frac{1}{2}[Q(t)\cos(4\pi f_c t) + I(t)\sin(4\pi f_c t)]
\end{aligned}
\tag{4}
$$

As can be seen, only I(t) or Q(t) appear in the baseband. The rest that appear at twice the carrier frequency can then easily be filtered away by a lowpass filter. The I and Q components can then be sampled by an ADC and by using the same PAM filter that was used by the transmitter as a matching filter and then downsample the signal, the symbols will be restored, see Figure 2.8.

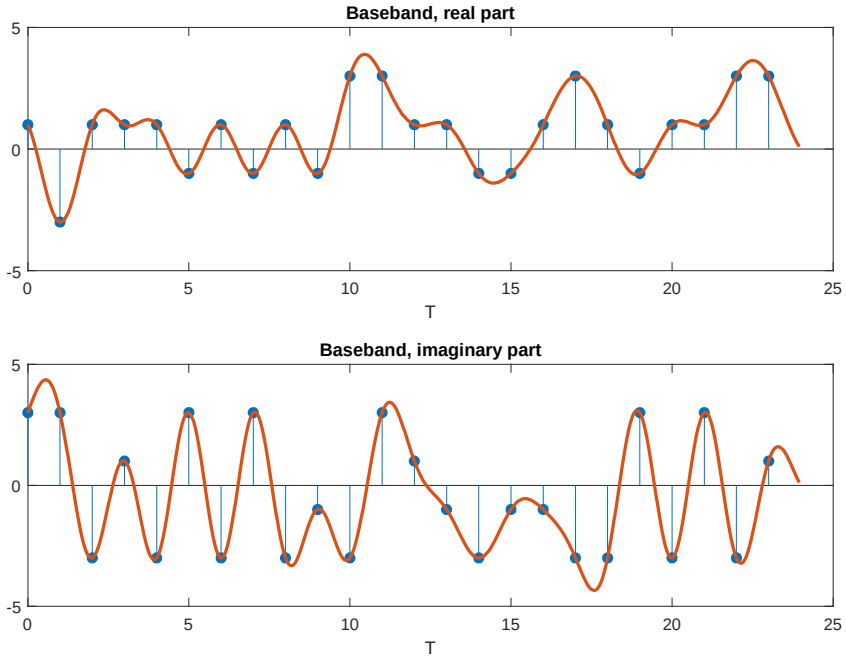


***Figure** 2.8: Baseband signal after matching filter (red) and downsampling (blue).*

In reality there will always be interference and noise in the received signal. Therefore, the received symbols will spread out, see Figure 2.9. As long as the symbols stay within the red decision lines they are received correctly, otherwise errors start to occur. These errors could to a certain degree be corrected if an error correction code was used by encoding the message with redundant information. 64-QAM and 256-QAM are more sensitive to noise if the transmission power stays the same, as they have more points that are closer to each other and therefore also narrower decision borders.

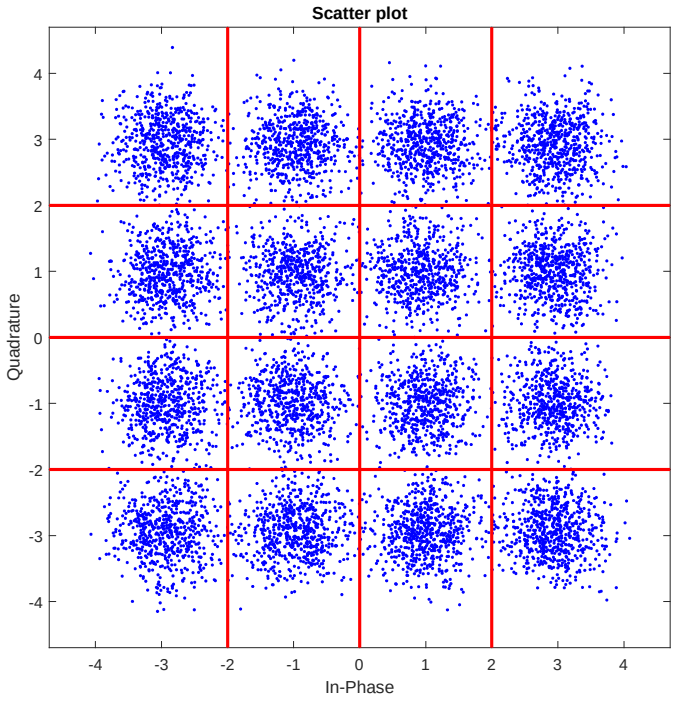


***Figure** 2.9: Symbols affected by noise and interference.*

2.4 Pulse-Amplitude Modulation, PAM

The symbols are encoded as a series of samples, s_i, in a discrete time signal, s[n]. In a time-continuous signal, s(t), we want to fill in what is between these samples by using a pulse-shape function, p(t), as a filter so that $s_i(t) = s_i \cdot p(t)$. This is called Pulse-Amplitude Modulation, PAM, since the symbols modulate the amplitude of pulses [8].

Due to bandwidth limitations these pulses must have a certain shape. A rectangular pulse would have infinite bandwidth. The most effective use of the bandwidth is to use Nyquist pulses, which are sinc functions. A Nyquist pulse has a rectangular function in the frequency domain and has a bandwidth of 1/T Hz, which would let us send one symbol per second for every Hz of bandwidth. One of the main problems with using Nyquist pulses is that a sinc function is non-causal, it extends for an infinite amount of time in both negative and positive direction. By using more bandwidth, a compromise can be made. Another type of pulse is the raised cosine pulse whose transition band resembles a cosine wave.

A raised cosine pulse has the spectrum given by

$$H(f) = \begin{cases} T, & |f| \leqslant \frac{1-\alpha}{2T} \\ \frac{T}{2}\left(1+\cos\left(\frac{\pi T}{\alpha}\left(|f|-\frac{1-\alpha}{2T}\right)\right)\right), & \frac{1-\alpha}{2T} < |f| < \frac{1+\alpha}{2T} \\ 0, & \textit{elsewhere} \end{cases} \tag{5}$$

and is visualized in Figure 2.10.

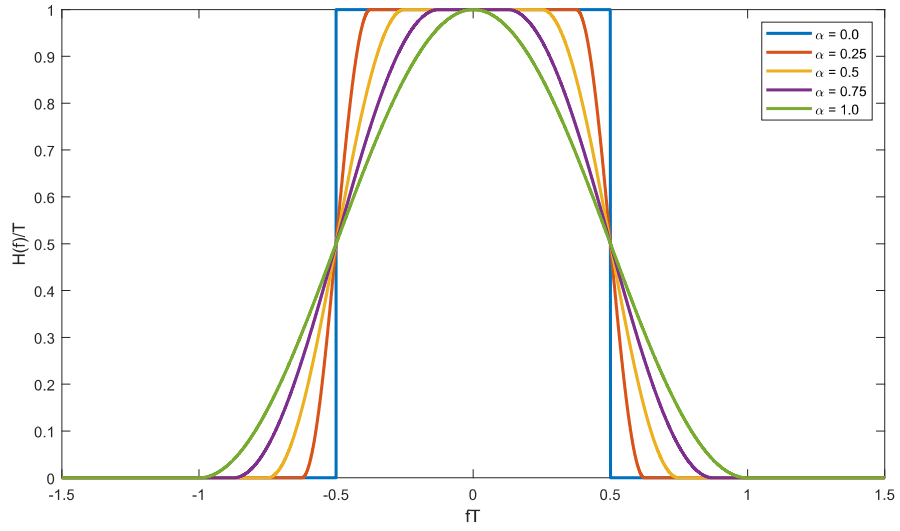


***Figure** 2.10: Spectrum of raised cosine pulses.*

The impulse response is given by

$$h(t) = \begin{cases} \frac{\pi}{4T}\,sinc\left(\frac{1}{2\alpha}\right), & t = \pm\frac{T}{2\alpha} \\ sinc\left(\frac{t}{T}\right)\frac{\cos(\alpha\pi t/T)}{1-(2\alpha t/T)^2}, & \textit{elsewhere} \end{cases} \tag{6}$$

and is visualized in Figure 2.11.

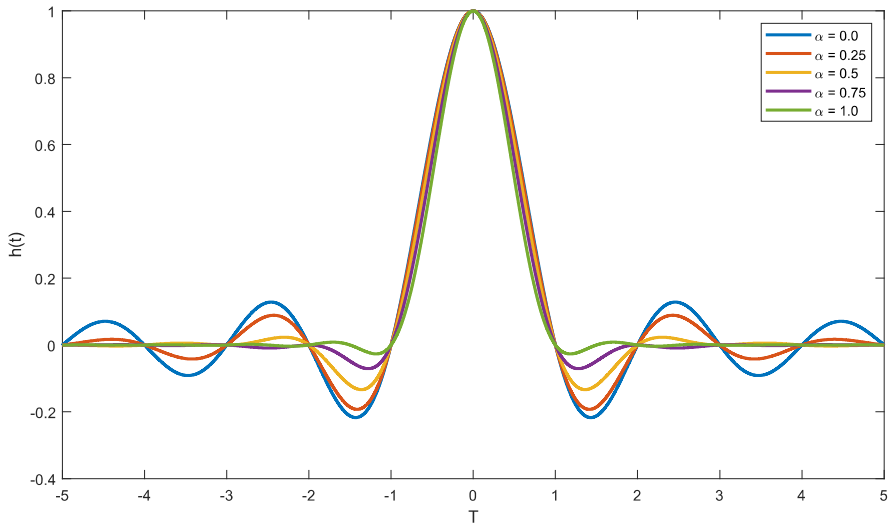


***Figure** 2.11: Raised cosine pulses in time domain.*

The shape of a raised cosine pulse resembles a sinc pulse but it goes towards zero quicker. The roll-off factor, α, is the excess bandwidth and with more bandwidth the pulse goes towards zero quicker. A raised cosine pulse is also non-causal but since it quickly goes towards 0 for moderately large $|t|$ a delayed truncated version can be used. Depending on how much it is truncated, the side-lobes, which are unavoidable due to the truncation, become more or less large. A longer pulse would have a better stop-band attenuation but also require more filter coefficients and therefore more computation power. Sinc and raised cosine pulses fulfill the Nyquist ISI criterion so that when one pulse is at its maximum, the other pulses do not interfere since their amplitude are always zero then. When designing the pulse-shaping filter, a root raised cosine pulse will be used instead whose frequency response is the square root of the frequency response of the raised cosine filter. The same filter will be used as a matching filter on the receiver side. The root raised cosine filter on the transmitter side will cause ISI but when combined with the matching filter at the receiver this will disappear as the product of two root raised cosine filter is equal to a raised cosine filter.

The Pulse-Amplitude Modulation and the matching filter is done in digital domain, since analog versions of these filters would have too high requirements. Since the spectrum of x[n] goes all the way up to the Nyquist frequency, as seen in Figure 2.12 a), it has to be upsampled by a factor M, which results in y[m]. This results in y[m] where the additional spectrum let us use the PAM filter, p[m]. The spectrum, when M = 4, of y[m] and p[m] is shown in Figure 2.12 b). The resulting signal, z[m], is the pulse-amplitude modulated version of x[n] and its spectrum is shown in Figure 2.12 c).

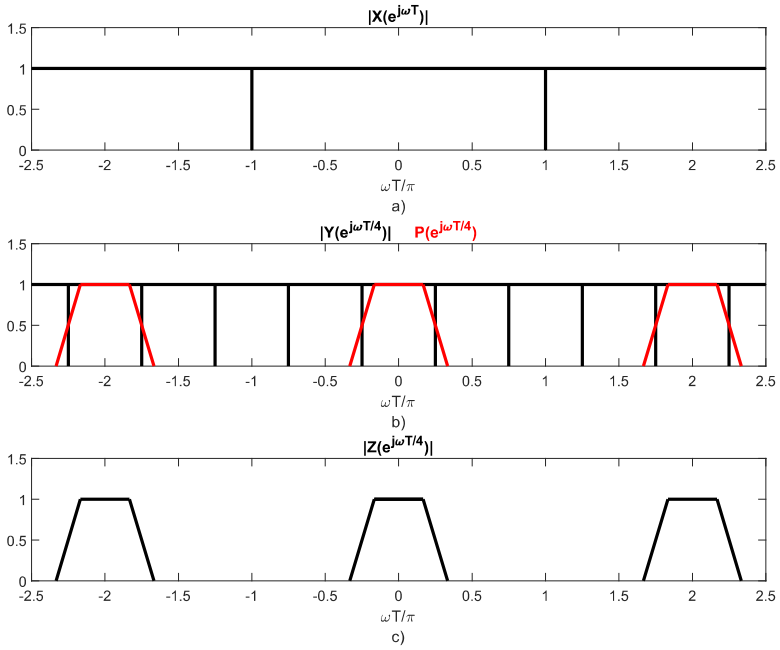


***Figure** 2.12: (a) Digital baseband signal, (b) Upsampled baseband and filter, (c) Filtered baseband.*

2.5 Pulse Width Modulation, PWM

Pulse width modulation, PWM, is a method to modulate a signal as a series of rectangular pulses with fixed amplitude and frequency by changing the width of these [9]. These pulses have a period time, T, and by controlling the pulse duration within this time we can get an average value that represents the value of the modulating signal during this time. The pulse width is usually described by the term *duty cycle* and expressed in percent where 100% is a width equal to the whole period, see Figure 2.13.

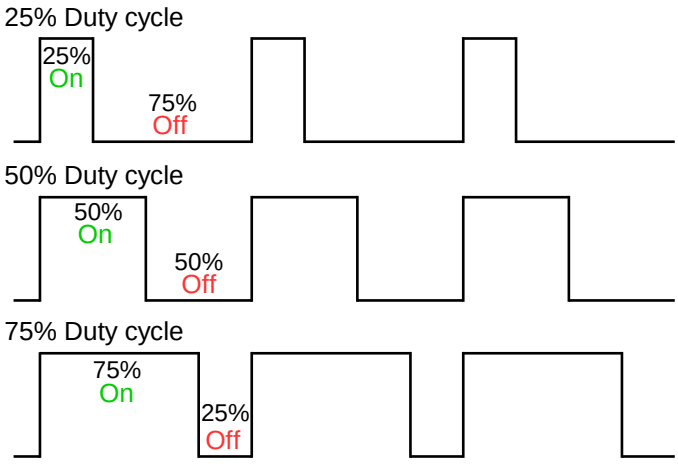


Figure 2.13: *Pulse width modulation, different duty cycles.*

One way to create a PWM-signal is to use the intersective method. With this method we have to start with a source signal, here consisting of four tones, 500 Hz, 1000 Hz, 1500 Hz and 2000 Hz, each with an amplitude of 0.25, the blue curve in Figure 2.14a. This source signal is compared with a modulating waveform, the red wave in Figure 2.14a, with frequency that is higher than the highest frequency of the source signal. This waveform can be a sawtooth or as here, a triangle wave. This triangle wave has a frequency of 20 kHz, which gives a 50 µs period time for the pulses. If the source signal is higher the modulating waveform, than the PWM-signal is high, otherwise it is low. The resulting PWM signal is seen in Figure 2.14b.

The PWM-signal contains high frequency components which need to be filtered out, see Figure 2.14c where a part of the spectrum is seen. These unwanted frequencies start to appear around the same frequency as the modulating waveform so a higher frequency for the modulating waveform will move these to higher frequencies. When doing this in the digital domain the sampling frequency will also affect the spectrum since the width of the pulses will have ha finite resolution. This will limit how high frequency the modulating waveform can have.

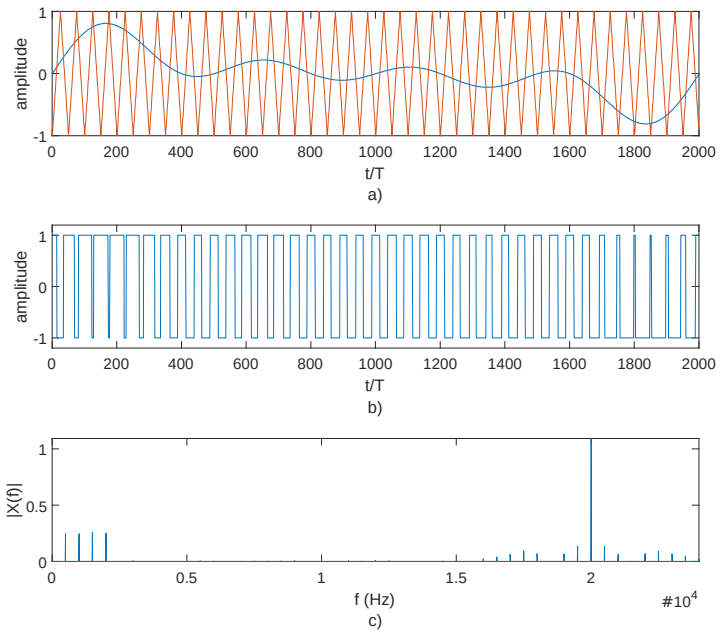


***Figure** 2.14: a) Source signal and triangle wave, b) resulting PWM-signal and c) spectrum of PWM-signal.*

In this thesis the receiver will sample the RF signal as a PWM-signal using the same method, which was proven to work in [6]. For this to work the frequency of the reference signal, f_r, has to be at least the same as the bandwidth of the RF signal if a triangle wave is used, or twice the bandwidth if sawtooth wave is used. This is because copies of the RF spectrum will appear at frequencies bellow and above the center frequency, f_c, after sampling. There are also harmonics around $3f_c$, $5f_c$,... that will fold down due to aliasing after sampling. Depending on carrier, reference and sampling frequency these harmonics could therefore appear at the RF signal and cause distortion, see Figure 2.15. The ideal frequency of the reference wave has to be found for each carrier frequency to minimize the distortion caused by the sampling.

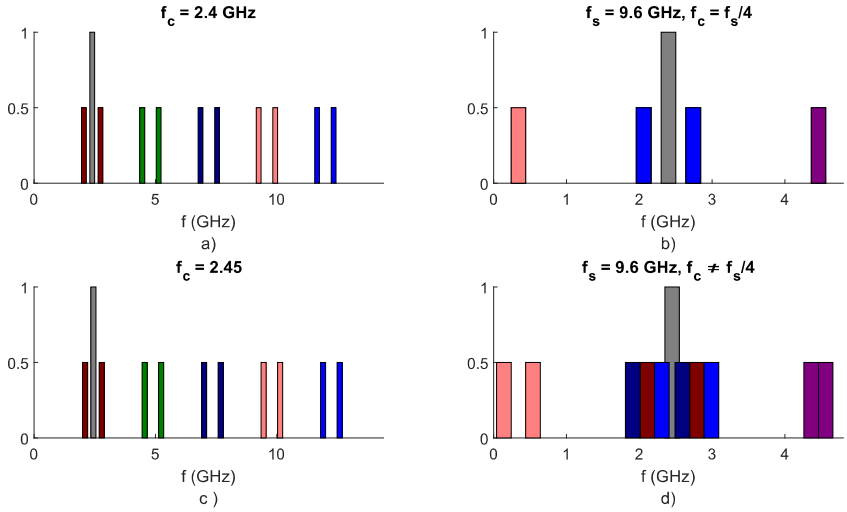


***Figure** 2.15: a) PWM spectrum before sampling, f_c = 2.4 GHz, b) PWM spectrum after sampling when $f_s = 4f_c$, c) PWM spectrum before sampling, f_c = 2.4 GHz, d) PWM spectrum after sampling when $f_s \neq 4f_c$.*

2.6 Polyphase DDC

The data from the transceiver comes in parallel with N 1-bit samples every clock cycle and contains the RF signal. This signal needs to be down converted directly from the RF frequency to a complex baseband signal with a sample rate that is 1/N of the of the original sample rate by using a digital down converter, DDC. A DDC consists of direct digital synthesizers (DDS), lowpass filters and downsamplers, see Figure 2.16. The quadrature DDS creates a two local oscillator (LO) consisting of a sine wave and a cosine wave where their frequencies are equal to the carrier frequency. These waves are multiplied with the sampled RF signal and, depending on the phase of the wave, the I or Q component of the RF signal will move down to baseband. Since the complex baseband signal is down sampled by a factor N, a lowpass filter is required to avoid aliasing.

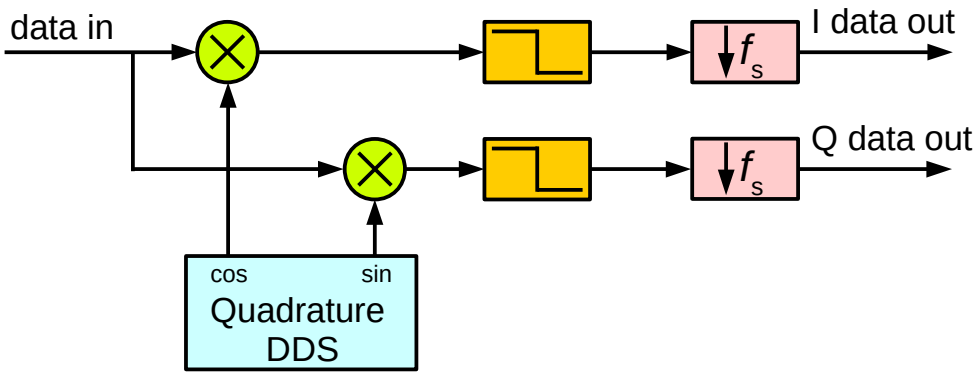


Figure 2.16: *DDC block diagram.*

Since only every N sample will be kept after the down sampling and the rest discarded, a polyphase implementation can be used [10]. The FIR filter can be split up into N smaller filters that work in parallel and N DDS would have to be used to create N LO with different phases. For the FPGA implementation this is more or less required to avoid an unnecessary complex DDC since the transceiver clock would run at N times higher frequency than the FPGA clock. The FPGA would have to calculate N samples every clock cycle by using N lowpass filters with all coefficients included. Figure 2.17 shows an example where the resulting baseband signal is downsampled by a factor 4 and the original FIR filter containing 8 coefficients is split into 4 smaller filter containing 2 coefficients each.

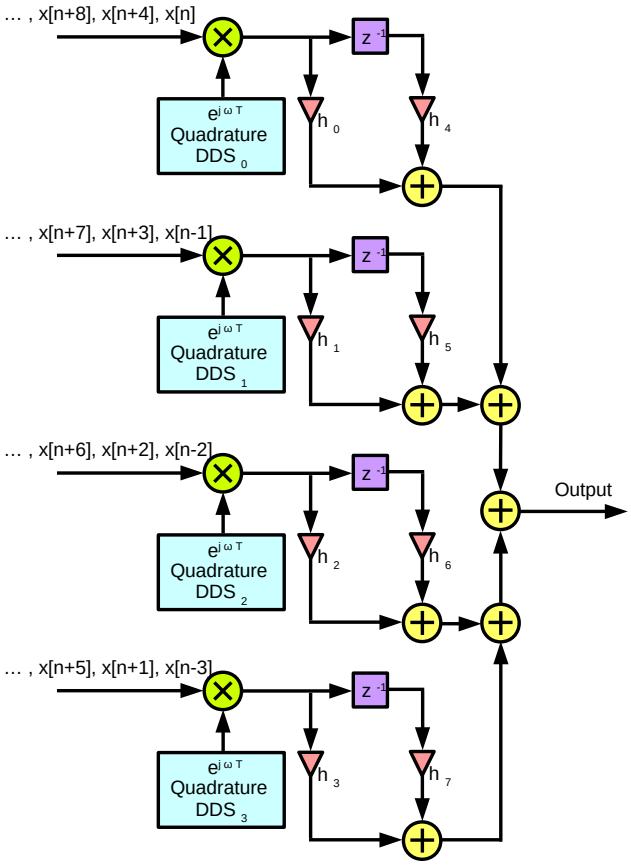


Figure 2.17: *Polyphase DDC that downsample by a factor 4.*

3 Method

3.1 Introduction

In this thesis, MATLAB, Quartus Prime and ModelSim were the software which were mainly used. MATLAB was used to simulate a mathematical model of the communication system and to create test signals and a pulse-width modulated triangle wave. Quartus Prime was used to write VHDL code and compile the code. To be able to work with the Intel Stratix 10 FPGA, the Pro version of Quartus Prime was needed, which only supports the Stratix 10 FPGA. Since the compilation time for the Stratix 10 TX was at least around 15 minutes, a DE10-lite board with an Intel MAX 10 FPGA was also used with the lite version of Quartus Prime. Minimum compilation time for this FPGA was less then one minute which saved a lot of time when VHDL code that did not require the transceivers was tested. The lite version of Quartus Prime also had some helpful features that the Pro version did not have. ModelSim was only used for simulation of the VHDL code written in Quartus Lite.

A model of the receiver was made in MATLAB first, where the behavior of the receiver was simulated with different parameters. The receiver was then built in Quartus Prime and simulated in ModelSim to verify that the output was the same as the model in MATLAB. When it was verified that the VHDL code worked, tests on the FPGA were performed, first with test signals made in MATLAB and then tests with external test signals.

3.2 MATLAB

QAM signals upconverted to RF could be created in MATLAB by using its included functions, see Figure 3.1. First an array of symbols represented by integers had to be created. With 64-QAM every symbol consists of 6 bits and in decimal these symbols can have values between 0 and 63. The number of possible symbols is also called “alphabet size”. When many symbols were needed, the function “randi()” was used to create an array of random integers. To get the same numbers each time the function “rng()” was used to control the random number generator. The function “qammod()” was then used to convert the array of symbols to an array of complex number, which was the digital complex baseband signal where each sample corresponds to a symbol.

Before the PAM filter could be applied, the baseband signal had to be upsampled. To simplify things the baseband was upsampled to the same sampling frequency that was used on the FPGA, 150 MHz, which gave a upsampling factor of 10 at a symbol rate of 15 MHz. The same PAM filter could then be used as a matching filter at the receiver side. The PAM filter was created by using the function “rcosdesign()”, which returned the impulse response of a root-raised-cosine filter. The roll-off value was chosen to 1/3, which gave a passband bandwidth of 20 MHz. To get a stop-band attenuation of at least 35 dB the filter had to span 10 symbols, which gave a filter of order 100.

The upconversion to RF was done by first upsampling the baseband signal by a factor of 64, which gave the same sampling frequency as the transceiver would use, 9.6 GHz. After the lowpass filter the signal was multiplied by $e^{j\omega cT}$ to move the baseband up to positive frequencies and then was the function “real()” used to get the negative frequencies and thus create a real RF signal. The lowpass filter was of the same type as the PAM filter, a root-raised-cosine filter. The filter was designed in a similar way as the PAM filter, a roll-off factor of 1/3 her also gave a total bandwidth of 100 MHz and the filter length was chosen so that the stop-band attenuation was at least 35 dB.

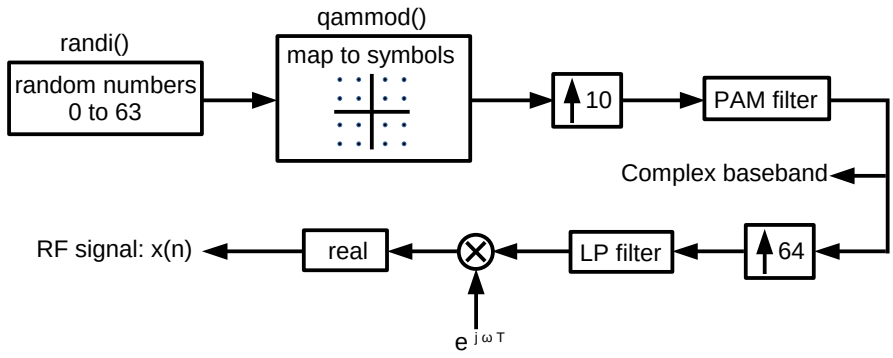


Figure 3.1: *Method to create QAM signals at radio frequency in MATLAB.*

A MATLAB model was created before the VHDL implementation to simulate the system and measure the performance with different design parameters like carrier frequency, sampling frequency, symbol rate, etc, see Figure 3.2. The inputs to the comparator was the RF 64-QAM, x(n), signal and a reference triangle wave, r(n), with the same sampling frequency as the RF signal. If, at the time, the amplitude of x(n) was higher than r(n), the comparator would output 1, otherwise -1. This simulated how the transceiver was expected to work. The amplitude of the triangle wave was always 1 and a maximum amplitude of the RF signal was as default set to 1 but could be varied to test how sensitive the system was to different gains of the RF signal.

The resulting pulse-width modulated signal was multiplied by $\cos(2\pi f_c)$ and $-\sin(2\pi f_c)$ to downconvert the I and Q parts respectively. The resulting complex baseband signal was then filtered with the same lowpass filter that was used when creating the RF 64-QAM signal. Since the roll-off factor was 1/3, the passband was flat up to 50 MHz and frequencies between 75 and 100 MHz folded down to frequencies between 50 and 75 MHz. This gave a maximum passband bandwidth of 100 MHz for the QAM signals. After this filter the baseband signals was downsampled by a factor 64, which corresponds the same frequency used by the FPGA, 150 MHz.

The I and Q parts of the baseband was then filtered by a matching filter to remove noise and restore the symbols and then downsampled. EVM_{rms}, number of symbol errors and bit error rate were then calculated to measure the performance. MATLAB uses floating-point numbers and the VHDL

implementation used fixed-point numbers. A modified version of MATLAB model, where all filter coefficients were converted into integers, was therefore also used to simulate the expected behavior of the VHDL implementation.

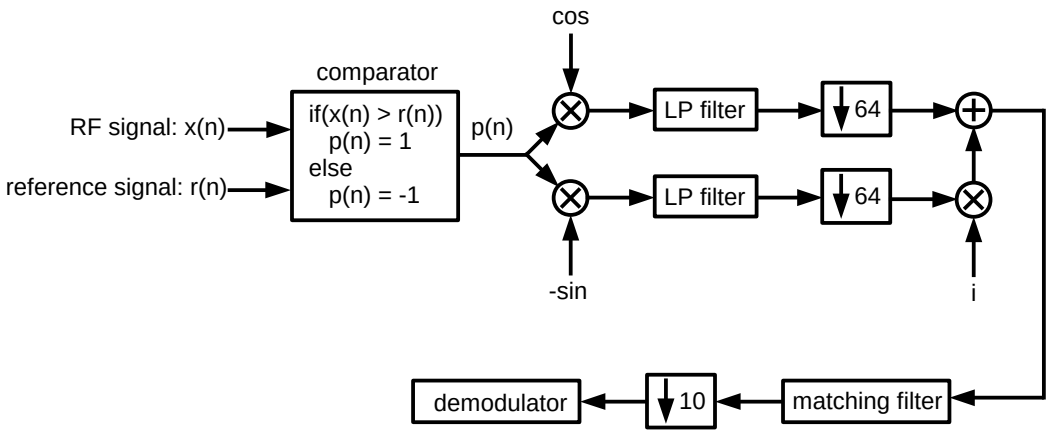


Figure 3.2: *MATLAB model of the receiver.*

3.3 Analog RF test signals

The analog RF signal was created by using the vector signal generator SMBV100A from Rohde & Schwarz. This vector signal generator can read files stored on USB memory sticks that contain complex baseband signals. The vector signal generator upconverts the baseband signal to a RF signal and repeats it. The .wv files (R&S ARB waveform file) containing the baseband signal was created with the software “R&S ARB Toolbox”. This program accepts .txt files containing discrete-time signals with Q and I components. The baseband signal was created in MATLAB as described in section 3.2. The real and imaginary parts were then separated into a matrix and from there copied into a .txt file.

3.4 Triangle reference wave

The triangle wave can be described as the sum of a fundamental frequency and its odd harmonics, see Equation (7). Due to bandwidth limitations and that the amplitude of higher harmonics will go down quickly, 3 harmonics are enough. The triangle wave can then be approximated as a sum of 4 frequencies, see Equation (8).

$$x_{triangle}(t)=\frac{8}{\pi^2}\sum_{i=0}^{N-1}(-1)^i(2i+1)^{-2}\sin(2\pi f_0(2i+1)t) \tag{7}$$

$$x_{triangle}(t)=\frac{8}{\pi^2}\left(\sin(2\pi f_0 t)-\frac{1}{9}\sin(6\pi f_0 t)+\frac{1}{25}\sin(10\pi f_0 t)-\frac{1}{49}\sin(14\pi f_0 t)\right) \tag{8}$$

To create the reference signal, it was first needed to find the best frequency. This was done by simulating the system in MATLAB with different reference frequencies and finding the lowest EVM_{rms}. This frequency would be optimized for an RF signal with 20 MHz bandwidth and a carrier frequency of 2.4 GHz. The reference signal would be pulse-width modulated later with a sampling frequency of 28.9 GHz. To make it easier, one period of the pulse-width modulated reference wave should use a number of bits that is an integer so that just one period has to be stored in VHDL code and then be repeated. 1000 to 3000 bits per period were tested, which would equal frequencies between 9.63 to 28.9 MHz. In Figure 3.3 the resulting EVM_{rms} is shown for each possible frequency. As can be seen, a frequency of 17 MHz works well and would use 1700 bits per period.

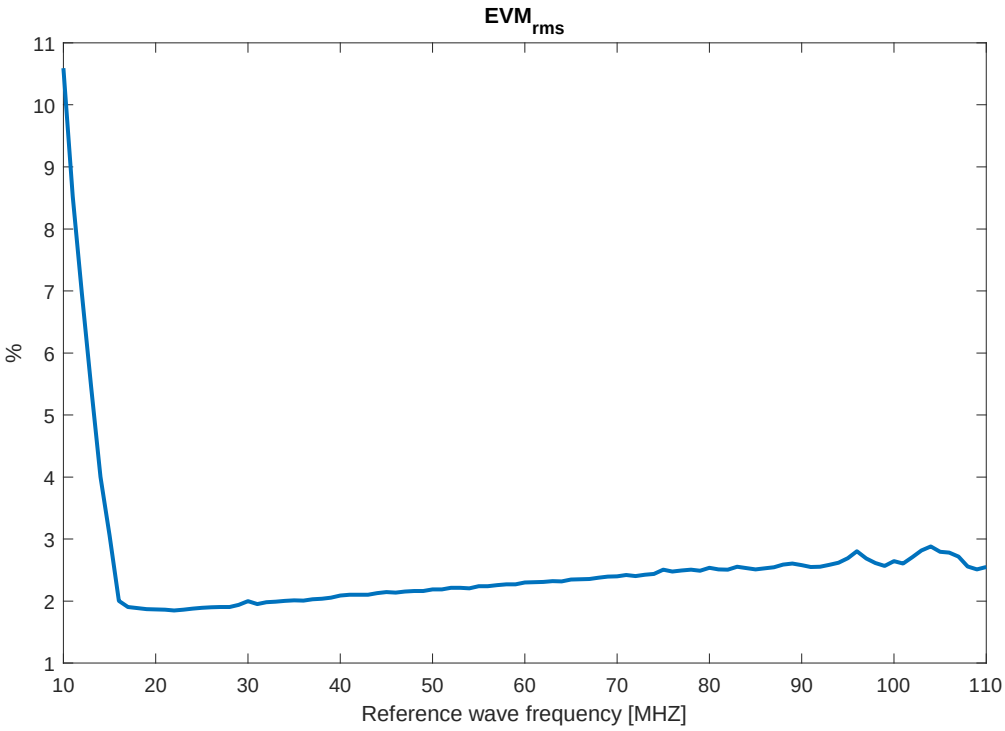


***Figure** 3.3: EVM_{rms} for different reference frequencies.*

The pulse-width modulation was done in MATLAB and also here a reference wave was needed to get the best SNR for the triangle wave. Since only one period will be stored later on the FPGA the frequency of the reference wave had to be an integer multiple of the frequency of the triangle wave, otherwise the reference wave would stop in the middle of one period and restart. The SNR was measured by doing the pulse-width modulation and then use a lowpass filter to remove frequencies above the third harmonic. After running several simulations with different frequencies, 341 MHz was the lowest good frequency to use, see Figure 3.4. A comparison between the original triangle wave and the filtered pulse-width modulated version, which is expected to be seen on the oscilloscope later, are shown in Figure 3.5. The sequence of ones and zeros was later used in VHDL code.

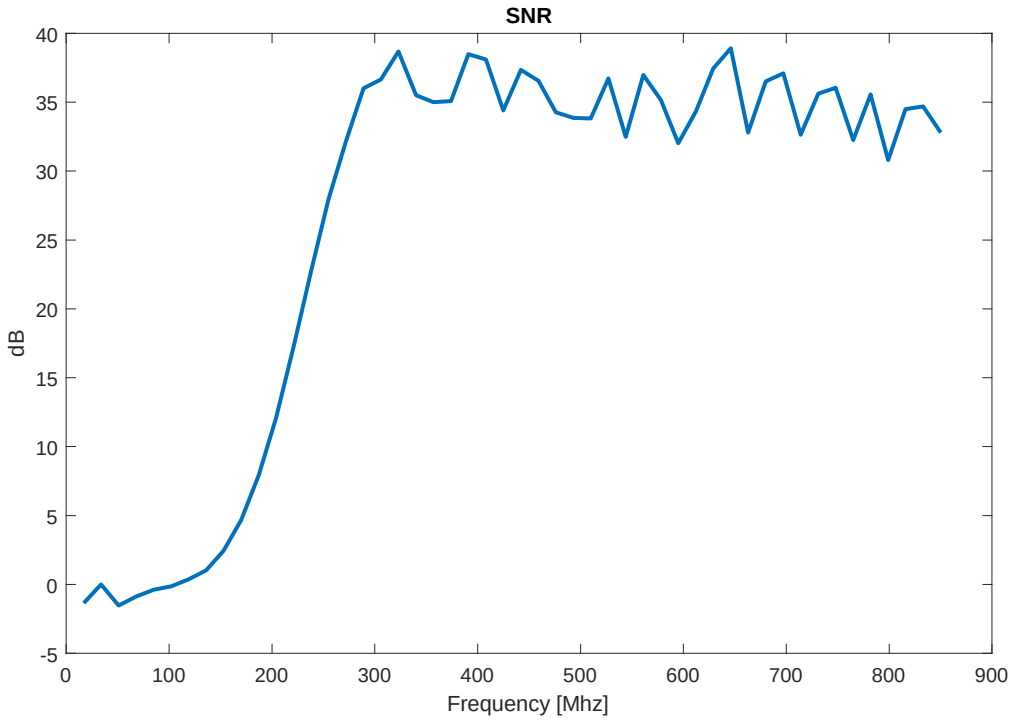


***Figure** 3.4: SNR for different reference frequencies.*

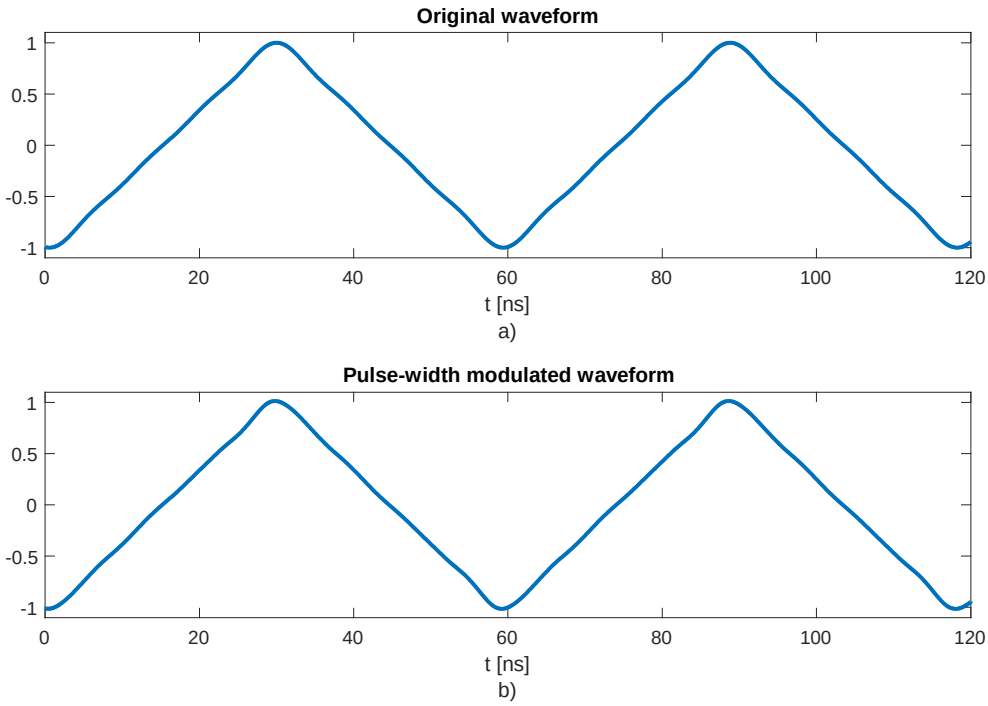


***Figure** 3.5: (a) the original triangle wave with 3 harmonics and (b) pulse-width modulated triangle wave after the lowpass filter.*

3.5 Implementation – FPGA

3.5.1 Receiver

The receiver implemented on the FPGA consists of a transceiver block, two DDC filters, two PAM filters, two down samplers, two delta-sigma modulators and a triangle wave generator, see Figure 3.6. The two DDC filters from the MATLAB model downconvert, lowpass filters and downsamples the PWM-sampled RF signal. As the outputs from the lowpass filters had to be down-sampled by a factor of 64, the filters only needed to be calculated every 64^{th} output sample. This makes it possible to multiply the coefficients with cosines and sines since one period fits exactly in 4 samples, which 64 is evenly divisible by, and do the downconversion at the same time. This actually transforms the lowpass filters into bandpass filters as seen in Figure 3.7. Another way to look at the downconversion is that the DDC filters bandpass filter the RF signal, remove the unwanted I or Q component and then fold down the passband to baseband due to aliasing.

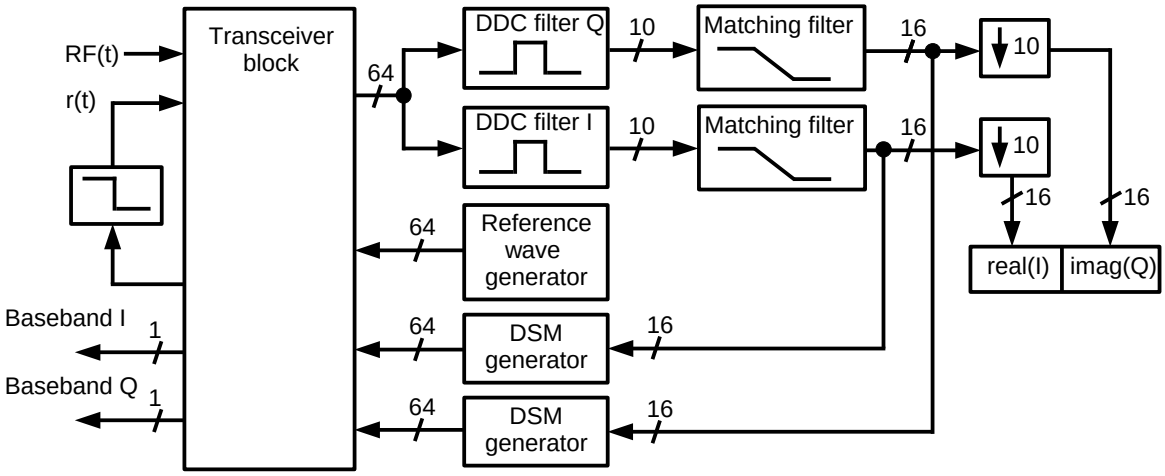


Figure 3.6: *FPGA receiver architecture.*

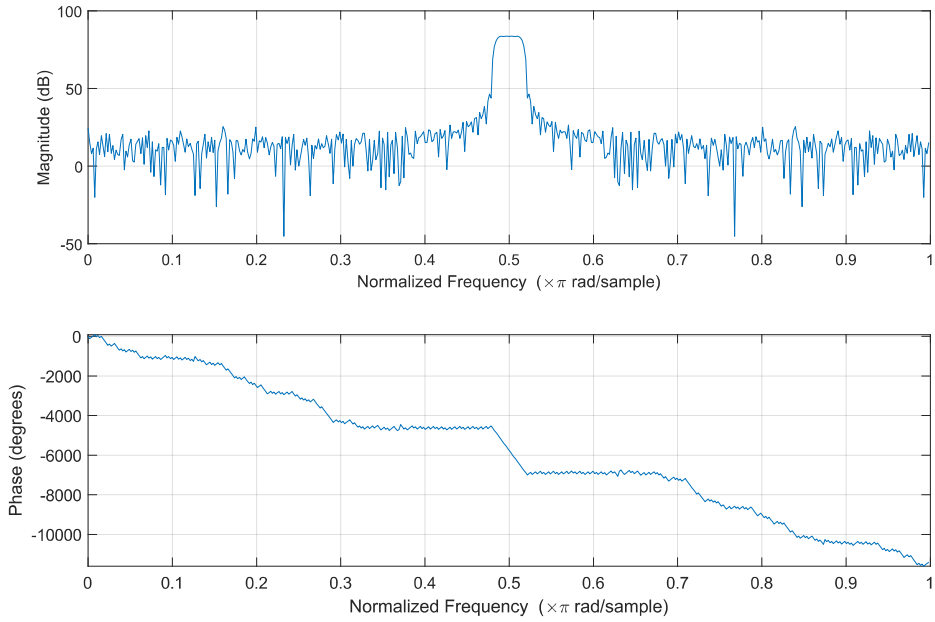


Figure 3.7: *Frequency and phase response of DDC filters.*

The inputs to the DDC filters are either one or zero, therefore are no multipliers needed since the output either will be the coefficient or the coefficient times negative one. The inputs to the DDC filters therefore control how the coefficients will be added together. The coefficients implemented on the FPGA for the DDC and PAM filters used fixed-point representation instead of the double

precision floating-point representation used in MATLAB. The number of bits decides how much quantization noise the filters will add. Figure 3.8 shows how the number of bits used for the coefficients affects the EVM_{rms}. As can be seen, 10 bits are enough to just have a very small effect on the EVM_{rms}. With signed integers the coefficients can have a value between [-512, 511]. The filters were scaled so that the coefficient with the largest value became 511 and the rest were rounded to closest integer.

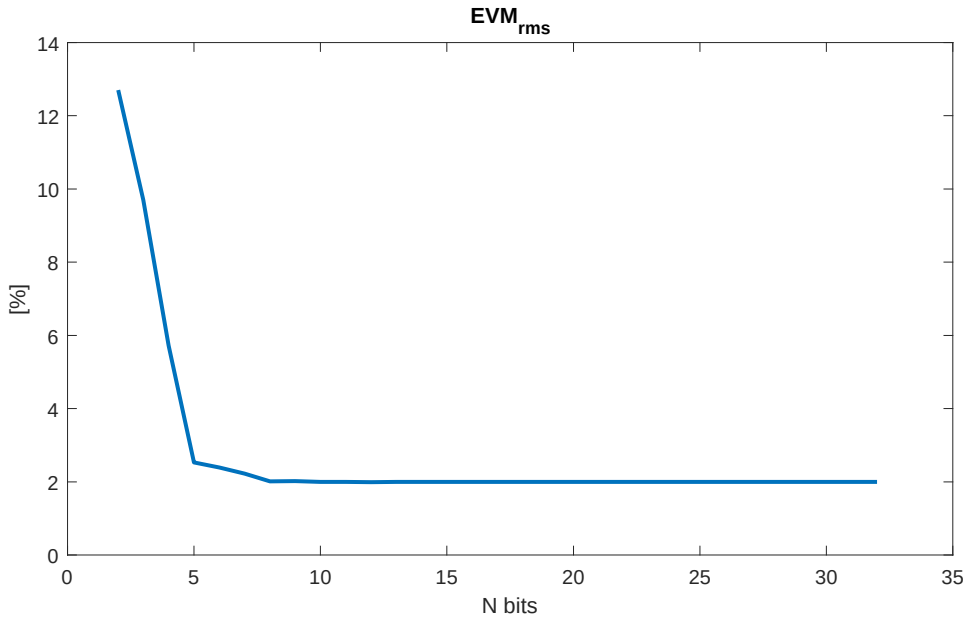


Figure 3.8: *EVM_{rms} as a result of the number of bits used for the DDC and PAM filter coefficients.*

The maximum possible absolute value from the outputs of the DDC filters was 22538 which required 16 bits to avoid overflow. Since the PAM filters also had 10 bits coefficients it would be good if the output could be truncated down to 10 bits. Figure 3.9 shows how the truncation affected the EVM_{rms} and that discarding the 6 least significant bits only had a small effect.

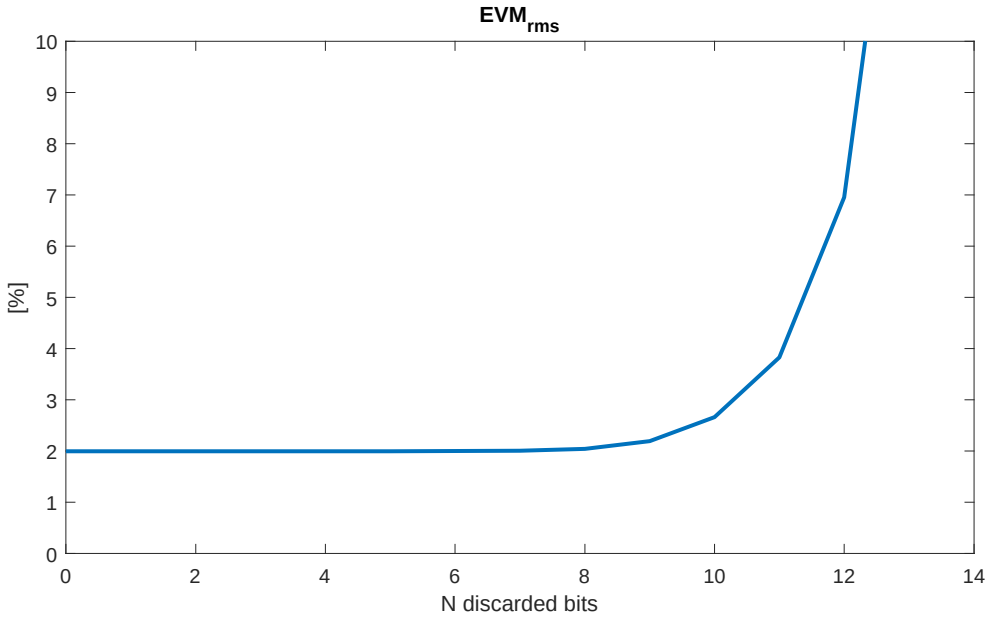


Figure 3.9: *EVM_{rms} as a result of the number of bits truncated after DDC.*

After the truncation the largest possible absolute value was 352. With this value the largest possible absolute value after the matching filters would be 2483008 which required 23 bits. If this could be truncated down to 16 bits the whole complex number could be stored with 4 bytes which fits well in 32/64 bits processors. Figure 3.10 shows how the truncation affects the EVM_{rms} and discarding 7 bits has marginal impact.

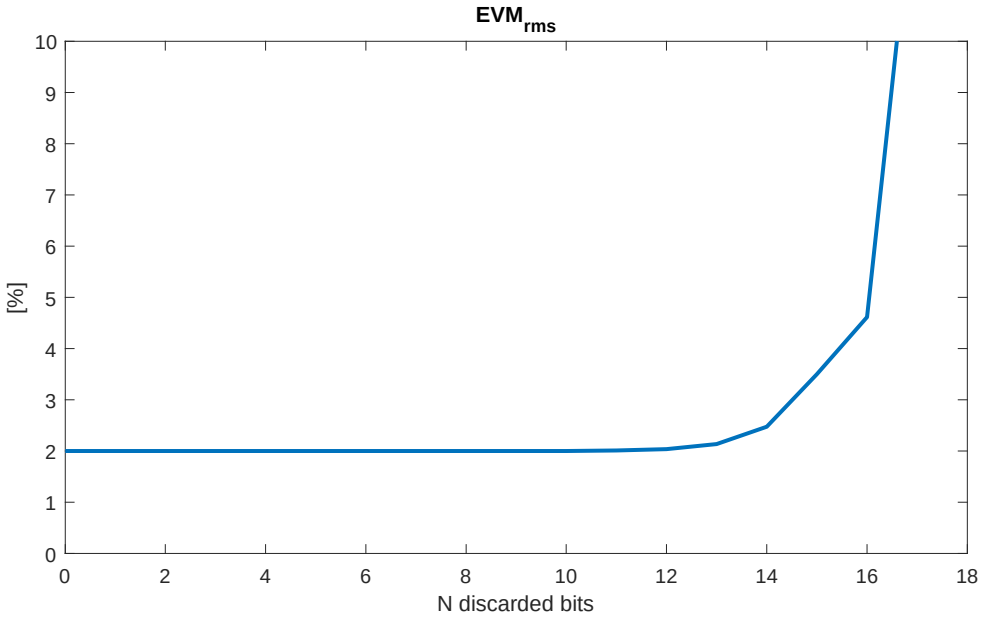


Figure 3.10: *EVM_{rms} as a result of the number of bits truncated after the PAM filter.*

3.5.2 Triangle wave generator

The triangle wave generator sends a PWM representation, 64 bits at a time, to a transceiver that transmit these bits. This signal is then lowpass filtered to recreate the triangle waveform. The PWM representation of the triangle is made as described in section 3.4. These bits were then stored in a VHDL file where a process, controlled by the parallel clock of the transceiver, sent 64 bits to the transceiver and updated those when these were sent by the transceiver.

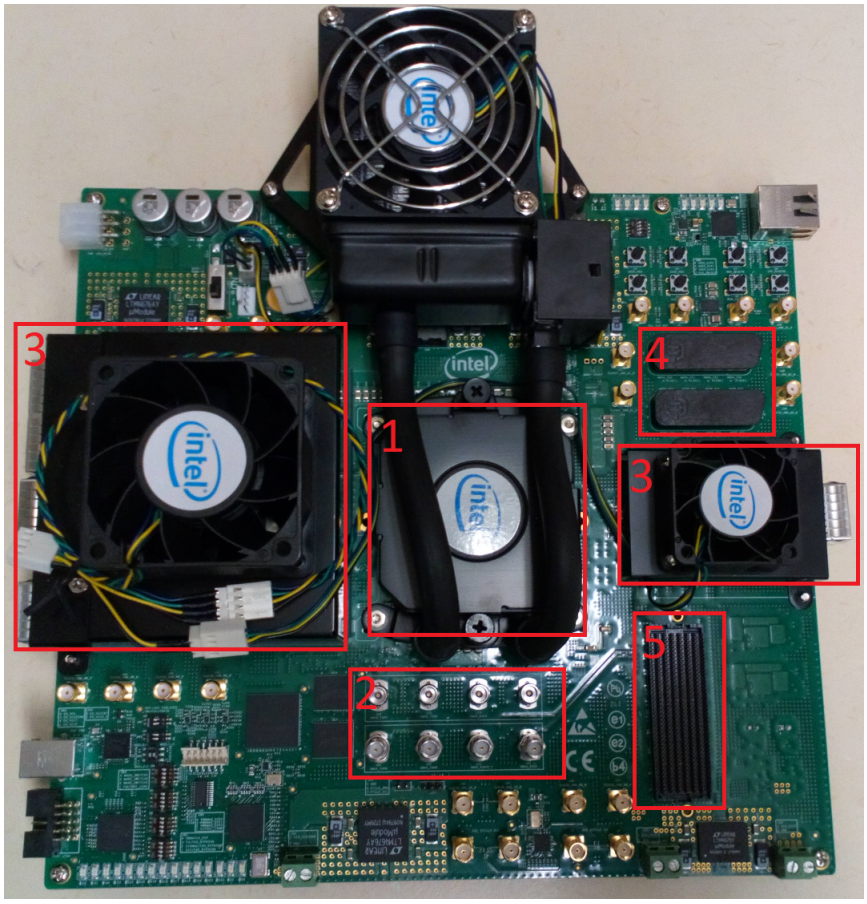
3.5.3 Getting results from the FPGA

The 32-bits complex values from the PAM filters should preferably have be transferred to a PC and analyzed there. Since there was no time to get data transfer through Ethernet or USB working, another way had to be found to get some information that validated the implemented receiver. By using a method similar to delta-sigma modulation, two transceivers could transmit the real part and the imaginary part of the symbols. An oscilloscope could then plot the real part on the x-axis and the imaginary part on the y-axis and display the symbols on a constellation diagram.

The transceivers take 64 bits in parallel and by controlling how many of these bits are ones and how many are zeros, the average value can be changed between 0 and 1 in 65 steps by using something like delta-sigma modulation. With 15 Msamples/s the transceiver could send 640 bits during one

symbol time and by changing the parallel inputs to the transceiver the average value over one symbol time could be changed in in 641 steps. A simpler method was to have 65 steps represented by 65 different vectors in VHDL and repeat those during a symbol time. This was enough to see if the receiver worked or not.

3.5.4 Transceiver hardware setup



Figur 3.11: *Intel Stratix 10 TX Development Board, 1) FPGA, 2) SMA-connectors, 3) QSFPDD-connectors, 4) MXP-connectors, 5) FMC-connector.*

The Intel Stratix 10 TX is equipped with 144 transceivers with full-duplex channels that support transfer rates up to 58 Gbps in the PAM4 mode and up to 28.9 Gbps in the NRZ mode. Up to 64 bits can be communicated in parallel between the transceiver and the FPGA which would limit the maximum FPGA clock speed to around 452 MHz if the highest transceiver speed is used; (28900 / 64 MHz).

The Stratix 10 TX comes with five different transceiver configurations with between one and six transceiver blocks. The blocks consist of transceivers of type E-tile or H-tile. In this thesis the

difference between them will not be deeply discussed. The main differences are that E-tile transceivers are slightly faster in NRZ mode, (28.3 Gbps for H-tiles), and support PAM4. The E-tile transceivers are easier to get working since only one IP-block is needed. The H-tile transceivers need three IP-blocks, one for the transceiver, one for the PLL and one for the reset function. The transceiver configuration of the Stratix 10 TX used in this consists of five E-tile blocks and one H-tile block. Each block has 24 full-duplex channels which gives a total of 144 channels. Each channel is hardwired to a certain pin on one of the connectors on the board. Except from the eight 2.4 mm RF connectors that will be used in this thesis, there are also ten QSFPDD connectors, two MXP connectors and one FMC connector.

The transceiver hardware is accessed by using IP-blocks in Modelsim. H-tiles transceivers are not physically connected on this board so only E-tiles transceivers will be available to use [13]. An E-tile block can consist of up to 24 channels with differential inputs/outputs and can be configured as a RX, TX or TX/RX duplex. Figure 3.12 shows a E-tile block with one channel configured as TX/RX duplex. The data speed is controlled by a PLL that needs a reference clock. This clock could be taken from the built-in clock circuits or, which turned out to be easier to used here, from an external clock through an SMA connector.

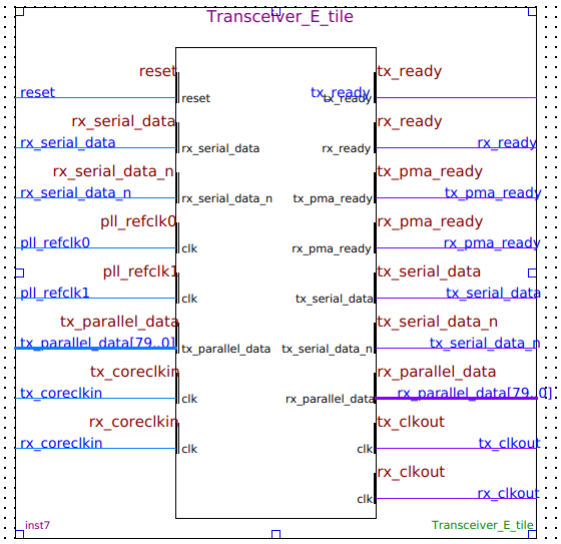


Figure 3.12: *E-tile transceiver block in Quartus.*

To transceivers had to transmit through three channels, two for the received complex symbols at 9.6 Gbps and one for the triangle wave at 28.9 Gbps. The transceiver block has an input called “tx_parallel_data”, which is an 80-bits wide bus where the data to send is placed and has to be updated when the clock output “rx_clkout” goes high. This bus has always the same width independent of how many parallel bits that are actually used. The output is differential and goes through “tx_serial_data” and “tx_serial_data_n” and are connected to one of the physical differential outputs on the board.

To receive the RF signal just one channel is needed where the RF signal and reference wave have to be connected to the positive and negative differential input for the same channel, “rx_serial_data” and “rx_serial_data_n”. When 64 bits have been received the 80-bits bus “rx_parallel_data” is updated and this is syncchronized the clock “rx_clk_out”. The problem with digital transceivers is that they are intended to be used with digital signals, like a PCI-express bus, and therefore expect a signal with two levels that change quickly at certain clock intervals. The transceivers have something called Clock Data Recovery, CDR, that uses the transition between two levels to find the clock frequency that was used at the transmitter. The transceiver uses this to adjust its own clock frequency so that it locks to the data clock in the received signal. With the analog RF signal and triangle wave as inputs there is no clock signal to be found. Therefore, the CDR has to be turned off so that the signals are compared and sampled at time intervals depending only on the transceiver clock, otherwise the transceiver would try to compensate its clock all the time and not sample anything useful. Figure 3.13 illustrates the problem. In (a) the inputs to the transceiver are shown, the 2.4 GHz carrier wave and the reference signal the carrier wave is compared to. In (b) the output from the comparator is shown together with the 9.6 GHz transceiver clock. As can be seen, the transitions between high and low after the comparator are not in phase with the rising edge of the transceiver clock.

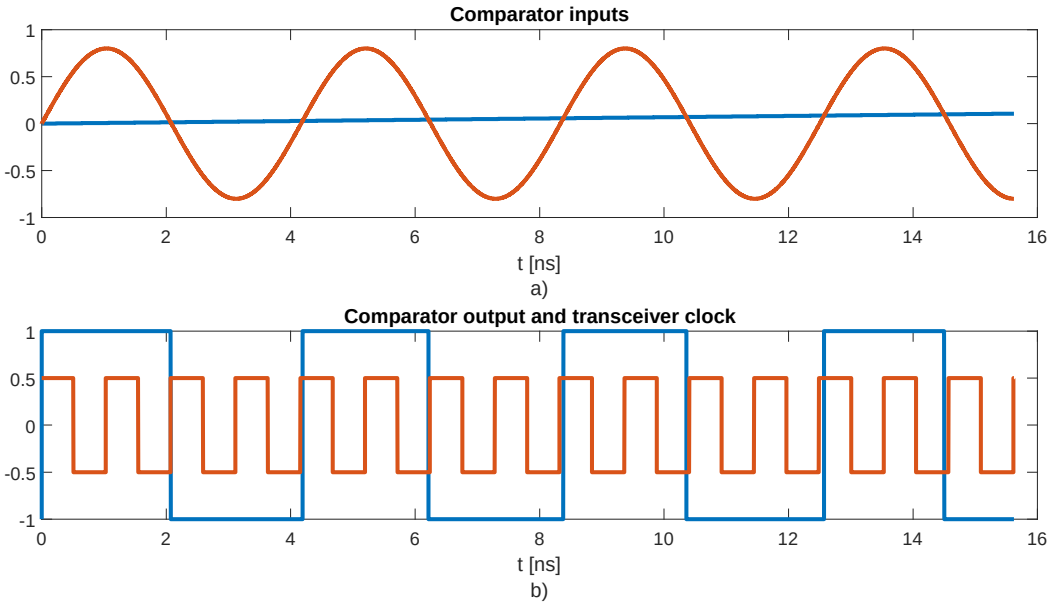


Figure 3.13: *(a) Transceiver inputs: carrier wave (red) and reference wave (blue), (b) comparator output (blue) and transceiver clock (red).*

On E-tiles transceivers, no way to turn off the CDR was found [11]. This made this transceiver type useless to sample the RF signal. For H-tile transceivers there was a setting that turned of the CDR so the incoming signal may be sampled at any time [12]. Since no H-tile transceiver was physically connected on this version of the Stratix 10 TX development kit, there was no simple way to test if H-tile transceivers could be used to sample the RF signal by using the PWM based method.

3.5.5 Test signal

To simulate the VHDL implementation in ModelSim, where no transceiver was included, the comparison was done in MATLAB. A baseband containing all 64 possible symbols in random order was created and the RF signal and sampling was done as described in section 3.2. The sampled signal was then stored in a VHDL file, which simulated what 64 bits the transceiver would output. Because the only available E-tile transceivers could not be used to sample the signal, the test signal was used to replace the transceiver in the FPGA implementation also.

3.6 FPGA test environment

The test environment illustrated in Figure 3.14 was used to test the FPGA implementation. Since the resulting complex baseband was not transmitted to a PC, only tests to see if the receiver actually could do the 1-bit PWM-based sampling could be done and that the resulting baseband resembled the baseband used to create the RF signal.

A vector signal generator was used to convert the complex baseband signal created in MATLAB to an RF signal and transmit it through a coaxial cable. This cable was connected to the positive input of one of the transceivers. Since only the RF signal was present, a bandpass filter was not necessary between the vector signal generator and the FPGA. The triangle wave was transmitted from the FPGA by one of the transceivers through a lowpass filter to the negative input of the same transceiver connected to the vector signal generator. A signal generator was used as an external clock for the transceiver PLLs. An oscilloscope connected to two of transceiver outputs was used to analyze the resulting complex baseband and the triangle wave. Since the board only had two transceivers connected to the SMA-connectors and the complex baseband output would use both, the triangle wave could not be tested at the same time. Instead a signal generator was then used to create the triangle wave.

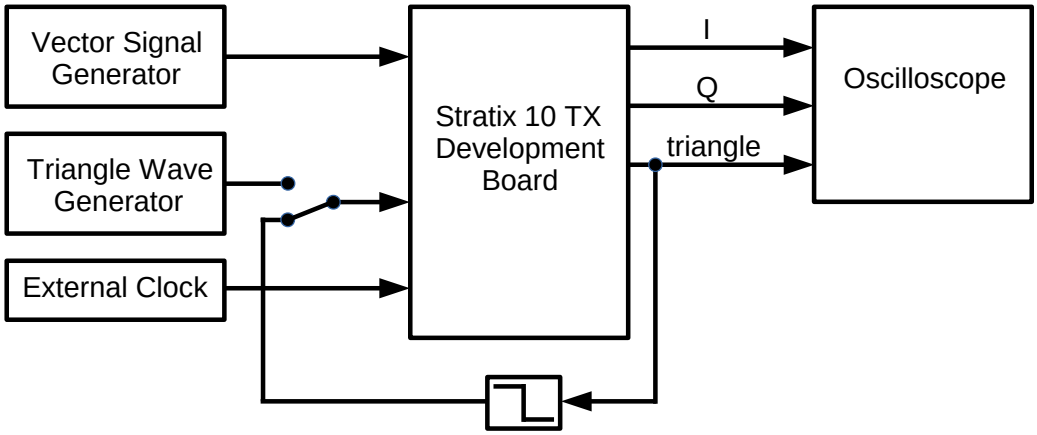


Figure 3.14: *Test environment.*

4 Results

The results are split into three parts. In the first part MATLAB was used to simulate the receiver indifferent ways to find the theoretical performance and find situations where the receiver performed bad. In the second part ModelSim was used to simulate the VHDL implemented receiver to see that the outcome was the same as in MATLAB. In the last part the receiver was tested on the FPGA.

4.1 MATLAB simulations

If nothing else is mentioned in the following simulations, the parameters in Table 4.1 were used for the simulations.

Modulation	64 QAM
Symbol rate	15 Msymbols/s
Number of symbols	100000
Bandwidth	20 MHz
Sampling frequency	9.6 GHz
Carrier frequency	2.4 GHz
Reference frequency	17 MHz

Table 4.1: *Standard parameters for MATLAB simulation.*

The performance was measured in EVM_{rms} and BER. Bit errors usually started to occur when the EVM_{rms} reached around 5-6%.

4.1.1 Simulation, 64 QAM, 20 MHz bandwidth

The first simulation of the receiver was done to test the functionality with the standard parameters to see the behavior all the way from original baseband signal to the resulting estimated symbols used to measure EVM_{rms} and BER. The spectrum of the original baseband can be seen in Figure 4.1. The QAM signal is a complex signal that is asymmetric around zero.

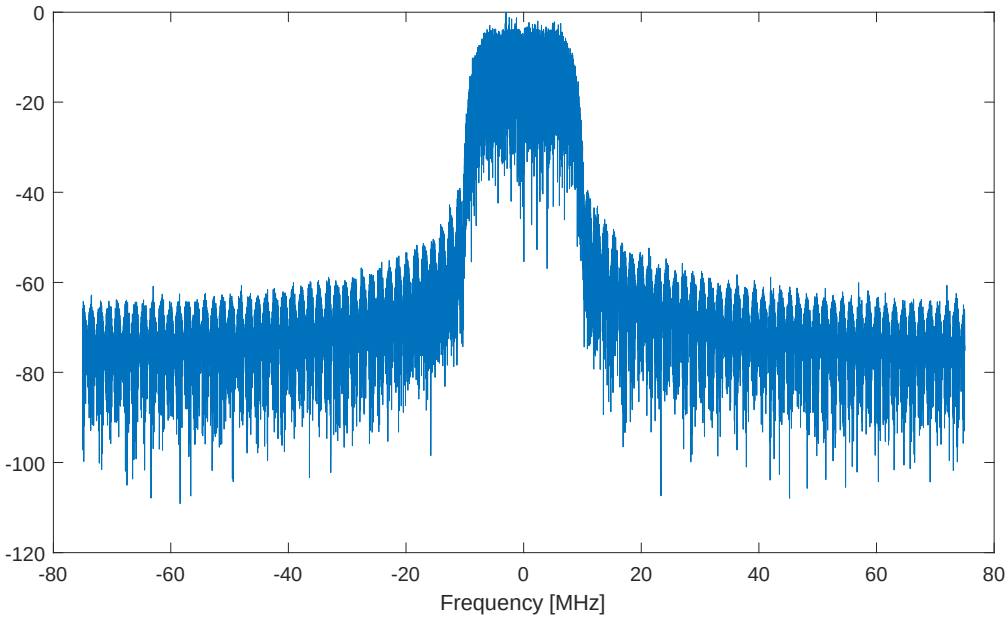


Figure 4.1: *Spectrum of original baseband signal.*

When upconverted to RF, the complex baseband became a real signal centered around the carrier frequency, see Figure 4.2. This is the signal the vector signal generator will send to the transceiver.

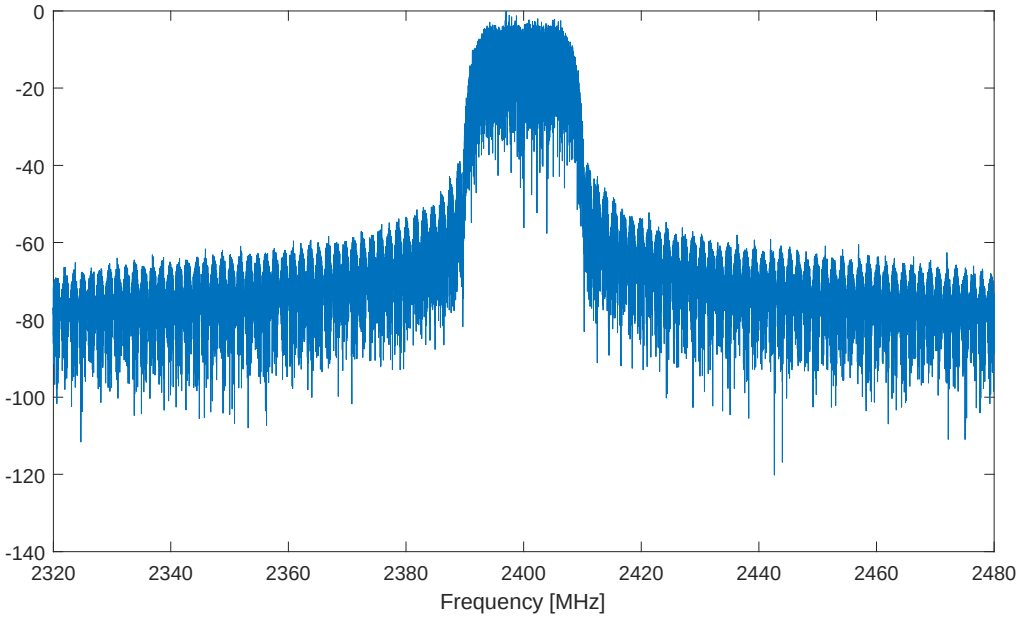


Figure 4.2*: Spectrum of transmitted RF signal.*

When sampled by the transceiver, the sampling process caused harmonics to appear around the RF signal, see Figure 4.3. The closest harmonics appeared at f_c +/- $2f_r$ as expected with a triangle wave as reference.

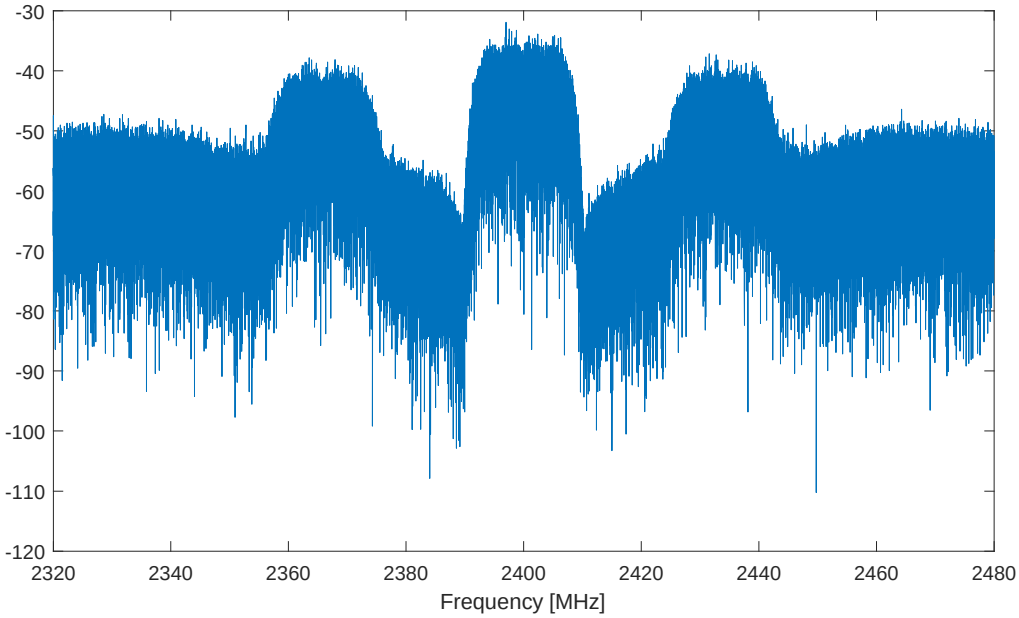


Figure 4.3: *Spectrum of sampled passband signal.*

After the downconversion the wanted signal appeared around zero frequency, see Figure 4.4. Since the filter was designed to keep everything between -50 MHz and 50 MHz the harmonics were still present.

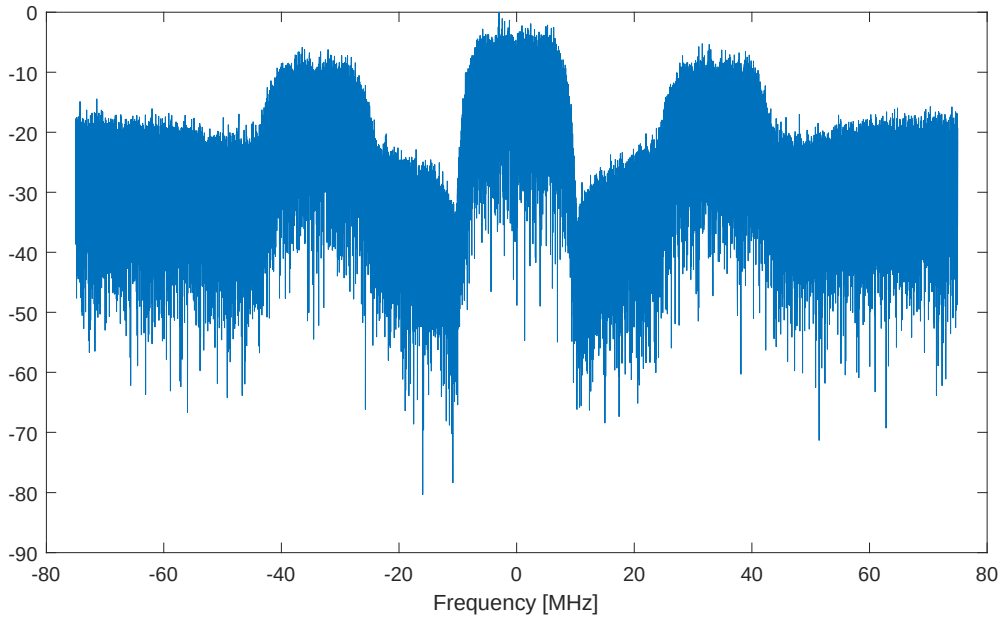


Figure 4.4: *Spectrum of downconverted baseband signal.*

After the matching filter the harmonics and the high frequency noise were attenuated and the spectrum of the baseband was restored to minimize the ISI, see Figure 4.5.

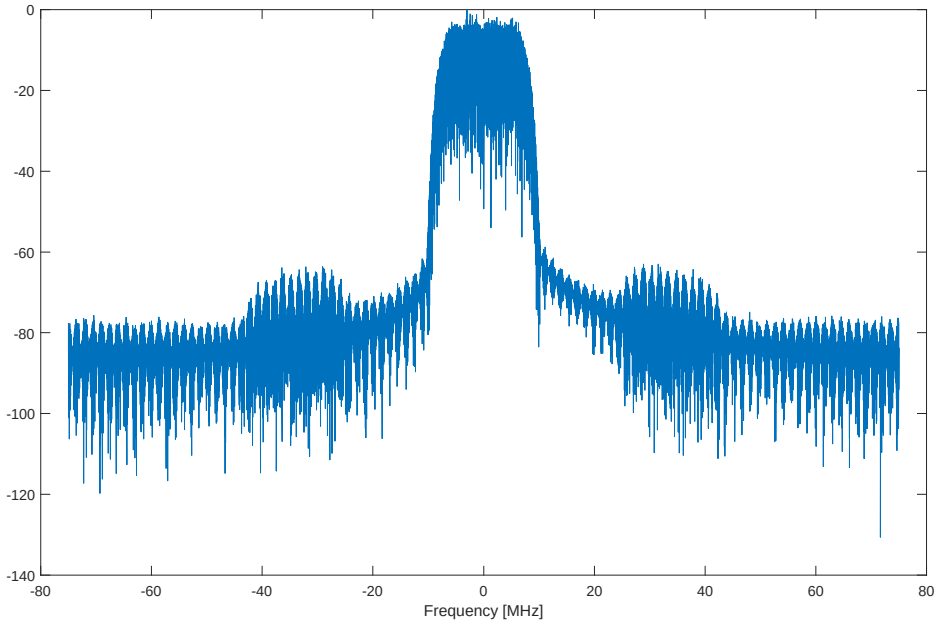


Figure 4.5: *Baseband signal after matching filter.*

After downsampling the symbols were scaled to match the constellation diagram used to map the bits. This was done multiplying the symbols with the expected average power divided by the average power of the received signal. Since the transmitted symbols are random, every possible symbol have equal probability and in a long enough signal they should appear equally often. By calculating the sum of the powers for all 64 symbols and divide by 64, the expected average power is calculated to be $\sqrt{42}$. The symbols can then be scaled with $\sqrt{42}/P_{\text{received, avg}}$. The resulting symbols are shown in Figure 4.6.

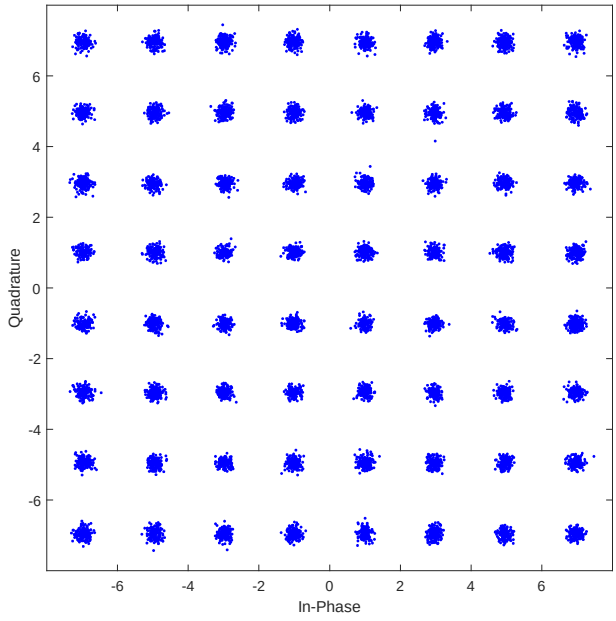


Figure 4.6: *Scatterplot of received symbols.*

4.1.2 Performance, different bandwidths

Here the MATLAB model was used to simulate different bandwidths by changing the symbol rate. Different bandwidths required different frequencies of the triangle wave for the receiver to work best. The simulations were also done with ordinary PCM sampling to see how much distortion the PWM-based sampling added. The simulations were done with 100000 symbols, except for when the bandwidth was 5 MHz where 50000 symbols were used instead because the memory required for the simulation was too large for 100000 symbols. The results are shown in Table 4.2.

Symbol rate (Msymbols/s)	3.75	7.5	15	30	75
Bandwidth (MHz)	5	10	20	40	100
Triangle wave frequency(MHz)	6	11	17	35	79
PWM EVM_{rms} (%)	1.50	1.56	1.86	2.58	5.04
Bit errors	0	0	0	0	1
Reference EVM_{rms} (%)	1.25	1.21	1.20	1.25	0.98

Table 4.2: *Performance for different bandwidths.*

The results showed that signals with higher bandwidth had worse EVM, especially for bandwidths higher than 40 MHz. No bit errors could be detected except for the highest bandwidth where one bit error was detected among the received 600,000 bits.

4.1.3 The effect of different sampling and carrier frequencies.

The VHDL implementation had a sampling frequency of 9.6 GHz and a carrier frequency of 2.4 GHz. This was chosen because the resulting oscillators would only require four numbers with the repeating values 0,-1,0,1,0,-1,0,1..., which made the implementation easier. Here the receiver was simulated with carrier frequencies between 2.3 and 2.5 GHz, sampling frequencies between 9.6 and 28.8 GHz and with 10000 symbols. If the frequency of the triangle wave stayed the same at 17 MHz for all carrier frequency, the resulting EVM_{rms} and BER would be as in Figure 4.7a) and 4.7b).

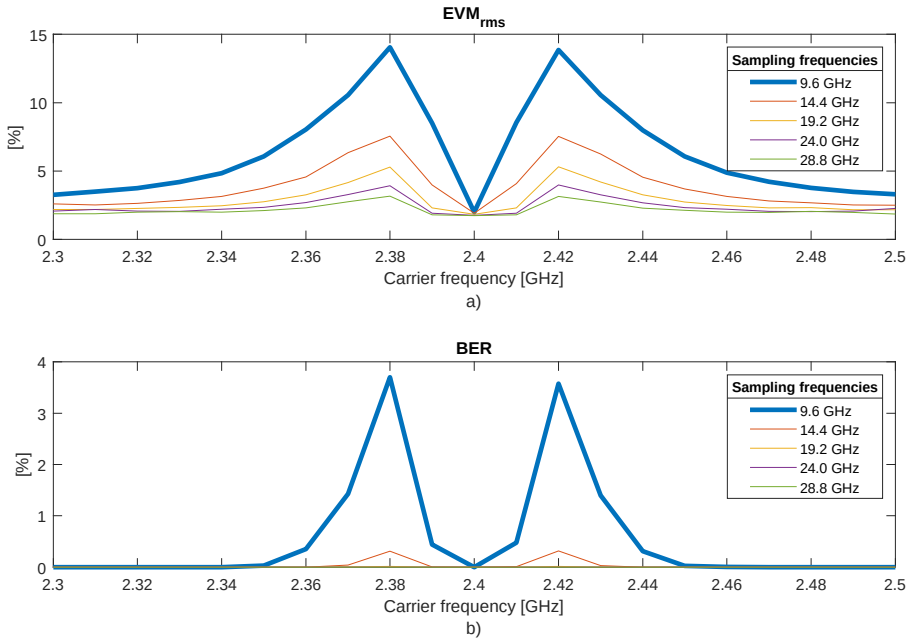


Figure 4.7: *(a) EVM_{rms} and (b) BER for different sampling and carrier frequencies with fixed triangle wave frequency.*

At a sampling frequency of 9.6 GHz the performance was only really good when the carrier frequency was 2.4 GHz. When moving away from 2.4 GHz, the EVM_{rms} peaked at 2.38 GHz and 2.42 GHz, which resulted in a BER of more than 3%. This was because of the harmonics caused by the PWM-based sampling got more spread out around the carrier frequency due to aliasing after the sampling. Higher sampling frequencies increased the performance as the aliasing folded down less harmonics. Another way to increase the performance was to have different frequencies of the triangle wave for each carrier frequency, see Figure 4.8c). By doing this, the resulting EVM_{rms} and BER would instead be as in Figure 4.8a) and 4.8b).

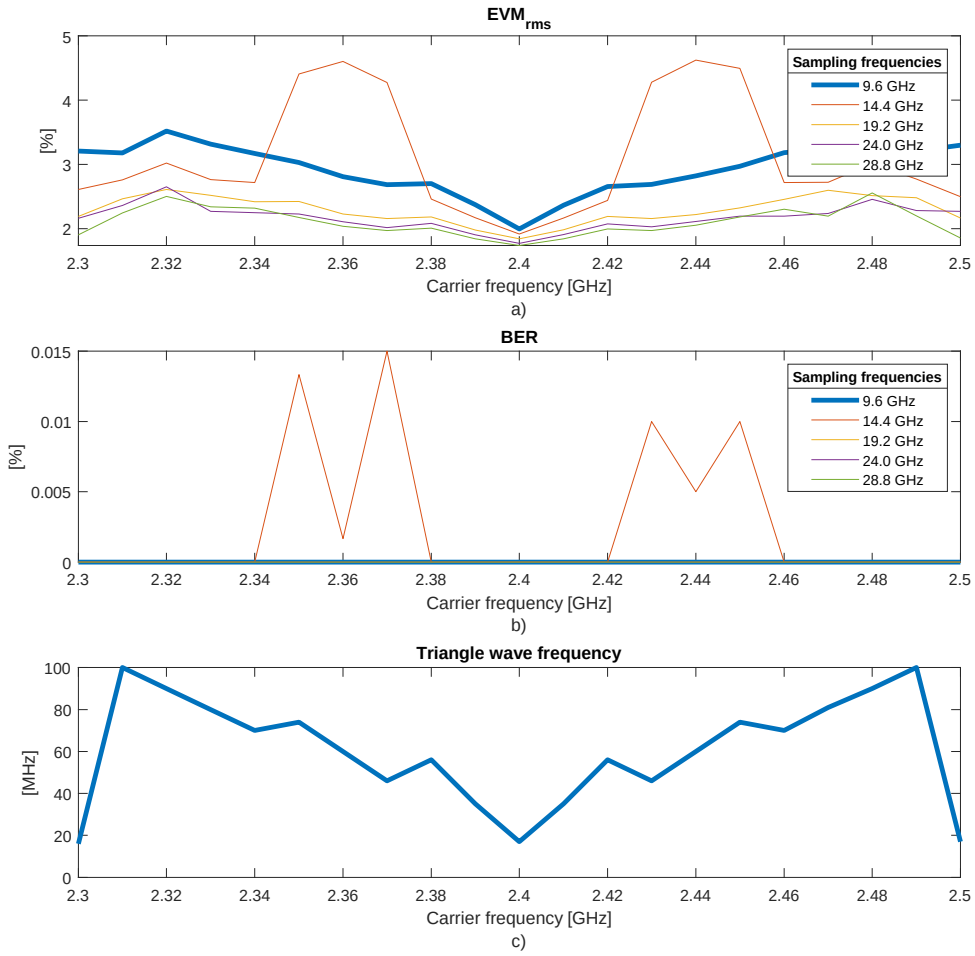


Figure 4.8: *(a) EVMrms, (b) BER for different sampling and carrier frequencies and (c) the frequency of the triangle wave.*

With variable frequency of the triangle wave a sampling frequency of 9.6 GHz worked for all carrier frequencies with no bit errors detected. Higher sampling frequencies improved the performance furthermore, except for 14.4 GHz, which did not work well for some carriers.

4.1.4 The effect of different passband gains.

The differential inputs to the transceiver had the reference wave as one of the inputs and the RF signal as the other input. The amplitude of the RF signal is expected to never be higher than the amplitude of the triangle wave but as close as possible, this would be controlled by a VGA. For example, if there are other signals together with the RF signal, see Figure 4.9, the VGA would have

to compensate and the RF signal would get weaker. The VGA may also set its gain to high so that it outputs an amplitude that is higher than the triangle wave.

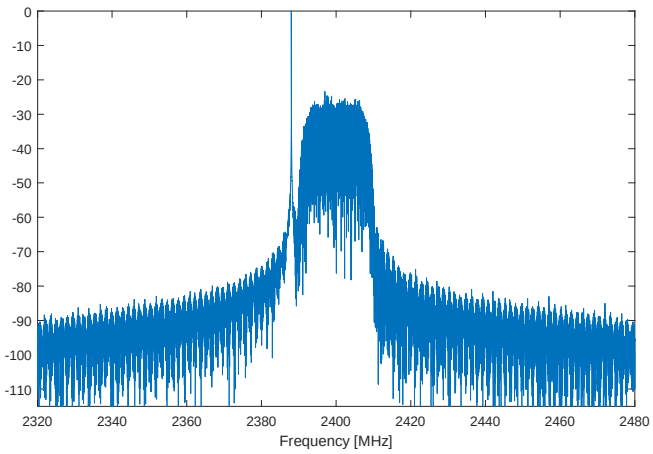


Figure 4.9: *RF signal with interferer.*

Different ratios between the maximum amplitude of the RF signal and the triangle wave were simulated and EVM_{rms} and BER were calculated. The results can be seen in Figure 4.10.

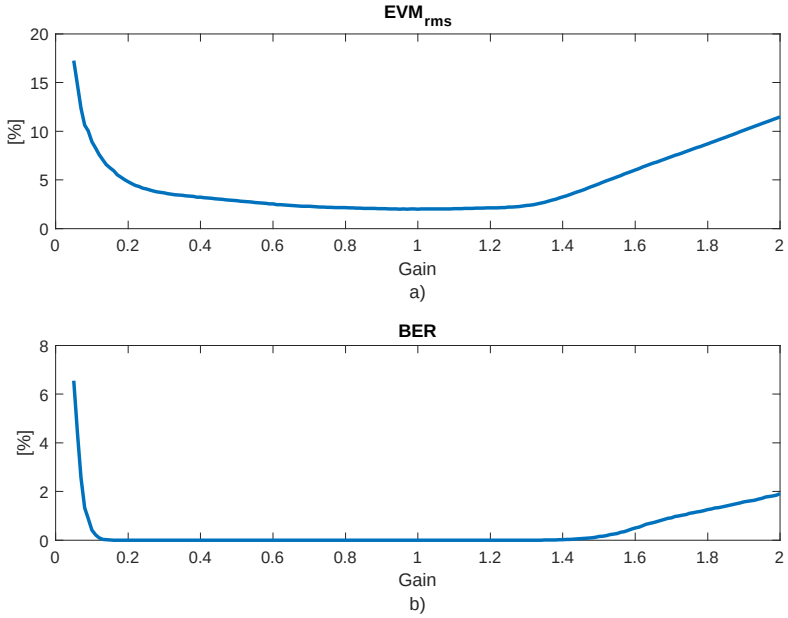


Figure 4.10: *(a) EVM_{rms} and (b) BER for different gains of the RF signal.*

The receiver worked best when the maximum amplitude of the RF signal was close to the maximum amplitude of the triangle wave. Between gains of 0.2 and 1.3, or -14 and 2.3 dB, no bit errors could be detected. Below 0.2 the amount of bit errors increased rapidly, which suggested that as much as possible around the RF signal should be filtered away before the VGA.

4.1.5 No ideal BP filter

With the vector signal generator there would be no others signals but when receiving a wireless signal there could be other signals on other frequencies present. An analog bandpass filter was therefore used since the PWM based sampling require a band-limited signal. This bandpass filter would have high requirements if only the test signal with a bandwidth of 20 MHz should pass through unaffected. More realistically the filter would have a higher bandwidth. When other signals are present the power of the wanted would be lower, as seen in Figure 4.9, compared to when the wanted signal is alone.

Here the receiver was simulated with five 64-QAM channels with 20 MHz bandwidth each where a bandpass filter let the three channels centered around 2.4 GHz pass through untouched, see Figure 4.11. The other channels were in the transition bands and everything else were attenuated at least 40 dB. The total bandwidth that had to be sampled was therefore 100 MHz instead of, as before, 20 MHz.

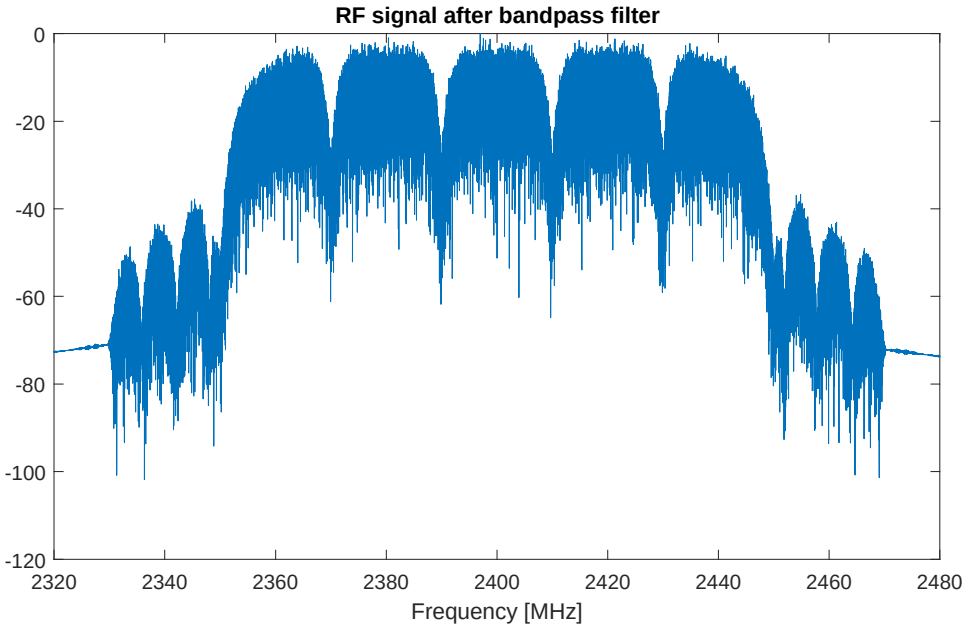


Figure 4.11: *RF signal with no ideal bandpass filter.*

Since the bandwidth was higher, the frequency of the triangle wave had to increase from 17 MHz to 61 MHz. The spectrum of the downconverted signal is shown in Figure 4.12. By running a test with 100000 symbols the EVM_{rms} was measured to be 7.31% and the number of bit errors was 425 which gave a BER of 0.071%. The scatter plot in Figure 4.13 shows the received symbols and it can be seen that they spread out much more now. As a comparison with an ideal bandpass filter, the EVM_{rms} was 1.86%. A narrow bandpass filter is therefore really important for best performance.

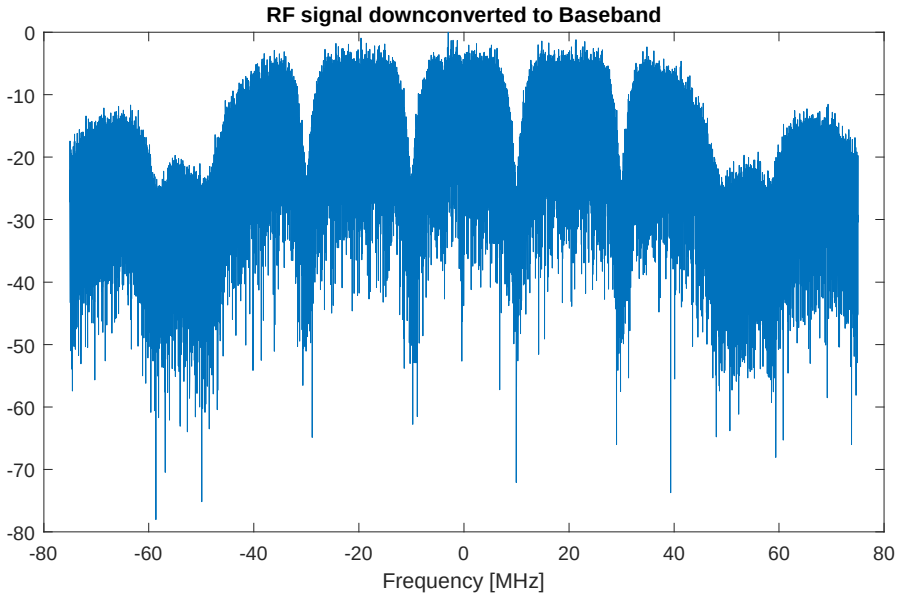


Figure 4.12: *The baseband after downconversion.*

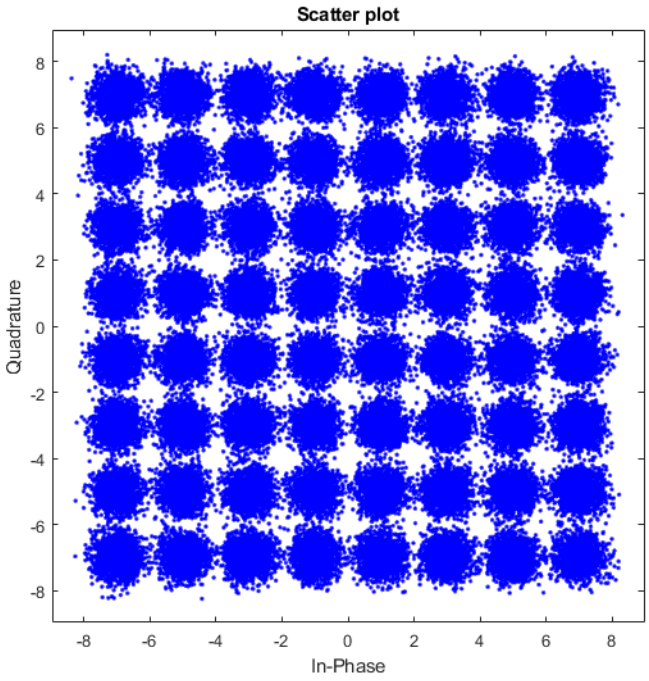


Figure 4.13: *Scatterplot of received symbols.*

With the RF signal seen in Figure 4.11, there were two more channels at 2.38 GHz and 2.42 GHz that passed through the bandpass filter and could be received. Channels could be selected by changing the frequency of the sine and cosine waves in the downconversion. The receiver could then use the same analog bandpass filter to receive multiple channels. Here the receiver was simulated in the same way as in section earlier but the channel at 2.42 was received instead. The spectrum after the downconversion is then instead as in Figure 4.14 where the channels had moved down 20 MHz.

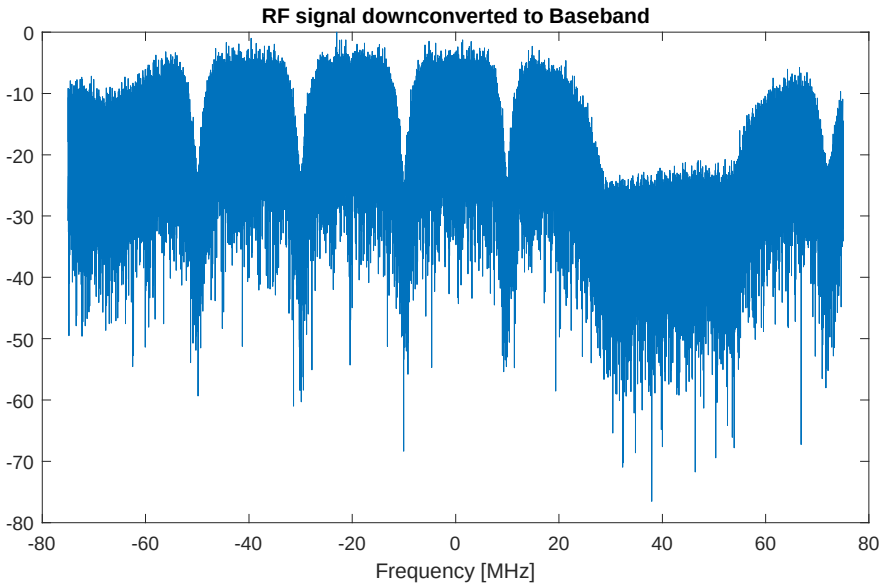


Figure 4.14: *The baseband after downconversion.*

The received symbols are shown in Figure 4.15. The EVM_{rms} was 7.54% and 611 bit errors gave a BER of 0.10%. Selecting the 2.38 GHz channel instead gave an EVM_{rms} of 7.84% and 908 bit errors gave a BER of 0.15%. The performance was not that much worse than when the carrier was 2.40 GHz so if a bandpass filter with a bandwidth much higher than the bandwidth of the RF signal had to be used anyway, the receiver might as well be built with multiple channels in mind.

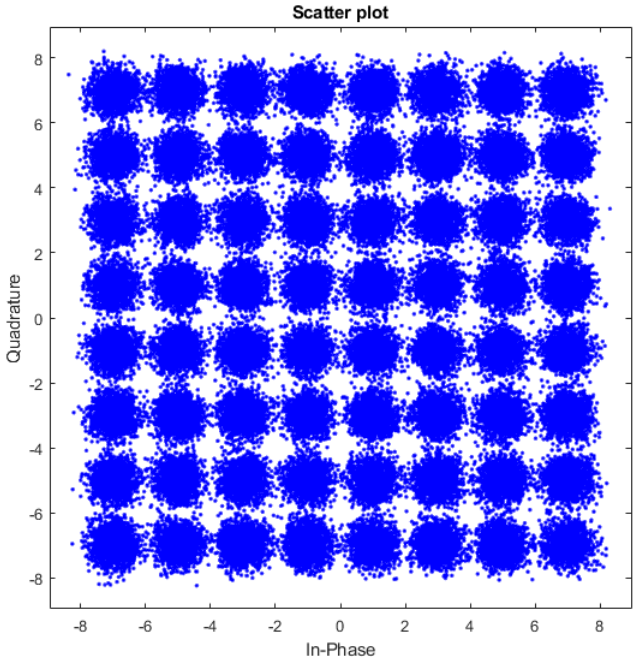


Figure 4.15: *Scatterplot of received symbols.*

4.2 Modelsim results

To test the VHDL implementation of the receiver in ModelSim, a 64-QAM test signal was created in MATLAB. This signal contained all possible constellation points in random order, see Table 4.3. The receiver was first simulated in MATLAB with the same filter coefficients and truncations that was later used in the VHDL implementation to get a reference for comparison with the results from the ModelSim simulation.

n	symbol	x[n]	n	symbol	x[n]	n	symbol	x[n]	n	symbol	x[n]
0	31	-3 - 3i	16	32	7 + 7i	32	47	5 - 3i	48	43	5 + 3i
1	39	7 - 3i	17	6	7 - 1i	33	52	1 - 7i	49	14	-5 - 1i
2	21	-1 - 5i	18	37	7 - 5i	34	28	-3 - 7i	50	0	-7 + 7i
3	33	7 + 5i	19	57	3 + 5i	35	20	-1 - 7i	51	35	7 + 3i
4	34	7 + 1i	20	41	5 + 5i	36	24	-3 + 7i	52	22	-1 - 1i
5	5	-7 – 5i	21	27	-3 + 3i	37	51	1 + 3i	53	1	-7 + 5i
6	54	1 -1i	22	16	-1 + 7i	38	36	7 - 7i	54	3	-7 + 3i
7	2	-7 + 1i	23	40	5 + 7i	39	63	3 - 3i	55	17	-1 + 5i
8	15	-5 - 3i	24	46	5 - 1i	40	30	-3 - 1i	56	23	-1 - 3i
9	10	-5 + 1i	25	13	-5 - 5i	41	48	1 + 7i	57	38	7 - 1i
10	53	1 - 5i	26	45	5 - 5i	42	26	-3 + 1i	58	12	-5 - 7i
11	29	-3 - 5i	27	55	1 - 3i	43	60	3 - 7i	59	8	-5 + 7i
12	44	5 - 7i	28	62	3 - 1i	44	49	1 + 5i	60	19	-1 + 3i
13	59	3 + 3i	29	7	-7 - 3i	45	25	3 - 7i	61	56	3 + 7i
14	61	3 - 5i	30	58	3 + 1i	46	42	1 + 5i	62	9	-5 + 5i
15	50	1 + 1i	31	4	-7 - 7i	47	18	-3 + 5i	63	11	-5 + 3i

Table 4.3: *The test signal, x[n], used to verify the receiver implemented in VHDL.*

The resulting constellation diagram from the MATLAB simulation is seen in Figure 4.16 and it shows that the points appear where they are expected to be.

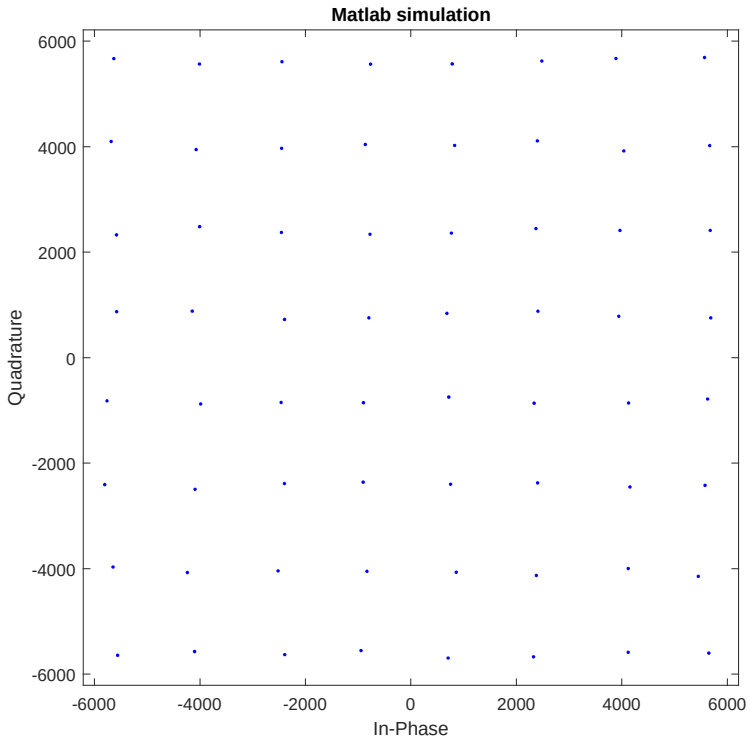


Figure 4.16: *MATLAB simulation.*

The VHDL implementation consisted of three parts. First the polyphase DDC filter, which downconverted the real passband signal down to a complex baseband signal and downsampled it by a factor of 64, see the red and green plots in Figure 4.17. After that the matching filters were applied to remove high frequency noise and remove the ISI caused by the PAM filter, see blue and yellow plots in Figure 4.17. The complex baseband signal was then downsampled by a factor of 10 to get y[n], which is the received version of x[n], see magenta and cyan plots in Figure 4.17.

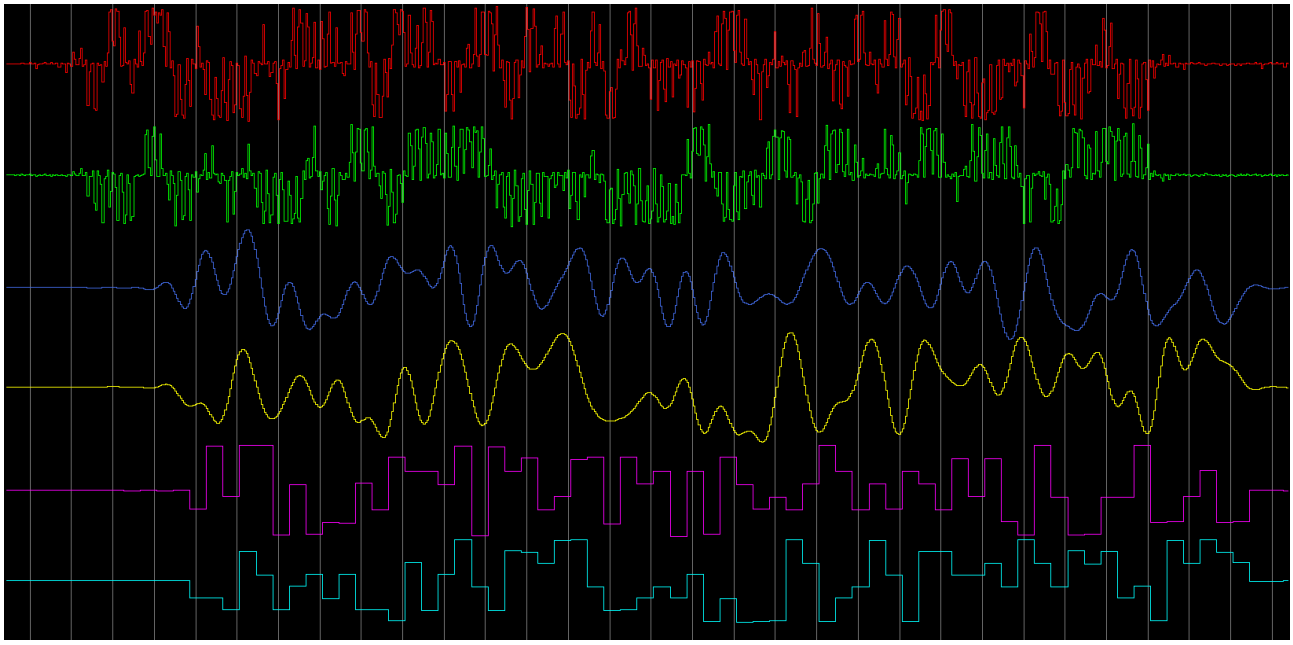


Figure 4.17: *Red: I after DDC, green: Q after DDC, blue: I after matching filter, yellow: Q after matching filter, magenta: baseband real part, cyan: baseband imaginary part.*

The constellation diagram of y[n] from the ModelSim simulation is shown in Figure 4.18 and the points appear at the same places as in the MATLAB simulation.

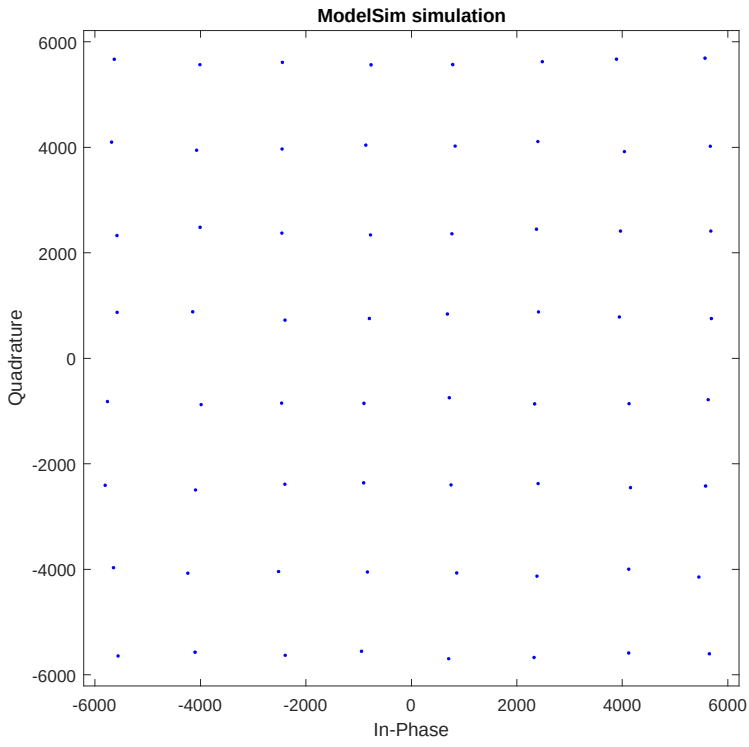


Figure 4.18: *ModelSim simulation.*

A closer look at the results are presented in Table 4.4, where the exact results from both simulations are shown together with the estimated symbols. It can be seen that the results from MATLAB and ModelSim are exactly the same, which verified that the VHDL implementation worked.

n	y[n]: MATLAB	y[n]: ModelSim	Estimated symbol	n	y[n]: MATLAB	y[n]: ModelSim	Estimated symbol	n	y[n]: MATLAB	y[n]: ModelSim	Estimated symbol
0	-2399 - 2388i	-2399 - 2388i	31	22	-765 + 5565i	-765 + 5565i	16	44	830 + 4024i	830 + 4024i	49
1	5579 - 2421i	5579 - 2421i	39	23	3889 + 5672i	3889 + 5672i	40	45	-2452 + 3969i	-2452 + 3969i	25
2	-833 - 4052i	-833 - 4052i	21	24	4128 - 860i	4128 - 860i	46	46	3943 + 784i	3943 + 784i	42
3	5668 + 4020i	5668 + 4020i	33	25	-4240 - 4074i	-4240 - 4074i	13	47	-797 + 755i	-797 + 755i	18
4	5688 + 754i	5688 + 754i	34	26	4122 - 3998i	4122 - 3998i	45	48	3965 + 2412i	3965 + 2412i	43
5	-5649 - 3970i	-5649 - 3970i	5	27	752 - 2400i	752 - 2400i	55	49	-3988 - 877i	-3988 - 877i	14
6	719 - 748i	719 - 748i	54	28	2337 - 864i	2337 - 864i	62	50	-5634 + 5670i	-5634 + 5670i	0
7	-5581 + 872i	-5581 + 872i	2	29	-5807 - 2408i	-5807 - 2408i	7	51	5678 + 2412i	5678 + 2412i	35
8	-4094 - 2496i	-4094 - 2496i	15	30	2410 + 880i	2410 + 880i	58	52	-900 - 854i	-900 - 854i	22
9	-4147 + 882i	-4147 + 882i	10	31	-5562 - 5644i	-5562 - 5644i	4	53	-5685 + 4099i	-5685 + 4099i	1
10	860 - 4069i	860 - 4069i	53	32	4155 - 2452i	4155 - 2452i	47	54	-5583 + 2328i	-5583 + 2328i	3
11	-2519 - 4043i	-2519 - 4043i	29	33	706 - 5696i	706 - 5696i	52	55	-864 + 4042i	-864 + 4042i	17
12	4121 - 5587i	4121 - 5587i	44	34	-2393 - 5631i	-2393 - 5631i	28	56	-905 - 2361i	-905 - 2361i	23
13	2372 + 2447i	2372 + 2447i	59	35	-946 - 5555i	-946 - 5555i	20	57	5628 - 785i	5628 - 785i	38
14	2380 - 4130i	2380 - 4130i	61	36	-2446 + 5612i	-2446 + 5612i	24	58	-4102 - 5572i	-4102 - 5572i	12
15	683 + 839i	683 + 839i	50	37	768 + 2361i	768 + 2361i	51	59	-4010 + 5567i	-4010 + 5567i	8
16	5568 + 5691i	5568 + 5691i	32	38	5651 - 5602i	5651 - 5602i	36	60	-776 + 2340i	-776 + 2340i	19
17	-5764 - 820i	-5764 - 820i	6	39	2404 - 2375i	2404 - 2375i	63	61	2483 + 5624i	2483 + 5624i	56
18	5451 - 4148i	5451 - 4148i	37	40	-2461 - 850i	-2461 - 850i	30	62	-4074 + 3945i	-4074 + 3945i	9
19	2399 + 4110i	2399 + 4110i	57	41	784 + 5569i	784 + 5569i	48	63	-4007 + 2484i	-4007 + 2484i	11
20	4039 + 3919i	4039 + 3919i	41	42	-2396 + 724i	-2396 + 724i	26				
21	-2456 + 2375i	-2456 + 2375i	27	43	2326 - 5672i	2326 - 5672i	60				

Table 4.4: *The received signal, y[n], from MATLAB and ModelSim simulations.*

4.3 FPGA results

The FPGA board started to malfunction during this thesis and became more and more difficult to program. It was discovered that it was possible to program just after power up but after a time the FPGA board broke down completely and did not react to any input at all. The most likely reason was problem with the power circuits on the board. The board was never repaired or replaced so therefore many planned measurements could not be done. The results in form of pictures from oscilloscopes and spectrum analyzers are therefore missing as they were not done before the FPGA board stopped working. The results in this section are only in written form from observations done when the receiver was in a more or less early stage of development.

4.3.1 E-tile transceiver

According to the theory study, most likely only the H-tile transceivers could be used to sample an RF signal by comparing it with a reference signal. The H-tile transceivers were not physically connected to any connector on the board so these could never be tested. E-tile transceivers on the other hand were connected but not expected to work, despite that were several tests still performed to see how the transceiver would react.

The transceiver was tested by connecting its differential inputs to two signal generators, one that generated a triangle wave and one that generated a sine wave. This should generate a PWM version of the sine wave. The resulting sampled bits was then transmitted by the transceiver to an oscilloscope where a sine would have appeared on the screen after lowpass filtering. This test failed because of CDR, which tried lock on to the data clock which was never present in the signals. What appeared on the screen was just noise.

Another test was done with a square wave whose frequency was the transceiver frequency divided by an even number. This would give the pattern 1,0,1,0..., 1,1,0,0,1,1,0,0..., and so on. This test succeeded since the same pattern was transmitted correctly. This proved that the transceiver worked with digital signals but could not accept two analog signals and sample those at any time.

4.3.2 Triangle wave output

The triangle wave output was tested by connecting the transceiver output to an oscilloscope with a lowpass filter. The transceiver transmitted the bits containing the PWM triangle and the triangle wave appeared on the oscilloscope. Creating a triangle wave used this method worked as intended.

4.3.3 FPGA – Receiver

The whole receiver was tested but instead of using a transceiver as input a PWM version of the RF signal was done in MATLAB and inserted into a VHDL file. Two transceivers were used to output the digital complex baseband to an oscilloscope, which displayed the signals by using the X-Y mode. When running the FPGA, the oscilloscope displayed the correct constellation points, which proved that the VHDL implementation also worked on the FPGA.

5 Discussion

5.1 Results discussion

5.1.1 MATLAB simulations

The MATLAB simulations confirmed that the FPGA implemented receiver should be able to receive a 64-QAM signal with 20 MHz bandwidth. This was with the assumption that the comparators in the transceivers were close to ideal and did not add too much noise. Increasing the symbol rate and there by the bandwidth increased the EVM_{rms} and when the bandwidth was 100 MHz bit errors started the occur. The PWM-based sampling used for this receiver worked best with RF signals that had a bandwidth within a few tens of MHz. Using 20 MHz was therefore a good choice for this receiver.

When the sampling frequency was 9.6 GHz, using a carrier frequency of 2.4 GHz gave the best performance. When other carrier frequencies were used, the frequency of the reference wave had to be increased to avoid that the harmonics folded down into the signal to be received due to aliasing. The receiver would have to be able to change the frequency of the reference wave if it should be able to receive signals with other carrier frequencies. By changing the reference wave, signals with carrier frequencies other than 2.4 GHz could be received without bit errors but the EVMrms for these went up the further they were from 2.4 GHz. Higher sampling frequencies improved the performance for all carriers so going up from 9.6 GHz and use the maximum speed of the Stratix 10 TX transceivers could be considered. Another way to improve the performance for carriers other than 2.4 GHz could be to always sample at at frequency that is four times higher than the carrier. The drawback with this is that with 9.6 GHz sampling frequency, each symbol occupies exactly 10 samples after the DDC filters. With a carrier of 2.42 GHz for example, each symbol would occupy 10.0833 samples instead if the symbol rate is constant. This would require resampling that would use extra processing power.

In previous simulations it was always assumed that the just the wanted 64-QAM RF signal reached the receiver and everything else was removed by the analog bandpass filter. With the vector signal generator this would be the case but when receiving a wireless signal there would be others more or less strong signals. Sharp analog bandpass filters with a bandwidth of 20 MHz centered around 2.4 GHz are hard to make so filters with higher bandwidth would be expected. The simulations with 5 adjacent channels with a total bandwidth of 100 MHz showed that this had great impact on the performance. The BER went from 0 to 0.07% and with noise included this would be even worse. 16-QAM could be the alternative depending on how good bandpass filter is used. A bandpass filter with high bandwidth could still be useful if you want to design a receiver that could switch between multiple channels without changing the bandpass filter. By just changing the VHDL code, signals at different carrier frequencies could be downconverted to baseband since they are all sampled by the transceiver. Simulations showed that channels located 20 MHz below and above 2.4 GHz just had

slightly worse EVM_{rms} than the channel located at 2.4 GHz so by using 16-QAM instead a multi-channel receiver could be built.

5.1.2 Quartus and Modelsim

The ModelSim simulation showed that the implementation worked and gave the same results as the MATLAB simulation with fixed-point numbers. Changes in the VHDL implementation could still be required depending on FPGA and clock speed since the timing requirements may not be met. Especially care has to be taken with the DDC filters where 641 16-bits numbers have to be added together. In the case when the carrier frequency is 2.4 GHz and the sampling frequency is 9.6 GHz, the DDC filters would not have needed so many coefficients since 320 of the them were zeros anyway and could be removed. The input bus could then also be reduced to 32 bits since every other bit would go the DDC filter for the I component and the rest to the DDC filter for the Q component. If other carriers would have been used, the DDC filters would to have been changed to lowpass filters and the down-mixing would have happen before by implementing 2*64 parallel mixers with sine and cosine generators.

When the truncations after the filters were done it was considered that the inputs to the filters could make the resulting outputs from the filters to be the highest possible numbers they could output. This was done to be absolutely sure that overflow would never occur. It is possible that the filter outputs would never be close to these theoretical maximum possible numbers. The truncations could therefore have been unnecessary hard and added unnecessary quantization noise.

5.1.3 FPGA

The main problem with implementing the receiver on the Stratix 10 TX development kit was that the CDR for the transceivers had to be configured to not adjust the phase and frequency based on the recovered clock from the incoming data, otherwise the PWM-based sampling would not work. Only with H-tile transceivers the CDR could configured that way and these where not connected on the board [12] [13]. The available model of the Stratix 10 TX only had E-tile transceivers, where the CDR could not be configured that way, connected on the board which made it impossible for the FPGA implemented receiver to work [11] [13]. In [6] the FPGA based transceiver was implemented in a KC705 board from Xilinx with a XC7K325T FPGA where the CDR can be locked to reference clock and never adjust the phase and frequency [14]. As a transmitter to make the triangle wave and output the downconverted baseband, E-tile transceiver worked well.

If the receiver could have been tested on the Stratix 10 TX by using H-tile transceivers and received an RF signal from the vector signal generator, the resulting complex baseband signal may not have looked as good as the ModelSim simulations. Phase and frequency offsets would cause synchronization errors, which could make it hard to see the symbols on an oscilloscope using the X-Y mode. Instead of using 64-QAM just some simple sine waves could have been used instead to just verify the functionality. Sending data to a PC using the Ethernet port and do the analysis there would have been the preferable choice if there were more time.

The Stratix 10 TX development board got hardware issues early and broke down. This made it impossible to do later measurements and tests or find ways to work around the transceiver problem. Maybe an external circuit with a comparator could have been used just to show that the concept worked, though with lower sampling and carrier frequency. The tests that were done before the breakdown showed that the receiver worked as in the ModelSim simulations.

5.2 Method discussion

At the beginning of this thesis it was assumed that the Intel Stratix 10 TX could be used as a receiver since it has digital transceivers and it was proven, according to [6], that digital transceivers could be used to sample analog signals. A study about the Xilinx Kintex-7 FPGA and its transceiver used in [6] should have been done to understand how and why that transceiver could be used. That would have helped to understand early if the Stratix 10 TX could be used or not. The Stratix 10 TX has two types of transceivers and the ones of type H-tile was most likely to work according to the user guides for the Stratix 10 TX and Kintex-7 FPGAs [12] [14]. Just because the Stratix 10 TX has H-tile transceivers, they are not necessary connected on the board which was also found too late since it was hard to find out what board the Stratix 10 TX was mounted on and the schematics for this board. Support from Intel or someone who had experience with the Stratix 10 TX would have helped a lot since the user guides for the transceivers consisted of hundreds of pages and it was hard to understand what the transceivers actually could do and how they worked [11] [12].

A lot of work was therefore done to design a receiver that never could be fully implemented on the available hardware which was the goal with this thesis. The simulations showed that the receiver in theory would have worked but this assumed an ideal comparator in the transceiver. If the transceivers would have added to much noise the simulations would have been invalid and the specifications would have had to change. According to [6], an FPGA with transceivers could also be used to build a transmitter. This was partly implemented to create the triangle wave which was tested on the FPGA and worked. If the conclusion that the available Stratix 10 TX could not have been used as a receiver had come early, it may have been a better choice to implement a transmitter. Except QAM signals, other type of signals could have been generated that easily could have been tested, like FM or AM radio.

5.3 The work in a wider perspective

When ordinary communication devices, such as base stations, need to be updated, only the software can be updated without changing the hardware. If the update could not be done in software the only choice is to change hardware components which require more energy and create more waste. By using FPGAs, new hardware features can be added by just reconfiguring the FPGA. As long as the FPGA can handle the features the same hardware can be used for many updates and thus reduce the need to manufacture more hardware.

6 Conclusions

A digital receiver where analog components are replaced by digital counterparts is possible to build but has its disadvantages. The easiest method to build a 1-bit ADC was the PWM-based method since it could be built by only using a transceiver and an analog filter. The disadvantages with PWM-based ADCs are that they add a lot of noise and that the bandwidth is limited. An analog bandpass filter is needed before the ADC and for best performance this filter has to be as narrow as possible, which is hard to build with high center frequencies. If the receiver should be able to handle many carrier frequencies several bandpass filters would be needed. A 64-QAM signal with a bandwidth of 20 MHz should work if the bandpass filter is narrow enough but going down to 16-QAM would work better since it less sensitive to noise and would put less demands on the bandpass filter.

The simplest way to do the VHDL implementation was to have a sampling frequency of 9.6 GHz and a carrier frequency of 2.4 GHz and combine the I and Q mixers with the lowpass filter and create polyphase bandpass filters that did the downconversion and downsampling at the same time. By doing this, no multipliers were needed so the filters could be implemented by just using adders which saved resources on the FPGA. The matching filters on the other hand needed multipliers but much less than the polyphase filters would have needed. The resulting complex digital baseband could be stored with 32 bits by using two 16 bits fixed-point numbers which have a SNR that is high enough and would fit well in the memory on a PC.

This thesis could not conclude whether the Stratix 10 TX FPGA could be used as an all-digital receiver or not. Only E-tile transceivers were connected on the board which could not be configured as intended. H-tile transceivers would more likely have worked but since these were not connected the available board could not be used. When building FPGA based receivers using transceivers as ADCs, it is important how the transceivers can be configured. The transceivers expect digital signals, so when analog signals are used the CDR has to be turned off to allow the sampling moments to happen at any time no matter what the input signals are. To continue the work with the receiver a different version of the Stratix 10 TX development kit would be required, or a different FPGA with transceivers that fulfill the requirements.

7 Bibliography

[1] J. Mitola, "The software radio architecture," in *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26-38, May 1995, doi: 10.1109/35.393001.

[2] Ashry, A. and Aboushady, H. (2013) “A 4th Order 3.6 GS/s RF ΣΔ ADC With a FoM of 1 pJ/bit", *IEEE Transactions on Circuits and Systems I: Regular Papers, Circuits and Systems I: Regular Papers, IEEE Transactions on, IEEE Trans. Circuits Syst. I*, 60(10), pp. 2606–2617. doi: 10.1109/TCSI.2013.2248832.

[3] L. Hernandez and E. Gutierrez, "Analytical Evaluation of VCO-ADC Quantization Noise Spectrum Using Pulse Frequency Modulation," in *IEEE Signal Processing Letters*, vol. 22, no. 2, pp. 249-253, Feb. 2015, doi: 10.1109/LSP.2014.2357071.

[4] H. C. Hor and L. Siek, "Review on VCO based ADC in modern deep submicron CMOS technology," *2012 IEEE International Symposium on Radio-Frequency Integration Technology (RFIT)*, Singapore, 2012, pp. 86-88, doi: 10.1109/RFIT.2012.6401622.

[5] S. Maier S, X. Yu, H. Heimpel and A. Pascht, “Wideband base station receiver with analog-digital conversion based on RF pulse width modulation”, *2013 IEEE MTT-S International Microwave Symposium Digest (MTT), Microwave Symposium Digest (IMS), 2013 IEEE MTT-S International*, pp. 1–3. doi: 10.1109/MWSYM.2013.6697415.

[6] R. F. Cordeiro, A. Prata, A. S. R. Oliveira, J. M. N. Vieira and N. B. De Carvalho, "Agile All-Digital RF Transceiver Implemented in FPGA," in *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 11, pp. 4229-4240, Nov. 2017, doi: 10.1109/TMTT.2017.2689739.

[7] Wikipedia contributors, "Quadrature amplitude modulation," *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Quadrature_amplitude_modulation&oldid=1044710342 (accessed September 20, 2021).

[8] Wikipedia contributors, "Pulse-amplitude modulation," *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Pulse-amplitude_modulation&oldid=1016107841 (accessed September 20, 2021).

[9] Wikipedia contributors, "Pulse-width modulation," *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Pulse-width_modulation&oldid=1045325839 (accessed September 20, 2021).

[10] Wikipedia contributors, "Polyphase quadrature filter," *Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Polyphase_quadrature_filter&oldid=1028318229 (accessed September 20, 2021).

[11] Intel Corporation, 2019-10-11, *E-tile Transceiver PHY User Guide*, [pdf] Intel Corporation, Available at: <https://www.intel.com/content/www/us/en/programmable/documentation/kqh1479167866037.html> [Accessed 20 February 2020]

[12] Intel Corporation, 2019-10-25, *L- and H-Tile Transceiver PHY User Guide*, [pdf] Intel Corporation, Available at: <https://www.intel.com/content/www/us/en/programmable/documentation/wry1479165198810.html> [Accessed 20 February 2020]

[13] Intel Corporation, 2020-03-11, *Intel Stratix 10 TX Signal Integrity Development Kit Schematic, revision b1*, [pdf] Shanghai: Intel Corporation, <https://www.intel.in/content/dam/altera-www/global/en_US/support/boards-kits/stratix10/si_tx/s10tx-si-b1.pdf> [Accessed 30 April 2021]

[14] Xilinx, Inc., 2018-08-14, *7 Series FPGAs GTX/GTH Transceivers User Guide*, [pdf] Xilinx inc, <https://www.xilinx.com/support/documentation/user_guides/ug476_7Series_Transceivers.pdf> [Accessed 2020-11-25]