Bachelorthesis

Thema:

Entwurf und Implementierung eines Phasenmodulationsverfahrens mit einem FPGA zur Realisierung der drahtgebundenen Kommunikation einer Bohrlochsonde

von
Friedrich Wiegel

22. August 2013

Studiengang Elektrotechnik
an der
Dualen Hochschule Baden-Württemberg Karlsruhe

Matrikelnummer, Kurs:	1929099, TEL10B2
Ausbildungsfirma:	Karlsruher Institut für Technologie
Betreuer der Ausbildungsfirma:	*Dipl. Ing. Stefan Dietze*
Gutachter der Dualen Hochschule:	*Prof. Dr.-Ing. Gerald Oberschmidt*

Ehrenwörtliche Erklärung

gemäß § 5(2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 18. Mai 2009.

Ich versichere hiermit ehrenwörtlich durch meine Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmitteln verwendet habe.

.....	.....
Ort, Datum	Friedrich Wiegel

Abstract

Diese Bachelorthesis befasst sich mit dem Entwurf und der Implementierung eines digitalen Übertragungsverfahrens mit einem FPGA. Es soll damit eine Vollduplex-Verbindung zwischen dem Server und der bis zur 4000m entfernten Bohrlochsonde sicher gestellt werden. Das als Übertragungsstrecke verwendete Koaxialkabel weist durch seinen Aufbau einen erhöhten Dämpfungskoeffizient auf und wird gleichzeitig zur Stromversorgung der Bohrlochsonde verwendet. Zuerst wurden die Details einer hinreichend robusten Modulationsart und das Kanalzugriffsverfahren mit den entsprechenden Kanälen festgelegt. Anschließend wurde anhand eines Modells die Reaktion des Systems auf unterschiedliche Störungen simuliert. Danach wurden verschiedene Teilverfahren der Modem-Umsetzung getestet und bewertet. Die präferierte Lösung wurde in VHDL realisiert und anschließend revidiert. Zum Schluss wurde die Bitfehlerrate der Übertragung ermittelt.

Abstract

The Duty of this bachelor thesis to develop the design and implementation of a digital communication method using FPGA. A full-duplex connection has to be guaranteed between the server and up to the borehole probe 4000m away. The coaxial cable used as a transmission path has a higher damping coefficient due to its structure and is used as a power supply cable for the borehole probe. First, the details of sufficiently robust modulation and channel access method with the corresponding channels have been specified. Afterwards the reaction of the system on different disturbances was simulated with the system model. Then various methods of the modem implementation were tested and evaluated. The preferred solution was implemented with VHDL and verified. Finally, the bit error rate of the transmission has been measured.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeausschnitte

Abkürzungsverzeichnis

ADC:	Analog Digital Converter
AMBA:	Advanced Microcontroller Bus Architecture
ASK:	Amplitude Shift Keying
AWGN:	Additive White Gaussian Noise
BB:	Basisband
BER:	Bit Error Rate
BPSK:	Binary Phase Shift Keying
CAN:	Controller Area Network
CPU:	Central Processing Unit
DAC:	Digital Analog Converter
DCM:	Digital Clock Managers
DDS:	Direct Digital Synthesis
DQPSK:	Differential Quadrature Phase Shift Keying
DSP:	Digital Signal Processor
DSP:	European Space Agency
FDMA:	Frequency Division Multiple Access
FIFO:	First In - First Out
FIR:	Finite Impulse Response
FPGA:	Field Programmable Gate Array
FSK:	Frequency Shift Keying
GRETH:	Gaisler Research's Ethernet

HiTES:	High Temperature Embedded System
HT:	Hochtemperatur Bereich
I2C:	Inter-Integrated Circuit
IIR:	Infinite Impulse Response
IP-Core:	Intellectual Property Core
JTAG:	Joint Test Action Group
LVDS:	Low Voltage Differential Signal
OOK:	On-Off Keying
PC:	Personal Computer
PLL:	Phase Locked Loop
PN:	Pseudo-Noise
PROM:	Programmable Read Only Memory
PSK:	Phase Shift Keying
QAM:	Quadrature Amplitude Modulation
QPSK:	Quadrature Phase Shift Keying
RRC:	Root-Raised-Cosine
S6Brd02:	Spartan-6 Board version 02
SNR:	Signal-to-Noise Ratio
SoC:	System-On-Chip
SRAM:	Static Random Access Memory
TDMA:	Time Division Multiple Access
TP:	Tiefpass
VHDL:	Very High Speed Integrated Circuit Hardware Description Language

1 Einleitung

Rund 99 Prozent unseres Planeten sind heißer als 1000°C. Dieses Reservoir stellt ein gewaltiges Energiepotenzial dar. Um dieses mithilfe der Geothermie effizient nutzen zu können, werden am Institut für Angewandte Informatik Arbeiten zur Verbesserung der Exploration von Geothermie Bohrungen durchgeführt [1]. Bei der Realisierung geothermischer Projekte werden Informationen aus vielen unterschiedlichen Quellen zu verschiedensten Zwecken verarbeitet. Das Institut arbeitet an einer Sonde, die in den von bis zu 4000m tiefen und 165°C heißen Bohrlöchern Messdaten z.B. von einer Kamera oder Sensoren, sammeln, verarbeiten und an die Oberfläche übermitteln kann. Zu diesem Zweck muss die Sonde unterschiedliche Peripherie wie z.B. hochauflösende Kamera, Sensoren und Aktoren besitzen. Um die entstehenden Datenmengen verarbeiten zu können, soll eine dauerhafte Hochgeschwindigkeitsverbindung mit dem Server sichergestellt werden. Dabei sollen die Energieversorgung und die Kommunikation zwischen der Sonde und dem Host-PC in der Regel über ein Koaxialkabel erfolgen. Bedingt durch die variable Länge und die Übertragungsqualität vorhandener Kabel ist eine eigene Implementierung eines digitalen Übertragungsverfahrens mit einer Nettoübertragungsrate von ca. 1 Mbit/s notwendig. Damit man Hardware ansteuern und Messdaten verarbeiten und versenden kann, hat man sich für die Verwendung eines Embedded Systems mit FPGA entschieden. Die in den Bohrlöchern vorherrschenden Temperaturen und die große Entfernung zu der Oberfläche stellen die Entwickler vor die Herausforderung, die Sonde ausreichend kühlen zu müssen. Je weniger elektronische Komponenten im gekühlten Bereich betrieben werden, desto einfacher ist die Realisierung. Beziehungsweise erlaubt die Verwendung von Hochtemperatur verträglichen Bauelementen bei gleicher Kühlleistung einen längeren Aufenthalt in der Tiefe. Daher wird die Strategie verfolgt, ein HiTES (High Temperature Embedded System) als ein System-On-Chip auf Grundlage eines Spartan-6 FPGA von der Firma Xilinx zu realisieren.

2 Einführung

2.1 Das Projekt HiTES

Das High Temperatur Embedded System (HiTES) dient als Schnittstelle zwischen dem sich auf der Oberfläche befindenden Server, der die Daten sammelt und visualisiert und den Sensoren bzw. der Kamera in der Tiefe. In dem aktuellen Entwicklungsstadium sollten die gesammelten Daten wahlweise, über einen Lichtwellenleiter oder über eine kabelgebundene Übertragungsstrecke, wie z.B. Koaxialkabel, zur Oberfläche gelangen. Die Übertragungsart hängt von dem jeweiligen Projekt ab. Das folgende Blockschaltbild (Abb. 2.1) soll zu einem besseren Verständnis des Projektes beitragen.

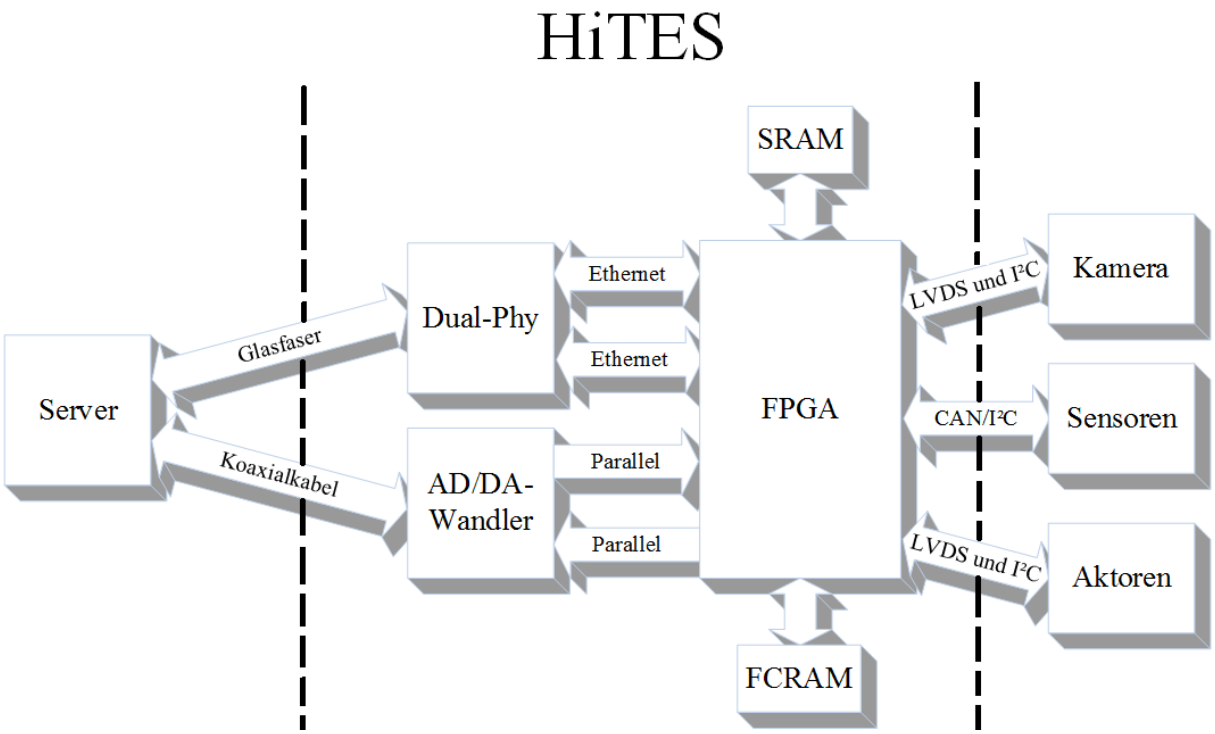

Abbildung 2.1: HiTES Blockdiagramm

2.2 Entwicklungsstand

Um genügend Flexibilität und Leistung zu gewährleisten, entschied man sich das System auf Basis eines FPGA der Firma Xilinx©zu realisieren. Die eingesetzten FPGA der Spartan6 Familie bieten einen optimalen Kompromiss zwischen der Performance, dem Energieverbrauch und der entstehenden Abwärme. Außerdem sind diese in der extended Edition auch für Automotive Temperaturen erhältlich. Ein FPGA bietet den Vorteil, dass das System durch Änderung der vorhandenen IP-Cores leicht an neue Anforderungen angepasst werden kann, ohne das es dafür gleich eine neue Platine entwickelt werden muss. Da HiTES im HT (Hochtemperatur) -Bereich der Sonde untergebracht werden soll, wurde die Platine selbst für den Temperaturbereich von bis zu 170° ausgelegt. Somit müssen nur einzelne Bauteile lokal gekühlt werden [2][3]. Dadurch ist es möglich, die notwendige Kühlleistung gering zuhalten. Die im fünften Praxissemester entworfene HiTES-Platine (S6Brd02) beinhaltet folgend aufgelistete Komponente, die wie auf der Abbildung 2.2 dargestellt platziert sind.

U1: Spartan-6 XC6SLX75 von Xilinx™

U2: 10/100Mb/s Dual Phy von National Semiconductor™

U6: 14-Bit, 80Msps LTC2249 ADC von Liner Technology™

U7: QProflash XQ32P PROM von Xilinx™

U8: 16-Bit, 50Msps LTC1668 DAC von Liner Technology

U4, U5, U10, U11: 4x2MB SRAM IS61WV102416BLL von ISSI™

U12, U13, U14: I2C Temperatursensor MAX31725 von Maxim Integrated®

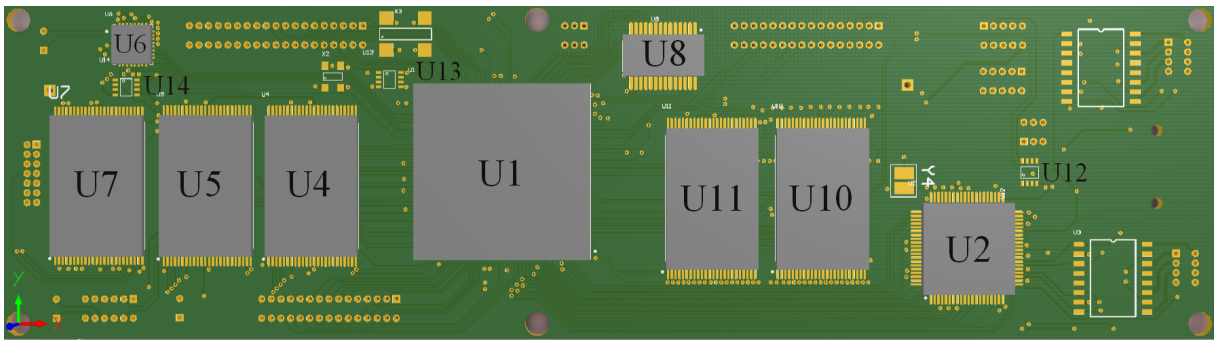

Abbildung 2.2: S6Brd02 Platine Sicht von oben

Die Abbildung 2.3 stellt den momentanen Implementierungsstatus des SoC (System-On-Chip) Designs im FPGA dar. Dabei bildet der Leon3-Prozessor das Herzstück des Systems. Die einzelnen Komponenten wie GRETH (Gaisler Research Ethernet) oder Memory Controller sind über den AMBA-Bus mit der CPU verbunden. Die CPU selbst ist ein Open Source Prozessor basierend auf SPARC-V8-Architektur, der als Soft-Core CPU (VHDL-Design) über die GRLIB-Library (IP-Core Bibliothek von Aeroflex Gaisler™) frei verfügbar ist. Der ursprüngliche Prozessorentwurf entstammt der Europäischen Weltraumorganisation (kurz ESA™), mittlerweile wird dieser aber von der Firma Aeroflex Gaisler™ weiterentwickelt und verbreitet. Die restlichen IP-Cores werden entweder auch von der GRLIB-Library zur Verfügung gestellt oder sind Eigenentwiclkung der Abteilung.

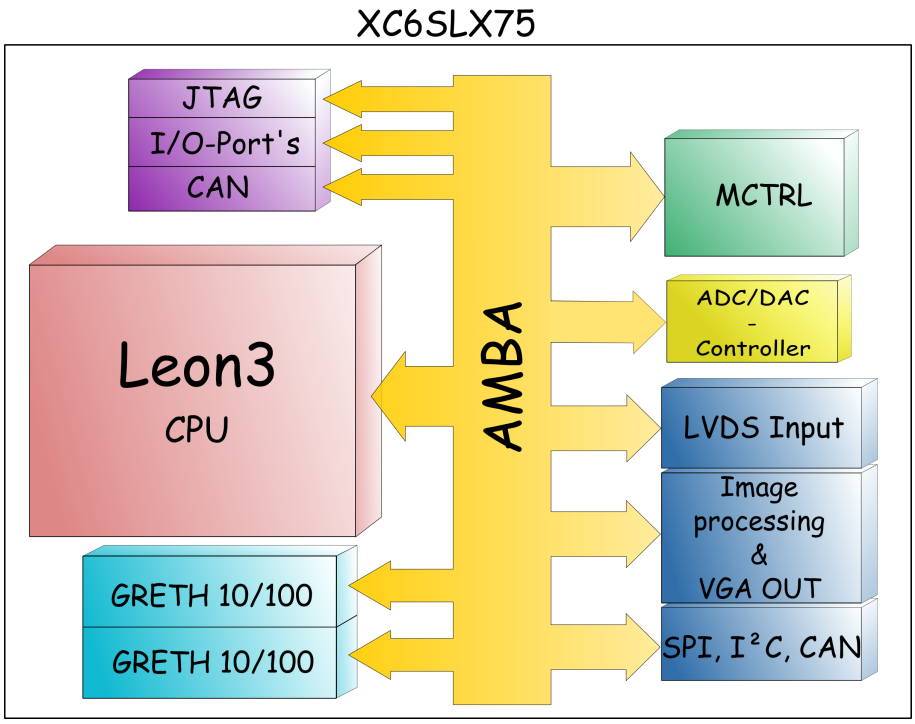

Abbildung 2.3: Übersicht über die implementierten IP-Cores

2.3 Ziele der Bachelorarbeit und Rahmenbedingungen

Diese Bachelorarbeit befasst sich mit dem Entwurf und Implementierung eines digitalen Modulationsverfahrens in VHDL (Very High Speed Integrated Circuit Hardware Description Language). Dabei sollen zuerst die passende Modulationsart und die geeigneten Frequenzbereiche ausgewählt werden. Anschließend erfolgt die Simulation der Übertragungsstrecke in Ptolemy II. Der Hauptteil der Arbeit besteht in der Implementierung des Modems in VHDL. Die folgend aufgelisteten Rahmenbedingungen legen die Anforderungen an das zu implementierende Modem fest.

- Übertragungskanal
  - Max. Länge der Übertragungsstrecke beträgt 4000m
  - Max. Umgebungstemperatur liegt bei 165°C
  - Übertragung erfolgt über einen Koaxialkabel (Impedanz: 54Ω, Dämpfung bei 8MHz: 33dB/km)
- Nettoübertragungsratte
  - Uplink von der Sonde zum Server: 1Mbit/s
  - Downlink von dem Server zur Sonde: 19,2kbit/s
- Mögliche Störquelle
  - Die Datenübertragung erfolgt über dasselbe Koaxialkabel wie auch die Energieversorgung der Sonde. Da die Spannungsversorgung fast bei Leistungsanpassung erfolgt, können hohe Spannungsschwankungen zu erheblichen Störungen bei der Übertragung führen.
  - Die eingesetzten DC/DC-Wandler verursachen schmalbandige Störungen, die schätzungsweise bei 150kHz liegen
  - In der Sonde sind auch Elektromotoren verbaut, die auch Störungen verursachen

Da über die einzelnen Einsatzfälle des Modems noch keine genauen Daten vorliegen und die Messungen der möglichen Störquellen erst später durch den Entwicklungspartner durchgeführt werden, muss das Modem leicht an die neuen Anforderungen anpassbar sein.

3 Auswahl eines geeigneten Modulationsverfahrens

Unabhängig davon ob eine Übertragung leitungsgebunden oder über einen Funkweg stattfindet, erfolgt bei der Modulation die Umsetzung des Quellensignals in eine andere Signalform und somit auch in einen anderen Frequenzbereich. Dadurch ist es möglich, das Ursprungssignal an die gegebenen Eigenschaften des Übertragungskanals anzupassen. Dabei werden alle Signale auf ihrem Weg von dem Sender zum Empfänger mit unvermeidlichen Störungen, wie z.B. thermischem Rauschen konfrontiert. Um diese Störungen so gering wie möglich zu halten, ist es wichtig, zumindest eine Vorstellung über die gegebene Übertragungsstrecke zu haben. Denn nur so lässt sich ein Modulationsverfahren auswählen, dass so wenig wie möglich gestört wird.
Die meisten gängigen Modulationsarten erfolgen mithilfe eines Sinusträgers, da dieser positive Eigenschaften besitzt, auf die später genauer eingegangen wird. Ausgehend von der allgemeinen Funktion eines Sinusträgers

$$x(t) = \hat{a_T} * \sin(2\pi f_T + \varphi) \tag{3.1}$$

gibt es insgesamt genau drei Möglichkeiten dem Träger die Information, die es zu übermitteln gilt, aufzuprägen.

1. Durch das Eingreifen in die Amplitude des Trägers $\hat{a_T}$. Diese Art der Modulation wird **Amplitude Shift Keying** (ASK) genannt.
2. Durch die Veränderung der Trägerfrequenz f_T: **Frequency Shift Keying** (FSK).
3. Die letzte Möglichkeit beruht auf der Verschiebung der Phase des Trägers φ und wird **Phase Shift Keying** (PSK) genannt.

Alle diese drei Modulationsarten haben ihre Vor- und Nachteile, die im folgendem Kapitel kurz vorgestellt werden.

3.1 Amplitude Shift Keying

Um die Erklärung der Funktionsweise der ASK so einfach wie möglich zu halten, besteht das Codewort des zu modulierenden Signals nur aus einer Binärstelle (1 bzw. 0). Dieses

Verfahren wird auch als On-Off-Keying (OOK) bezeichnet. Wie man der Funktion 3.2

$$u_{ASK}(t) = \hat{A}_T * c(t) * \sin(2\pi\omega_T) \tag{3.2}$$

entnehmen kann, wird der Träger in Abhängigkeit von dem momentanen Zustand des Codewortes $c(t)$ entweder an- oder ausgeschaltet. Die dazugehörige Zeitfunktion ist in der Abbildung 3.1 dargestellt.

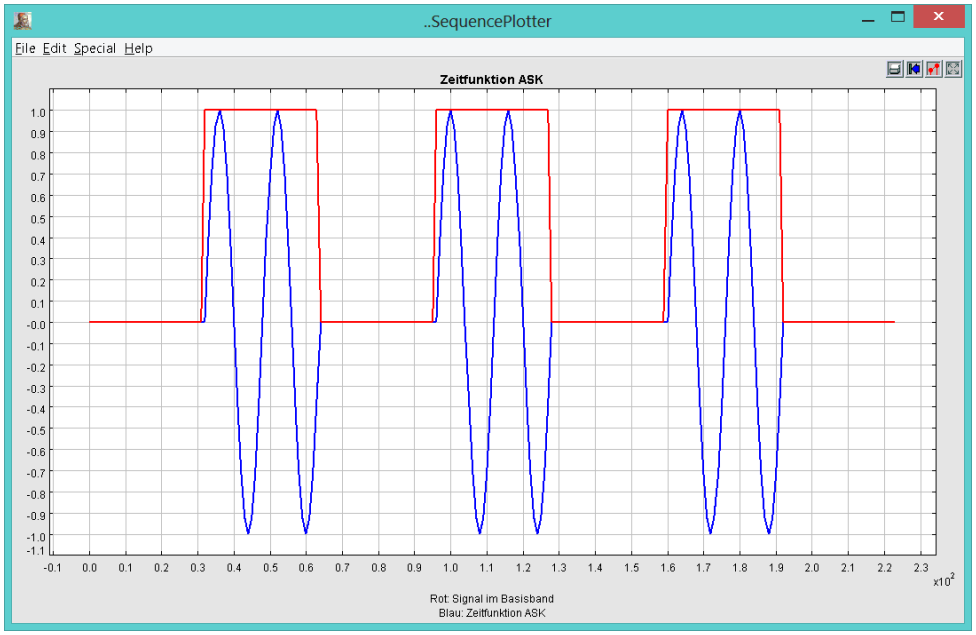

Abbildung 3.1: Das Signal im Basisband und die ASK

Die Schrittdauer beträgt dabei $T_s = T_{Bit}$ und die Perioden Dauer $T_S = 2 * T_s$. Betrachtet man dabei die zur Übertragung notwendige Bandbreite, ergibt sich folgender Zusammenhang[4]:

$$B_{HF} = 2 * \frac{1}{2 * T_s} = f_{Bit} \tag{3.3}$$

Das bedeutet, dass die notwendige Bandbreite zum Übertragen des Signals genau der Frequenz des Signals im BB (Basisband) entspricht.
Theoretisch gesehen ist es natürlich auch möglich, mehr als nur ein Bit pro Periode zu überragen. Dafür muss die gesamte Amplitude des Signals in so viele diskrete Schritte unterteilt werden, wie viele Zustände das gültige Codewort annehmen kann. Das hat aber denn Nachteil, dass der Unterschied zwischen zwei gültigen Codewörtern kleiner wird und die Übertragung somit auch viel empfindlicher gegenüber Störungen wird.

3.2 Frequency Shift Keying

Bei einer FSK wird jedem Sendesymbol eine bestimmte Sendefrequenz zugewiesen. Dabei muss die Anzahl der verwendeten Sendefrequenzen immer ganzzahlig sein. Eine binäre FSK bzw. 2-FSK ist die simpelste Variante der Frequenzumtastung. Wie der Name schon sagt, wird hierbei zwischen zwei festen Frequenzen f_0 und f_1 in Abhängigkeit von dem zu sendenden Bit umgeschaltet. Es gilt folgender Zusammenhang [4]:

$$Trägerfrequenz : f_T = \frac{f_0 + f_1}{2} \tag{3.4}$$

$$Frequenzhub : \Delta f_T = \frac{|f_0 - f_1|}{2} \tag{3.5}$$

$$f_0 = f_T - \Delta f_T \tag{3.6}$$

$$f_1 = f_T + \Delta f_T \tag{3.7}$$

Mit diesen Definitionen ergibt sich ein Sendesignal mit der Funktion:

$$u_{FSK}(t) = \hat{A}_T * \sin(2\pi(f_T \pm \Delta f_T) * t) \tag{3.8}$$

Dabei ist die Mittenfrequenz, bzw. Trägerfrequenz, die Frequenz der Schwingung die sich ohne Modulation einstellen würde, also bei $\Delta f_T = 0$. Die Abbildung 3.2 stellt das Sendesignal im Basisband die zugehörige Zeitfunktion der 2-FSK dar.

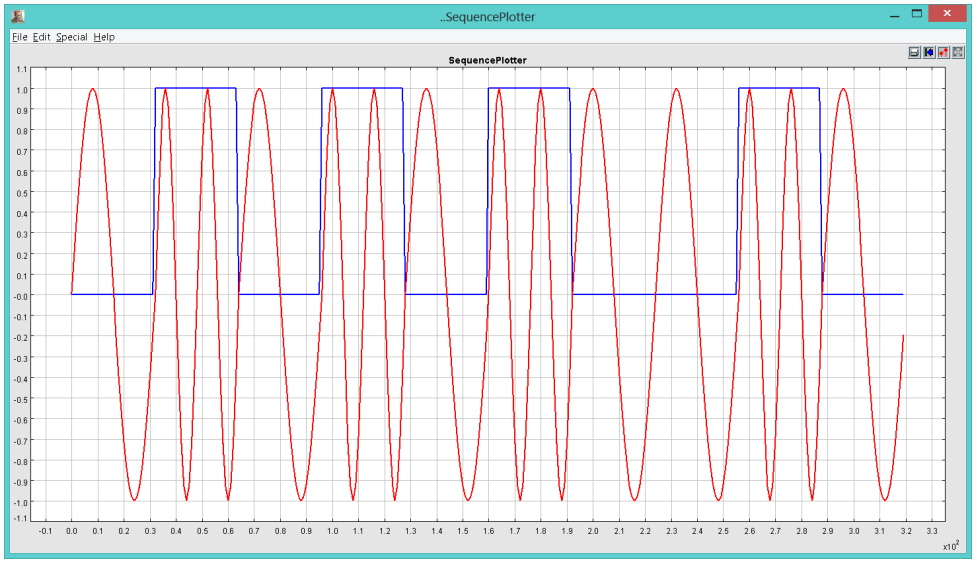

Abbildung 3.2: Das Signal im Basisband und die 2-FSK

Wie auch bei der OOK entspricht die zur Übertragung des Signals notwendige Bandbreite der Frequenz mit der das Signals im Basisband übertragen wird. Das heißt $B_{HF} = f_{Bit}$.

3.3 Phase Shift Keying

Bei dieser Art der Übertragung wird die Phase eines Trägers moduliert. Die binäre PSK (BPSK) bildet die Grundlage für alle Phase Shift Keying Modulationsarten und wird hier kurz vorgestellt. Das Codewort $c(t)$ einer BPSK besteht nur aus einer Binärstelle die entweder „0“ oder „1“ sein kann. Bei der Modulation geht es darum die beiden Zustände in die entsprechende Phase (z.B. 0° und 180°) der Schwingung umzusetzen. Dabei ergibt sich folgender Zusammenhang [4]:

$$u_{PSK}(t) = \hat{A_T} * \sin(2\pi f_T t + c(t) * \pi) \tag{3.9}$$

Die Abbildung 3.3 stellt das Sendesignal im Basisband und die zugehörige Zeitfunktion der BPSK dar.

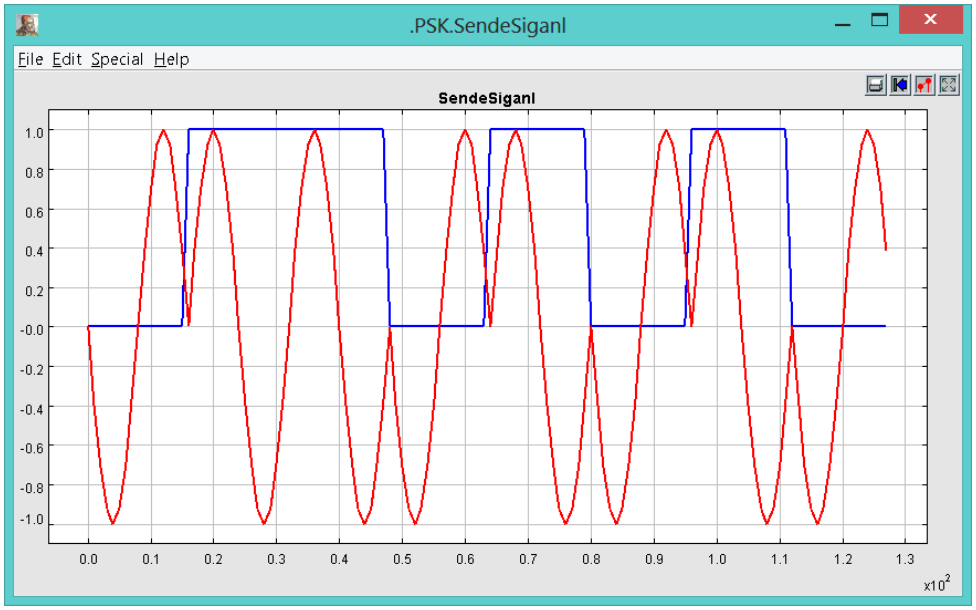

Abbildung 3.3: Das Signal im Basisband und die BPSK

Wie auch bei den beiden vorherigen Verfahren entspricht die zum Übertragen des Signals notwendige Bandbreite „B_{HF}“ der Frequenz des Signals „f_{Bit}“ im Basisband.

3.4 Festlegung des Modulationsverfahrens

Auf der Suche nach einer geeigneten Modulationsart sind neben der notwendigen Bandbreite auch Überlegungen zur Störanfälligkeit des Signals anzustellen. Bedingt durch die hohen Spannungsschwankungen auf dem Übertragungsweg wird von der Verwendung der ASK abgesehen. Auch die FSK kann eventuell durch die selektiven Störungen zu sehr beeinträchtigt werden. Außerdem zeigt sich dass, die BPSK verglichen mit beiden anderen

Verfahren eine viel geringere Bitfehlerwahrscheinlichkeit aufweist. Diese Feststellung kann auch der Abbildung 3.4 entnommen werden [4]. Hierbei ist die Bitfehlerwahrscheinlichkeit p_e aller drei Verfahren über den Signal/Rausch-Abstand $\frac{E}{N_0}$ dargestellt.

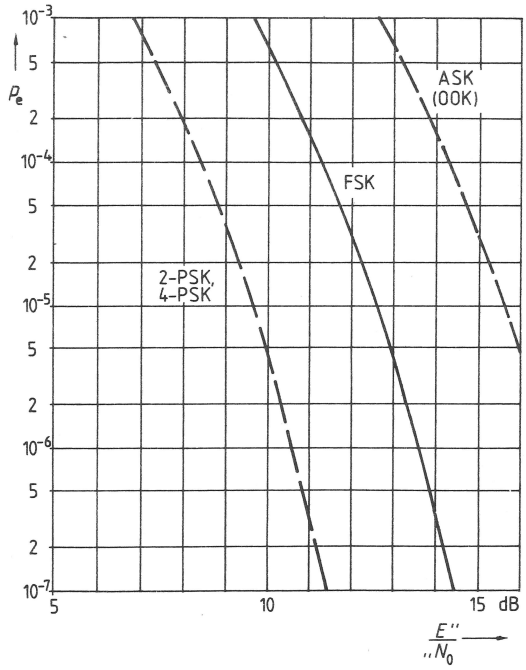

Abbildung 3.4: Bitfehlerwahrscheinlichkeit von BPSK, ASK (OOK) und (binär) FSK [4]

Ausgehend von den Anforderungen, die in Kapitel 2.3 vorgestellt wurden ergibt sich die Schlussfolgerung, dass die PSK durch ihre Robustheit für die gestellte Aufgabe die geeignetste Modulationsart darstellt. Um jetzt noch die zur Verfügung stehende Bandbreite von ca. 1 MHz effizient ausnutzen zu können, soll die Übertragung der Daten entweder mit der QPSK (**Q**uadrature **P**hase **S**hift **K**eying) oder der DQPSK (**D**ifferential QPSK) erfolgen. Bei diesen Modulationsarten beträgt die Dauer eines Signalschrittes $2 * T_{Bit}$. Das wiederum führt dazu, dass sich die zur Übertragung notwendige Bandbreite um die Hälfte reduziert. Dabei wird pro Trägerperiode genau ein Symbol übertragen. Von der Verwendung der Quadratur-Amplitudenmodulation (QAM), bei der gleichzeitig die Amplitude und die Phase einer Trägerschwingung moduliert werden, wird aus zwei Gründen abgesehen: Zum einen wegen der hohen Empfindlichkeit gegenüber Spannungsschwankungen verursacht durch die Versorgungsströme und zum anderen wegen der Komplexität und damit verbundenen Realisierungsaufwand.

3.5 Kanalzugriffsverfahren und Frequenzbereiche

Da die Steuerbefehle von dem Server zur Sonde und die gesammelten Daten von der Sonde zum Server übermittelt werden, ist es wichtig, ein Kanalzugriffsverfahren festzulegen. Denn nur so können mehrere Sender bzw. Empfänger dasselbe Medium „gleichzeitig“ verwenden. Es wurde entschieden, das FDMA-Verfahren (Frequency Division Multiple Access) einzusetzen. Hierbei werden unterschiedlichen Kanälen unterschiedliche Frequenzbereiche zugewiesen. Durch die verwendeten Schutzbänder wird die zur Verfügung stehende Bandbreite zwar nicht so gut ausgenutzt wie bei der TDMA Technik (Time Division Multiple Access), dafür bietet dieses Verfahren einen echten Vollduplex-Betrieb. Außerdem kann der Kanal auch bei TDMA nicht zu 100% ausgenutzt werden, da auch hier Schutzintervalle unerlässlich sind [5]. Um einzelne Kanäle sauber voneinander trennen zu können, werden bei der FDMA-Technik aufwendigere Sende bzw. Empfangsfilter benötigt. Dafür entstehen aber keine Timing Probleme, die zur Störung der Videoübertragung führen könnten. Außerdem muss bei der TDMA-Technik ein Synchronisationsprotokoll dafür sorgen, dass die Datenübertragung kollisionsfrei stattfindet. Das führt dazu, dass die Nettoübertragungsrate etwas sinkt.
Aktuell soll die zur Verfügung stehende Bandbreite wie in der Abbildung 3.5 dargestellt, in einzelne Kanäle unterteilt werden.

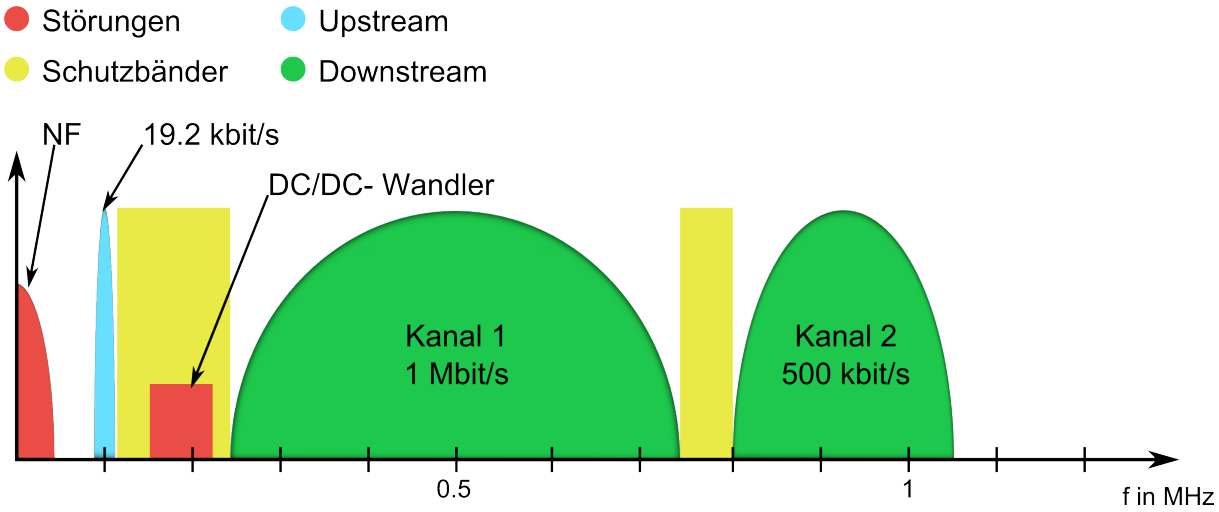

Abbildung 3.5: Festgelegte Frequenzbereiche

Kanal 1 im Uplink entspricht dem Basiskanal mit 1Mbit/s und kann eventuell mit dem Kanal 2 auf 1,5Mbit/s Bruttodatenrate erweitert werden. Bei den hier vorgestellten Datenraten ist eine Verminderung durch eine Kanalcodierung einzurechnen.

4 Modulation bzw. Demodulation mit QPSK/DQPSK

4.1 Modulation

4.1.1 QPSK Modulation

Wie schon vorher vorgestellt, werden bei der QPSK bzw. DQPSK zwei Bits pro Symbol übertragen. Dadurch ist es möglich die zu Übertragung des Signals notwendige Bandbreite zu halbieren. Zwei Bits bedeuten das die Trägerschwingung vier mögliche Phasenzustände (45°,135°, 225° und 315°) annehmen kann. Das dazu gehörige Konstellationsdiagramm ist in der Abbildund 4.1 dargestellt.

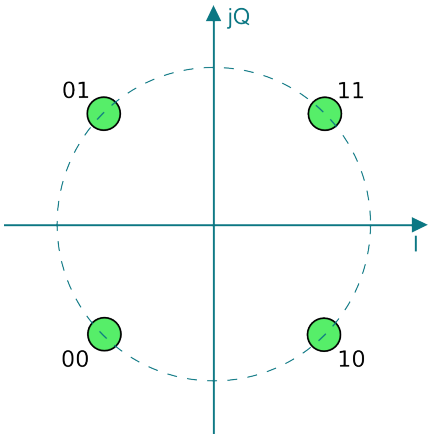

Abbildung 4.1: Konstellationsdiagramm QPSK

Als Träger kommen zwei kosinusförmige Signale derselben Frequenz zum Einsatz. Dabei wird ein Signal um 90° verschoben, damit diese bei gleicher Frequenz orthogonal zueinanderstehen. Anschließend kann einer mit der Information aus dem I-Zweig und der andere mit der Information aus dem Q-Zweig moduliert werden. Die beiden entstandenen Signale können dann den selben Kanal gleichzeitig verwenden. Das QPSK-Signal ist im Endeffekt die Addition zweier PSK-Signale [4]. Mathematisch gesehen ergibt sich folgender Zusammenhang:

$$Parallele r Bitstrom : I + jQ \tag{4.1}$$

$$Trägersignale : \cos(\omega_T t) + j \sin(\omega_T t) \tag{4.2}$$

$$S(t) = (I + jQ) * (\cos(\omega_T t) + j \sin(\omega_T t)) \tag{4.3}$$

$$= I \cos(\omega_T t) + jI \sin(\omega_T t) + jQ \cos(\omega_T t) - Q \sin(\omega_T t) \tag{4.4}$$

$$u_{QPSK}(t) = \mathrm{Re}\{S(t)\} = \mathrm{Re}\{I \cos(\omega_T t) + jI \sin(\omega_T t) + jQ \cos(\omega_T t) - Q \sin(\omega_T t)\} \tag{4.5}$$

$$u_{QPSK}(t) = I \cos(\omega_T t) - Q \sin(\omega_T t) \tag{4.6}$$

Zur Erzeugung der Daten für den I- und Q-Zweig muss der seriell vorhandene Bitstrom mit der Periode T_{Bit} parallelisiert werden. Nach der Parallelisierung beträgt die Symboldauer $T_S = 2 * T_{Bit}$. Dieser Zusammenhang führt dazu, dass die notwendige Übertragungsbandbreite um die Hälfte reduziert wird.

$$B_{HF} = f_S = \frac{1}{T_S} = \frac{1}{2 * T_{Bit}} = \frac{1}{2} * f_{Bit} \tag{4.7}$$

Bild 4.2 zeigt die Spektren der BPSK und QPSK bei der selben Übertragungsrate im direkten Vergleich.

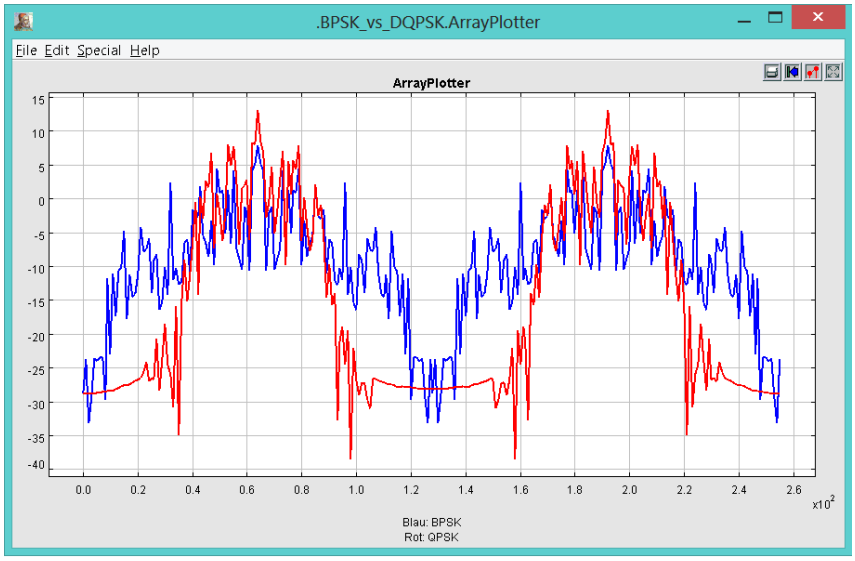

Abbildung 4.2: Spektren der BPSK(Blau) und QPSK(Rot)

4.1.2 Bandbegrenzung

Die vorher berechnete Übertragungsbandbreite stellt nur einen theoretischen Wert dar. Praktisch gesehen entsteht bei der Modulation mit einem rechteckigen binär Signal eine

$sinc(x)$-Funktion (rote Kurve, Abb. 4.3). Diese klingt zwar schnell ab, kann aber immer noch zu Störungen in den Nachbarkanälen führen.

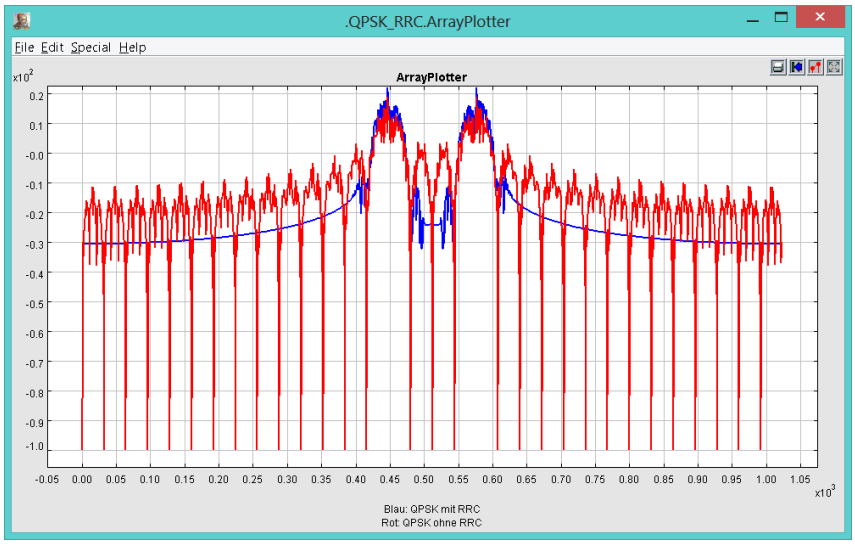

Abbildung 4.3: Spektrum einer QPSK ohne und mit einen Impulsformungsfilter

Um dem entgegen zu wirken, werden vor den Modulatoren Cos-Roll-Off-Filter mit einer $\cos^2$-förmig abfallenden Übertragungsfunktion eingesetzt. Diese begrenzen das Spektrum des Basisbandsignals auf einen endlichen Frequenzbereich. Mit einem weiteren Trick ist es auch möglich das SNR im Empfänger zu verbessern. Dafür muss das Cos-Roll-Off-Filter auf zwei identische Root-Raised-Cosine-Filter im Sender und Empfänger aufgeteilt werden. Dadurch entsteht zusätzlich zu dem Impulsformungsfilter noch ein Matched-Filter welches im Endeffekt auch das SNR durch Bandbreitenbegrenzung verbessert. Durch den Einsatz der Impulsformungsfilter entspricht die praktisch notwendige Übertagungsbandbreite $B_{HF} = 0.7 * fbit$ fast der theoretischen berechneten $B_{HF_N} = 0.5 * fbit$ (blaue Kurve Abb. 4.3).

4.2 Übergang von QPSK zu DQPSK

Bei der Demodulation eines QPSK-Signals treten zwei schwerwiegende Probleme auf:

1. Es existiert eine generelle Phasenunsicherheit von 180°, die zu einem Vertauschen der demodulierten Binärelemente führen kann.

2. Die Demodulation funktioniert nur dann, wenn die Frequenz und Phase des lokalen Sinusgenerators zu 100% mit der Trägerfrequenz übereinstimmt.

Das erste Problem kann leicht durch das Versenden einer bekannten Symbolsequenz (Pilot) beseitigt werden. Das zweite Problem ist schwerwiegender und ist auf die Asynchronität der beiden Quarze im Sender und Empfänger zurückzuführen. Diese lässt sich auf

Grund von Fertigungstoleranzen, Alterungsprozessen und Temperaturunterschieden nicht vermeiden. Und da die Quarze für den jeweiligen Sinusgeneratoren als Taktquelle dienen, können diese nie 100% synchron sein.
Die Information steckt aber nicht nur in der Absolutphase des Signals, sondern auch in der Differenz zweier aufeinanderfolgender Zustände. Somit ist es möglich, mit Hilfe der Phasendifferenzcodierung die beiden Probleme zu umgehen; Genau dieses leistet die DQPSK.
Mittels eines digitalen Phasendifferenzrechners werden die I- und Q-Signale in die differenzcodierte Signale I_D und Q_D umgewandelt. Dazu muss das aktuelle Signalpaar $(I_n + jQ_n)$ mit dem um einen Takt verschoben differentiellen Signalpaar $(I_{D_{n-1}} + jQ_{D_{n-1}})$ komplex multipliziert werden. Dieses Vorgehen führt aber dazu, dass die Werte am Ausgang des Phasendifferenzrechners immer weiter ansteigen. Um dem entgegen zu wirken muss das Ergebnis in dem Rückführungszweig mit $0,5 - j0,5$ multipliziert werden. Das Blockschaltbild des Phasendifferenzrechner ist in der Abbildung 4.4 dargestellt.

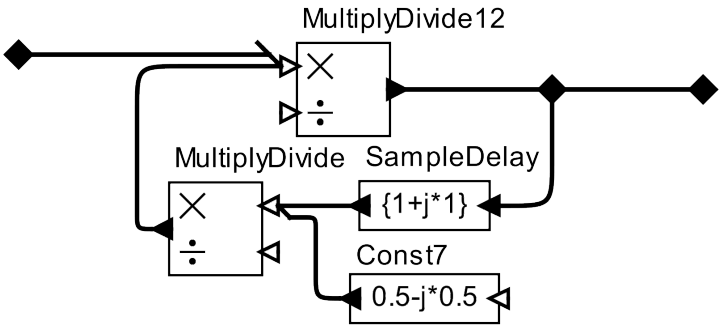

Abbildung 4.4: Blockschaltbild Phasendifferenzrechner im Sender

Die DQPSK hat aber auch einen Nachteil. Im Vergleich zur QPSK ist ein um 2,3dB größerer SNR notwendig. Dies ist darauf zurück zuführen, dass bei der Dekodierung sowohl das verzögerte als auch das unverzögerte Signal mit Rauschen überlagert ist.

4.3 Demodulation

4.3.1 Kohärente Demodulation

Um das empfangene Signal auf der Empfängerseite weiterverarbeiten zu können, wird dieser erneut in seine Basisbandlage gebracht. Dies geschieht, zumindest bei diesem Entwurf, mithilfe der kohärenten, also synchronen Demodulation. Nimmt man an, dass das DQPSK Signal idealerweise rauschfrei am Empfänger ankommt, so erfolgt der Demodulationsprozess in folgender Reihenfolge:

1. Das empfangene DQPSK Signal wird mit den beiden Signalen $\cos(\omega_T * t + \varphi)$ und $-\sin(\omega_T * t + \varphi)$, $\varphi = 0$ des Lokaloszillators multipliziert.

$$S(I) = (I_D \cos(\omega_T t) - Q_D \sin(\omega_T t)) * \cos(\omega_T t) \tag{4.8}$$

$$= \frac{1}{2} I_D (1 + \cos(2\omega_T t)) - \sin(2\omega_T t) \tag{4.9}$$

$$S(Q) = (I_D \cos(\omega_T t) - Q_D \sin(\omega_T t)) * -\sin(\omega_T t) \tag{4.10}$$

$$= \frac{1}{2} Q_D (1 - \cos(2\omega_T t)) - \sin(2\omega_T t) \tag{4.11}$$

2. Als nächstes werden die Signalanteile mit der doppelten Frequenz mit einem Tiefpasses weg gefiltert .

$$S(I) = LPF\{\frac{1}{2} I_D (1 + \cos(2\omega_T t)) - \sin(2\omega_T t)\} \tag{4.12}$$

$$S(I) = \frac{1}{2} I_D \tag{4.13}$$

$$S(Q) = LPF\{\frac{1}{2} Q_D (1 - \cos(2\omega_T t)) - \sin(2\omega_T t)\} \tag{4.14}$$

$$S(Q) = \frac{1}{2} Q_D \tag{4.15}$$

3. Wurden bei der Modulation Impulsformungsfilter verwendet, so müssen die beiden Signale auch im Empfänger die gleichen Root-Raised-Cosine-Filter passieren um die Impulsformung wieder rückgängig zu machen.

4. Als nächstes erfolgt die Umwandlung differentieller Signale in decodierte I- und Q-Signale. Dies geschieht in einem Phasendifferenzrechner. Dazu wird das aktuelle $(I_{D_n} + jQ_{D_n})$ Signal, durch einen um ein Takt verzögerten Signal $(I_{D_{n-1}} + jQ_{D_{n-1}})$ geteilt.

5. Zum Schluss erfolgt die Wandlung der parallelen I und Q Signale in einen seriellen Datenstrom.

4.3.2 Trägerfrequenz- bzw. Symboltaktrückgewinnung

Bei den meisten Modulationsarten funktioniert die kohärente Demodulation nur dann gut, wenn die Frequenz und die Phase des Lokaloszillators mit der Frequenz und Phase der Trägerwelle übereinstimmen. Da es in Wirklichkeit bedingt durch das Quarzproblem (Kapitel 4.2) nie der Fall ist, müssen die beiden Parameter im Empfänger wieder hergestellt werden. Abhängig von der Modulationsart gibt es unterschiedliche Möglichkeiten zur Trägerrückgewinnung [5].

- Trägerrückgewinnung mit einer PLL aus einem zugesetzten Trägerrest.
- Bei (D)BPSK oder (D)QPSK kann der Träger aus dem modulierten Signal abgeleitet werden. Zum Beispiel mit Costas-Loop oder einer Quadrierschleife.

Bei der DQPSK steckt die Information nicht in der Absolutphase, sondern in der Phasendifferenz. Das hat zur Folge, dass die Demodulation auch dann funktioniert, wenn die beiden Oszillatoren nicht synchron sind. Dies führt dazu, dass die Konstellation zwar gedreht wird, aber die Entscheidungen richtig getroffen werden, solange die Differenzfrequenz Δf den maximal zulässigen Wert Δf_{MAX} nicht überschreitet.

$$\Delta f_{MAX_{DQPSK}} = \frac{1}{8} * f_C > \Delta f \tag{4.16}$$

Die genauen Parameter (Frequenz und Phase) des Symboltakts sind bei einer DQPSK aber wiederum unerlässlich. Denn im Empfänger müssen die differenziellen Signale I_D und Q_D wieder decodiert werden. Dies kann in dem Phasendifferenzrechner nur mit dem richtigen Symboltakt erfolgen. Der Symboltakt kann z.B. mit einer Early/Late-Loop zurück gewonnen werden. Wie das genau geschieht, wird in dem nächsten Kapitel erklärt.

5 Konzeption und Simulation

Das Bild 5.1 zeigt das Blockschaltbild des DQPSK-Modems mit der dazu gehörigen Übertagungsstrecke.

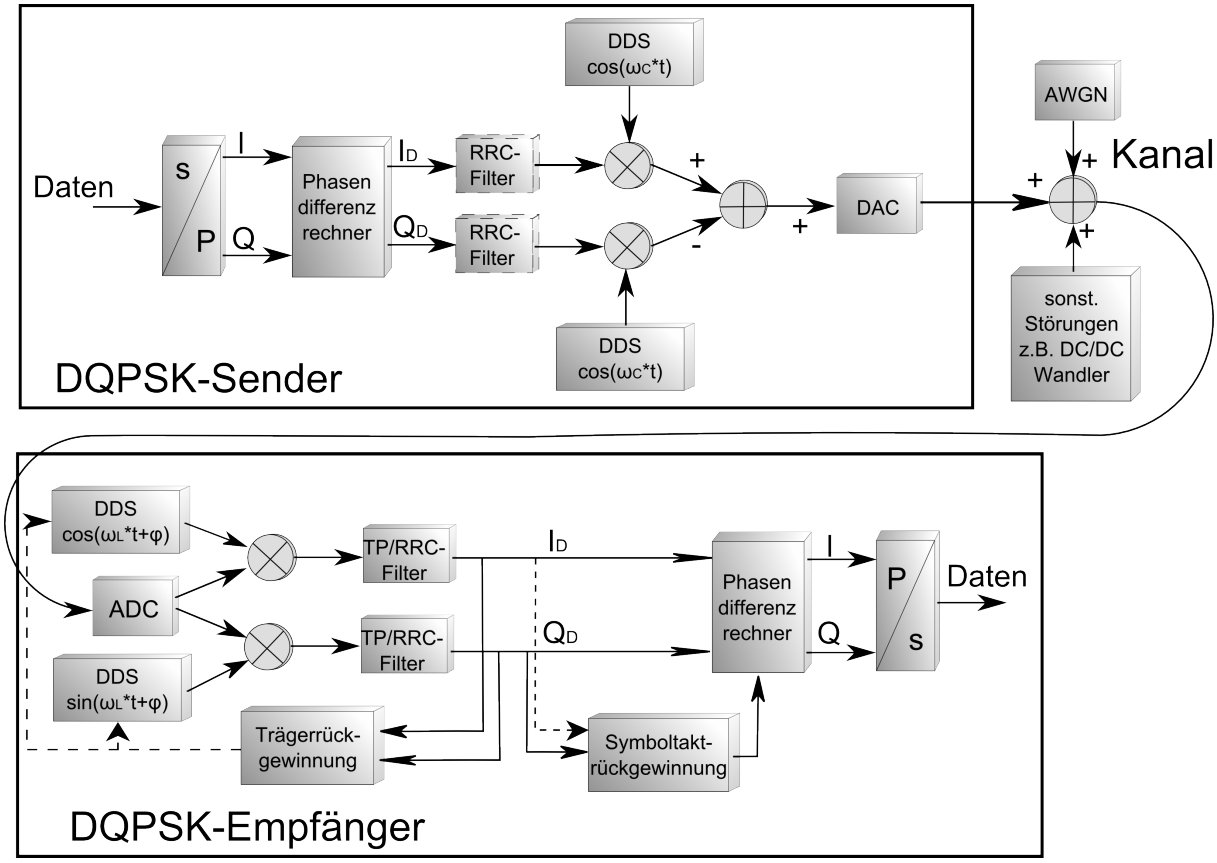

Abbildung 5.1: Blockschaltbild des DQPSK-Modems

Das Modem wird genau so modular aufgebaut wie auf der Abbildung 5.1 dargestellt, dadurch wird zum einen der Implementierungsvorgang erleichtert und zum anderen können leichter Anpassungen bei neuen Anforderungen durchgeführt werden. Die gestrichelten Pfade stellen alternative Lösungen dar.

5.1 Simulation und Verifikationsprogramme

5.1.1 Ptolemy II

Vor der eigentlichen Implementierung des Modems in VHDL wird zuerst ein Modell des Übertragungssystems in Ptolemy II erstellt.
„Ptolemy II ist ein auf Java basierendes Open Source Programm der UC Berkeley“ [14]. Dieses ermöglicht Modellierung und Gleitkomma-Simulation von verschiedenen Systemen. Für den Entwurf von Modellen verfügt das Programm über einen grafischen Editor mit einer Vielzahl von vorgefertigten Ausführungseinheiten (Actors).
Das erstellte Modell (Abb. 5.2) bietet die Möglichkeit, das Verhalten des Systems auf Änderung der Übertragungsparameter zu überprüfen. Außerdem können Auswirkungen unterschiedlicher Störungen simuliert werden. Bedingt durch die Funktionsweise des Programms ist es leider unmöglich, den Sender und Empfänger mit unterschiedlichen Abtastfrequenzen arbeiten zu lassen. Somit ist es schwierig, das Verhalten des Systems bei asynchronen Systemtakten zu simulieren.

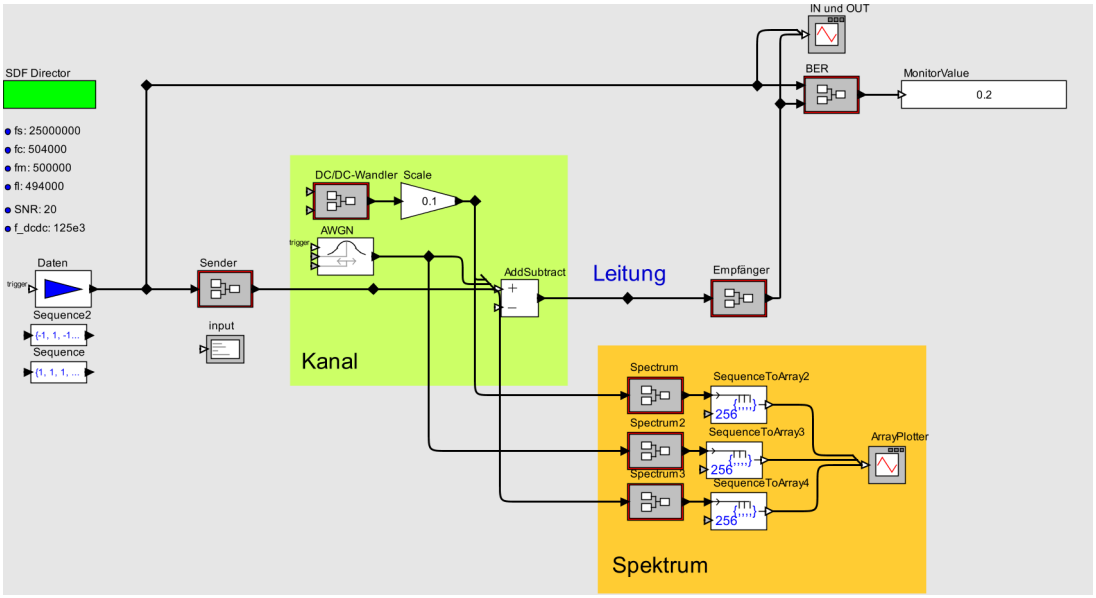

Abbildung 5.2: Modell der Übertragungsstrecke in Ptolemy II

5.1.2 ISim-Simulator

Der ISim-Simulator von der Firma Xilinx™ ist ein Hardware Description Language (HDL) Simulator für VHDL Designs. Damit werden die Designs auf ihre Funktionalität untersucht. Und es bietet einem die Möglichkeit zu prüfen, wie sich die Quantisierung des Signals auf das Verhalten des Systems auswirkt.
Zum Simulieren des Designs ist eine Testbench notwendig. Mit dieser werden die Simulationsparameter und Eingangsvariablen festgelegt. Leider wird beim Erstellen der Simulation

nicht derselbe Compiler verwendet wie bei der Synthese, deswegen kann die Simulation zwar einwandfrei funktionieren, aber das synthetisierte Design nicht, weil, z.B. irgendetwas weg optimiert wurde. Außerdem werden bei der Simulation die Signallaufzeiten nicht berücksichtigt.

5.1.3 ChipScope

ChipScope ist ein softwarebasierender Logikanalysator, dass in den FPGA als IP-Core mit eingebunden wird. Durch das Verwenden des „integrated logic analyzer“ (ila) in dem bestehende VHDL Design können sowohl die Ein-und Ausgänge als auch interne Signale analysiert werden [15]. Da ChipScope zusammen mit dem Anwender-IP-Core auf dem FPGA gebunden wird, entstehen auch einige Einschränkungen. Der Speicher für die Abtastwerte ist durch die zur Verfügung stehende Elemente des FPGA beschränkt. Da die Abtastrate der Designfrequenz entspricht, können Glitches nicht detektiert werden. Es kann auch passieren, dass die Funktionalität des Designs durch Einbinden des ChipScope verändert wird. Besonders zeitkritische Implementierungen sind oft davon betroffen.
Die Konfiguration des ChipScopes erfolgt über einen grafischen Editor, wo man die Triggerquelle und die Signale, die man sehen möchte, auswählt.

5.2 Modulator

Der für die Simulation in Ptolemy II erstellter Sender ist wie im Bild 5.3 dargestellt aufgebaut. Die verwendete Abtastfrequenz beträgt 25MHz. Die Trägerfrequenz liegt bei 500kHz. Zum Simulieren der Frequenzabweichung der beiden Oszillatoren wird die Trägerfrequenz in der Simulation zwischen 504kHz und 494kHz variiert. Die Koeffizienten für die Root-Raised-Cosine-Filter werden mit Matlab™ berechnet. Der Roll-Off-Faktor liegt bei 0.22 und die Cut-Off Frequnz beträgt 500kHz. Die beiden FIR Filter sind 101. Ordnung.

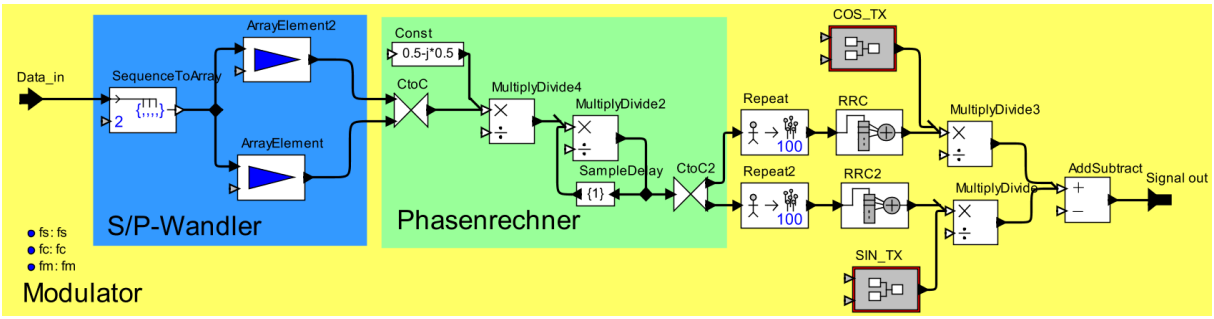

Abbildung 5.3: Aufbau des Senders in Ptolemy II

Schaut man sich das Spektrum des Sendesignals an (Abb. 5.4), so sieht man das bei der

Übertragung mit RRC-Filter(rot) die Nebenmaxima sehr viel schneller abklingen als bei der Übertragung ohne die Impulsformungsfilter (blau). Somit werden auch die Nachbarkanäle weniger gestört.

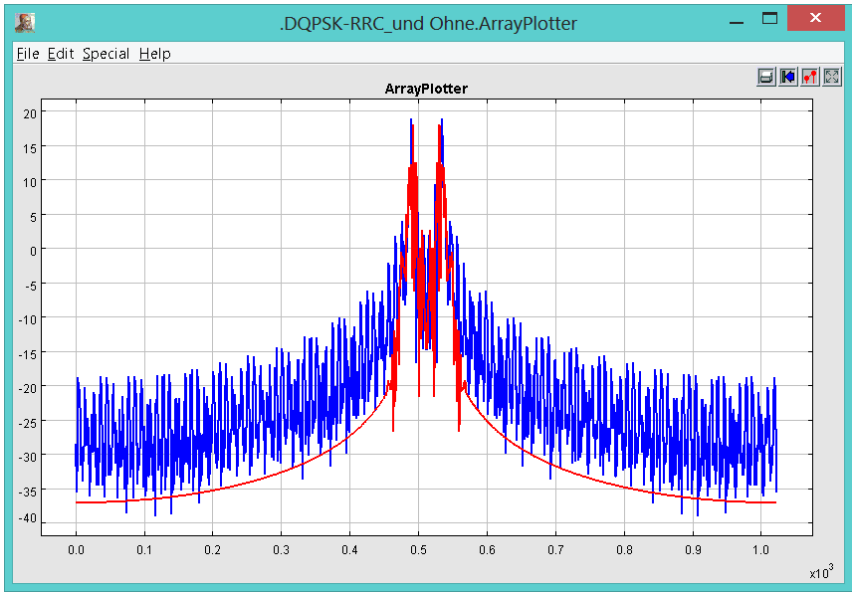

Abbildung 5.4: Spektrum des Sendesignals mit und ohne RRC-Filter

5.3 Demodulator

5.3.1 Kernfunktionen

Das Modell des Empfängers ist wie im Bild 5.5 dargestellt aufgebaut. Wie auch bei dem Sender ist die Frequenz der Lokaloszillatoren zwischen 504kHz und 494kHz variierbar. In Verbindung mit dem Sender können die grundlegenden Funktionen des Modems getestet werden.

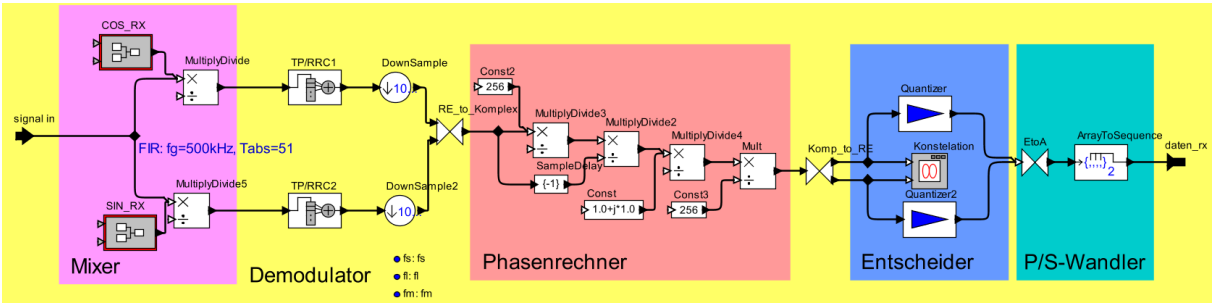

Abbildung 5.5: Aufbau des Empfängers in Ptolemy II

Als Erstes wird die Auswirkung der unterschiedlichen Oszillatorfrequenzen im Sender und Empfänger simuliert. Wie erwartet dreht sich zwar die Konstellation (Abb. 5.6), aber es

entstehen keine Fehler, solange die Differenz zwischen den Frequenzen kleiner als der maximal zulässige Wert $\Delta f_{MAX_{DQPSK}} = \frac{1}{8} fc = 62,5kH$ ist.

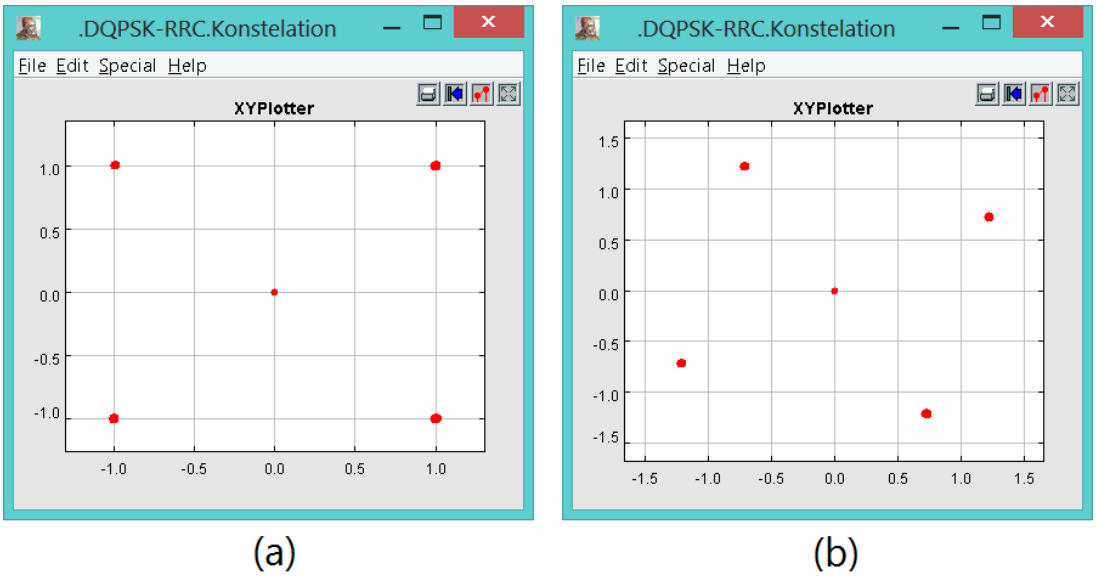

Abbildung 5.6: Konstellationsdiagramm (a) $\Delta f = 0Hz$ und (b) $\Delta f = 10kHz$

Anschließend wird das Sendesignal mit weißem Rauschen variabler Leistung überlagert. Damit simuliert man Datenübertragung über einen AWGN-Kanal (Additive White Gaussian Noise). Als Ergebnis der Simulation kann man festhalten, dass eine sichere Datenübertragung erst ab einem SNR von 20dB gewährleistet werden kann. Denn schaut man sich die entstehenden Konstellationsdiagramme (Abb. 5.7) an, so sieht man, dass schon bei 10dB das ankommende Signal zu stark verrauscht ist.

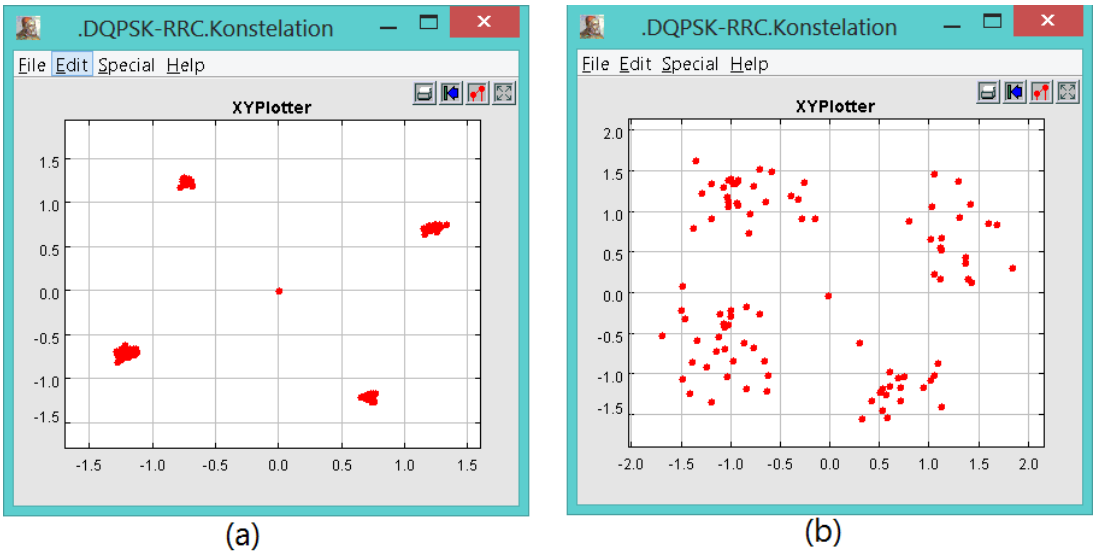

Abbildung 5.7: Konstellationsdiagramm bei einem SNR von 20dB (a) und 10dB (b)

Bedenkt man das in der Realität noch Quantisierungsrauschen dazukommt, so werden die Daten nicht mehr zu gebrauchen sein.
Zur Simulation von schmalbandigen Störungen, die z.B. durch die DC/DC Wandler verursacht werden können, kommt ein Sinusgenerator zum Einsatz. Da noch keine genauen Kennwerte über die möglichen Störungen vorliegen, kann deren Einfluss nur abgeschätzt werden. So wurde z.B. durch die Simulation festgestellt, dass schmalbandige Störungen bei 200kHz keinen signifikanten Einfluss auf die Übertragung haben solange das SNR größer als 6dB bleibt. Das heißt, dass bei einer Sendeleistung von 10W bzw. Sendespannung (U_{eff}) von 23,24V die Störspannung am Empfänger maximal $2,94\mu$V betragen darf. Diese Abschätzung erfolgt wie dargestellt:

Gegeben:
Sendeleistung $P_S = 10$W, Wellenwiderstand $Z_0 = 54\Omega$, Minimaler Abstand 6dB.

Daraus ergibt sich eine Sendeleistung von

$$P_{S_1}dB = 10 * \log(\frac{P_S}{1mW}) = 10 * \log(\frac{10W}{1mW}) = 40dBm \tag{5.1}$$

mit der Kabeldämpfung

$$a_{dB} = 33\frac{dB}{km} * 4km = 132dB \tag{5.2}$$

beträgt das Sendesignal am Empfänger

$$P_{S_2} = 40dBm - 132dBm = -92dBm \Leftrightarrow 630,95 * 10^{-15}W \tag{5.3}$$

$$U_{S_2} = \sqrt{P_{2_S} * Z_0} = \sqrt{630,95 * 10^{-15}W * 54\Omega} = 5,83\mu V \tag{5.4}$$

Daher darf die Störung nicht größer als

$$P_{Störung} = 40dBm - (132dB + 6dB) = -98dBm \Leftrightarrow 158,48 * 10^{-15}W \tag{5.5}$$

$$U_{Störung} = \sqrt{P_{Störung} * Z_0} = \sqrt{158,48 * 10^{-15}W * 54\Omega} = 2,94\mu V \tag{5.6}$$

werden.

5.3.2 Symboltaktrückgewinnung

Wie schon in den Kapiteln 4.2 und 4.3.2 angesprochen, funktioniert die Demodulation eines DQPSK Signals nur dann, wenn eine Symbolsynchronisation betrieben wird. Bei der eingesetzten S6Brd02 Platine weichen die Quarze bedingt durch die Fertigungstoleranzen um ±50ppm voneinander ab. Allein diese kleine Abweichung ist ausreichend, damit die Symboltaktsignale im Sender und Empfänger auseinander laufen (Abb. 5.8). Zusätzlich

dazu werden die beiden Platine bei unterschiedlichen Umgebungstemperaturen betrieben. Durch den vorhandenen Temperaturdrift weichen die Frequenzen der beiden Oszillatoren somit noch mehr voneinander ab.

Abbildung 5.8: Asynchrone Symboltaktsignale im Sender(blau) und Empfänger(grün)

In folgenden Unterkapiteln werden drei Verfahren zur Symbolsynchronisation vorgestellt und miteinander verglichen um das passende Verfahren auszuwählen.

Costas-Loop

Regulär verwendet man die Costas-Loop zur Trägerrückgewinnung aus dem Modulationsprodukt. Dadurch das bei der verwendeten DQPSK pro Periode genau ein Symbol übertragen wird (Kapitel 3.4), kann man mit einem Komparator aus der Trägerschwingung auch den Symboltakt rekonstruieren. Bild 5.9 zeigt den Aufbau der Costas-Loop für die (D)QPSK.

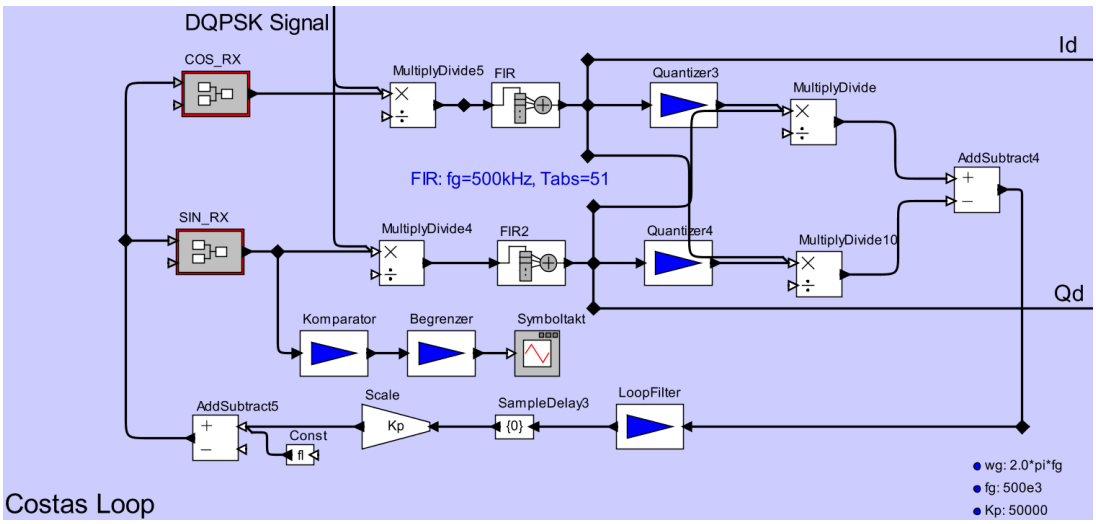

Abbildung 5.9: Aufbau der Costas-Loop in Ptolemy II

Das empfangene DQPSK Signal $I_D \cos(\omega_T t) - Q_D \sin(\omega_T t)$ wird zuerst mit $\cos(\omega_T * t + \varphi)$

und $-\sin(\omega_T * t + \varphi)$ des Lokaloszillators multipliziert. Nach den Tiefpässen erhält man:

$$S_I \sim I_D \cos(\varphi) + Q_D \sin(\varphi) \tag{5.7}$$

$$S_Q \sim Q_D \cos(\varphi) - I_D \sin(\varphi) \tag{5.8}$$

Als Nächstes werden in den Quantisierern die Vorzeichen der Signale gebildet.

$$S'_I = sgn\{S_I\} \tag{5.9}$$

$$S'_Q = sgn\{S_Q\} \tag{5.10}$$

Am Ausgang der Multiplizierer treten nun folgende Signale auf:

$$S''_I \sim S_I * sgn\{S_Q\} \sim (I_D \cos(\varphi) + Q_D \sin(\varphi)) * sgn\{S_Q\} \tag{5.11}$$

$$S''_Q \sim S_Q * sgn\{S_I\} \sim (Q_D \cos(\varphi) - I_D \sin(\varphi)) * sgn\{S_I\} \tag{5.12}$$

Nach der anschließenden Subtraktion und Tiefpassfilterung durch den Schleifenfilter erhält man ein Signal welches $\sim \varphi$ ist [4] [5].
Die Konfigurierung der Costas-Loop hat sich in der Simulation als sehr schwierig erwiesen. Da bei jeder Periode die Phase des Trägers geändert wird, rastet die PLL zwar ein aber die Regelgröße bleibt nicht konstant genug. Die Frequenz im Empfänger befindet sich immer in einem grenzstabilen Zustand. Das wiederum führt dazu, dass die demodulierten Signale stark jittern (Abb. 5.10).
Aufgrund dessen und dem vergleichsweise anspruchsvollen Aufbau der Costas-Loop wird von der Symboltaktrückgewinnung durch die Costas-Loop abgesehen.

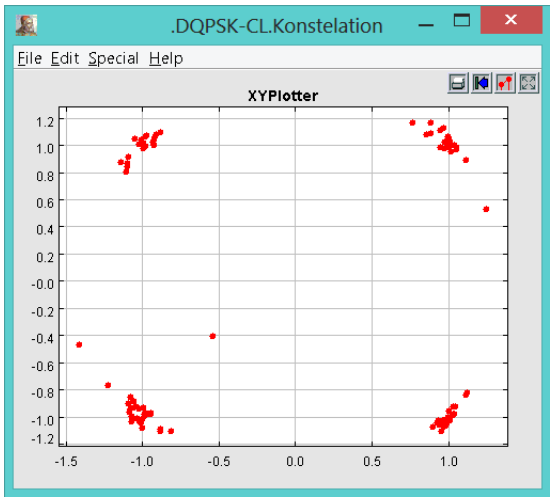

Abbildung 5.10: Konstellationsdiagramm bei der Symboltaktregelung mit Costas-Loop

Zweifache Quadrierschleife

Wie auch die Costas-Loop wird die Quadrierschleife dazu verwendet, die Trägerschwingung zurückzugewinnen. Mit der man anschließend einen Komparator ansteuert, um den Symboltakt zu rekonstruieren. Das Blockschaltbild der Quadrierschleife ist in Bild 5.11 dargestellt.

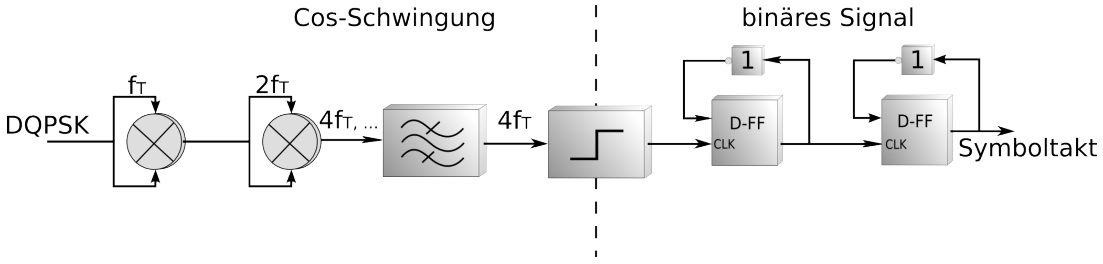

Abbildung 5.11: Blockschaltbild einer zweifachen Quadrierschleife zur Symboltaktrückgewinnung

Zuerst wird das DQPSK Signal $S_{DQPSK}(t) = \cos(\omega_T t + \frac{\pi}{2}(i-1))$ zwei Mal quadriert. Das Ergebnis beinhaltet unter anderem eine harmonische Schwingung der Frequenz $4f_T$.[4]

$$S_{DQPSK}(t)^4 = \frac{1}{8}(3 + 4\cos(2\omega t + \pi(i-1)) + \cos(4\omega t + 2\pi(i-1)) \tag{5.13}$$

Diese wird anschließend über den Bandpass ausgesiebt. Über den darauf folgenden Komparator wird ein binäres Signal mit dieser Frequenz erzeugt. Zum Schluss wird die Frequenz, mithilfe der beiden D-Flipflops, durch vier geteilt. Wie man es aus folgendem Zeitdiagramm entnehmen kann, hat diese Art der Symboltaktrückgewinnung zumindest in der VDHL Simulation mit ISim funktioniert.

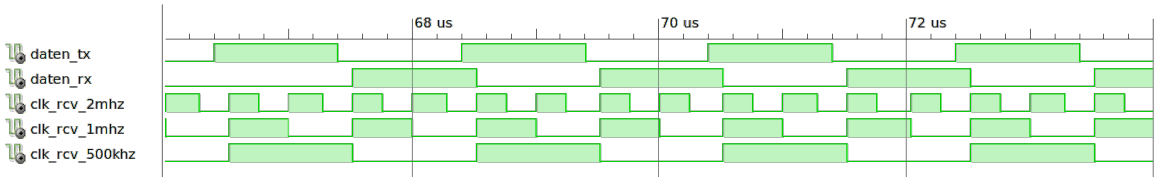

Abbildung 5.12: Zeitdiagramm Symboltaktrückgewinnung mit der zweifachen Quadrierschleife

Leider ist für die Aussiebung der Freeuenz $4f_T$ ein sehr schmalbandiger Bandpass bei 2MHz erforderlich. Dieser benötigt in der FIR Ausführung mindestens 100 DSP-Slices, die bei dem verwendeten FPGA nicht zur Verfügung stehen. Vermutlich würde dafür auch ein IIR Bandpass 4. Ordnung ausreichen. Aus zeitlichen Gründen wird diese Lösung aber nicht weiter verfolgt.

Early-Late-Gate-Loop

Die Early-Late-Gate-Loop stellt eine weitere Möglichkeit der Symbolsynchronisation dar. Dabei erfolgt die Symboltaktrückgewinnung aus dem demodulierten Signal. Der prinzipielle Aufbau der Schleife ist in der Abbildung 5.13 dargestellt.

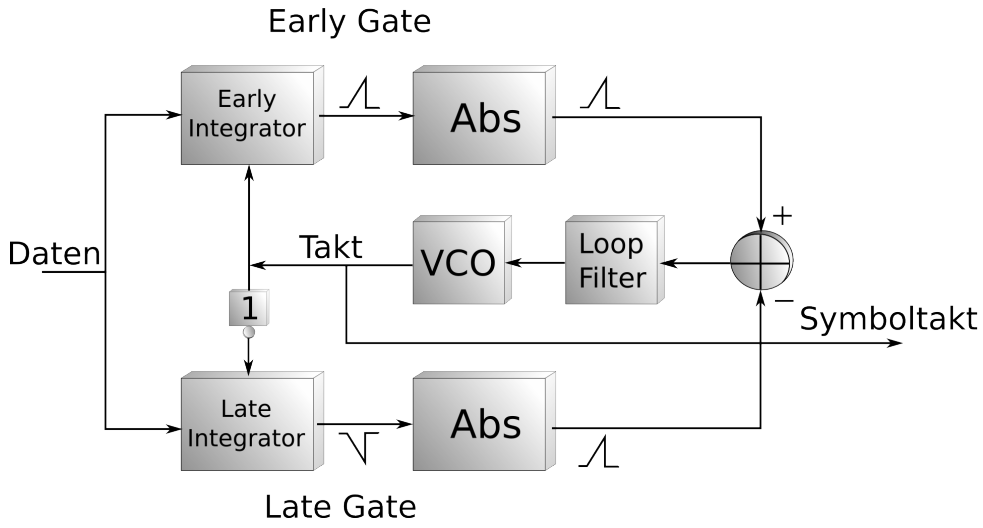

Abbildung 5.13: Blockschaltbild Early-Late-Gate-Loop

Ein demoduliertes binäres Signal, das man z.B. nach dem RRC-Filter abgreifen kann, bildet die Eingangsgröße für Early-Late-Schleife. Der Takt und der negierte Takt steuern die beiden Integrierer so an, dass das bipolare Datensignal in den positiven Halbperioden integriert wird. Die Ergebnisse der Integration werden anschließend als Beträge voneinander subtrahiert. Die Differenz der beiden Größen wird anschließend über einen Schleifenfilter dem VCO zugeführt.
Ist der lokale Symboltakt korrekt (Fall 1. und 2. in der Abb. 5.14), so stimmen die Beträge der beiden Integrationsergebnisse überein und das Fehlersignal e_d ist gleich null. In dem 3'ten. Fall ist der Symboltakt nicht mehr synchron mit den Daten. Das hat zufolge, dass das Ergebnis der Integration im Early-Zweig kleiner ist als der im Late-Zweig. Dadurch entsteht ein Fehlersignal e_d, das kleiner Null ist. Dieses steuert den VCO über den Schleifenfilter so an, dass der Takt in die richtige Lage gebracht wird. Im 4'ten. Fall passiert genau das Gegenteil und der Takt wird somit in die andere Richtung verschoben [6] [7] [8].

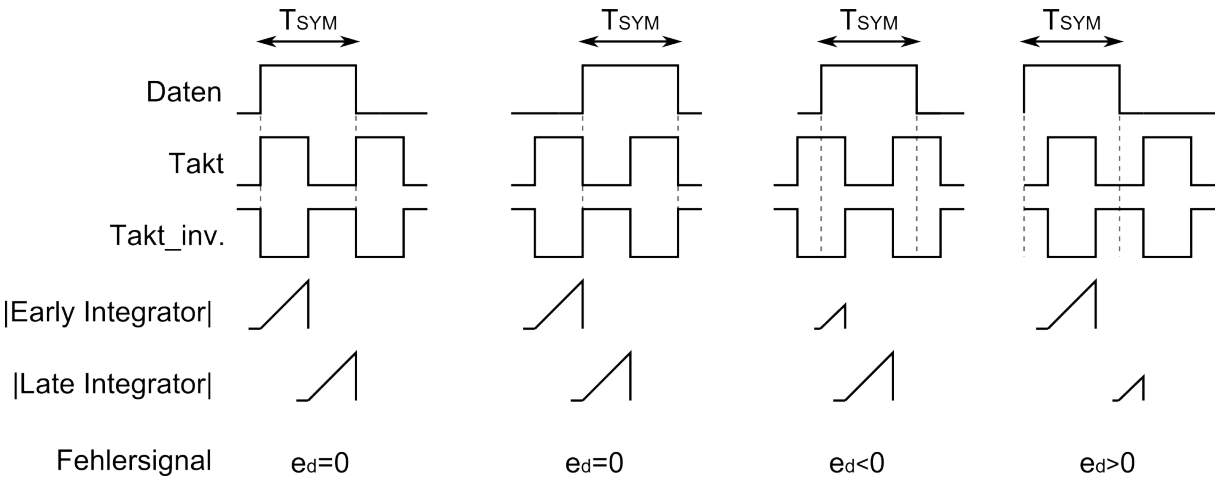

Abbildung 5.14: Signalformen bei der Early-Late-Synchronisation

Zum Prüfen der Funktionsweise wird eine Early-Late-Gate-Schleife implementiert, die zwar auf demselben Prinzip beruht aber etwas anders aufgebaut ist. Die genaue Erklärung der Umsetzung erfolgt im Kapitel 6.2.7.

Dadurch, dass bei dem Early-Late-Gate-Verfahren die Taktsynchronisierung unabhängig von der Trägerrückgewinnung erfolgt, und dieses Verfahren auch dann eingesetzt werden kann, wenn das Symbol länger als eine Periode der Trägerschwingung dauert, wird in dem Modem dieses Verfahren zu Taktsynchronisation eingesetzt.

6 Implementierung des DQPSK-Modems in VHDL

Auf Basis der Überlegungen aus den vorhergehenden Kapiteln, soll nun die Realisierung der einzelnen Elemente des Modems und deren Funktionalitäten näher beschrieben werden. Den schematischen Aufbau des Gesamtsystems zeigt die Abbildung 6.1. Die Root-Raised-Cosine-Filter werden zwar erstellt aber nicht in das Design eingebunden (siehe Kapitel 6.1.6). Bezogen auf die Fließkommasimulationen aus dem Kapitel 5 bzw. Abbildung 5.1 müssen hier die internen Signale in dem Maße quantisiert werden, dass die Modulation bzw. Demodulation hinreichend funktioniert.

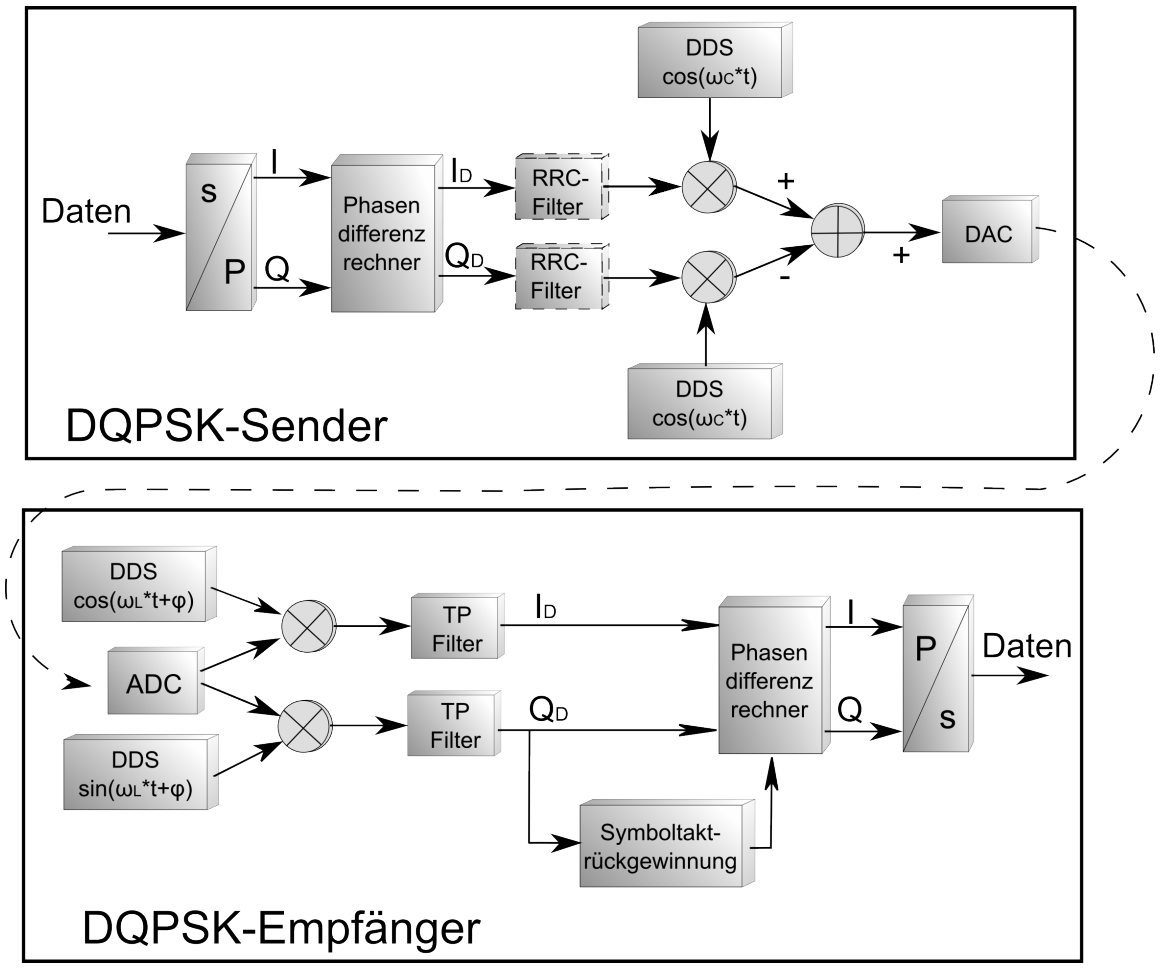

Abbildung 6.1: Blockschaltbild des Gesamtsystems

6.1 Modulator

6.1.1 Taktmanagement

In der Komponente Modulator kommen drei unterschiedliche Taktsignale zum Einsatz. Der 25MHz Takt steuert den Digital Analog Wandler und das DDS (Direct Digital Synthesis) an. Die Wandlung der Daten von einem seriellen Bitstrom auf zwei parallele erfolgt mit 1MHz. Der Phasendifferenzrechner wird mit dem Symboltakt von 500kHz angesteuert. In den folgenden Unterkapiteln wird erklärt, wie die Realisierung der einzelnen Taktsignale erfolgt.

Digital Clock Manager und 25MHz Takt

Auf der verwendeten S6Brd02-Platine ist ein Quarzoszillator mit 27MHz vorhanden. Zur Erzeugung der Taktsignale, die zwischen 3,145MHz und 280MHz liegen, sollte der DCM (Digital Clock Manager) Core der Firma Xilinx®verwendet werden. Dadurch wird sichergestellt, dass die generierten Taktsignale über spezielle FPGA-Taktleitungen verlegt werden und somit geringen Jitter und Latenzen aufweisen. Mit dem DCM wird zuerst ein Takt mit Frequenz von 50MHz erzeugt. Daraus kann man wiederum die 25MHz, 1MHz und 500kHz Takte ableiten. Die 25MHz werden einfach über die Frequenzteilung mit einem D-Flipflop gewonnen. Die Generierung des DCM-Cores erfolgt mit einem Coregenerator-Assistenprogramm, in dem man das gewünschte Verhalten einstellt und die Ein- und Ausgabeparameter definiert. Es werden folgende Einstellungen vorgenommen:

- Frequenzsynthese
  - Dadurch kann über eine integrierte DPLL ein beliebiger Ausgangstakt eingestellt werden, solange dieser im zulässigen Bereich liegt.
- Minimize output jitter
- Input/Output Clock: 27MHz und 50MHz
- Inputs/Outputs : Reset und Locked
  - Der „Locked“ Ausgang wird für die weiteren Elemente als Chip Enable Signal verwendet.

1MHz und 500kHz Taktgenerator

Das 1MHz Signaltakt muss mit einem Synchronzähler generiert werden. Wie man dem unten dargestellten Quellcode entnehmen kann, reagiert der Zähler auf steigende Flanke des 50MHz Taktes. Sobald der Zähler 24 erreicht hat, erfolgt eine Zustandsänderung an dem 1MHz Ausgang und erzeugt somit einen symmetrischen Takt.

```
1  process(INPUT_50MHz,CE)
2  begin
3      if CE = '0' then
4          output_temp <= '0';
5          cnt <= "00000";
6      else
7          if rising_edge(INPUT_50MHz) then
8              cnt <= cnt + 1;
9              if cnt >= pn_clk_teiler-1 then
10                 output_temp <= not output_temp;
11                 cnt <= "00000";
12             end if;
13         end if;
14     end if;
15 end process;
16 OUTPUT_1MHz <= output_temp;
```

Code 6.1: Prozess zur Generierung 1MHz Taktes

Die 500kHz werden genauso wie auch die 25MHz einfach über die Frequenzteilung mit einem D-Flipflop erzeugt.

6.1.2 PN- bzw. Test-Sequenz Generator

Um zu testen, ob eine Übertragungsstrecke auch tatsächlich funktioniert, sind nahezu zufällige Daten notwendig. Diese werden mittels PN-Sequenzen realisiert. Eine PN-Sequenz ist eine deterministische Sequenz, die aber für den Betrachter als Folge von zufällig verteilten Einsen und Nullen erscheint.
Die einfachste Art solche Pseudo-Zufallsfolgen zu erzeugen, besteht in der Verwendung von rückgekoppelten Schieberegistern. Die Rückkopplung erfolgt von mehreren Schieberegisterstufen, die modulo-2-addiert (xor) auf den Eingang zurückgeführt werden. Es müssen Minimum zwei Stufen rückgekoppelt werden, wobei der Ausgang des Schieberegisters grundsätzlich angeschlossen sein muss [10].
Der Implementierte PN-Sequenz Generator besteht aus sieben Stufen die wie in der Abbildung 6.2 dargestellt rückgekoppelt sind. Die Rückkopplung wurde dabei so gewählt, dass die Länge der Sequenz auch gleichzeitig die maximal mögliche Länge $L_m = 2^m - 1$ ist. Diese beträgt 127 Bits.

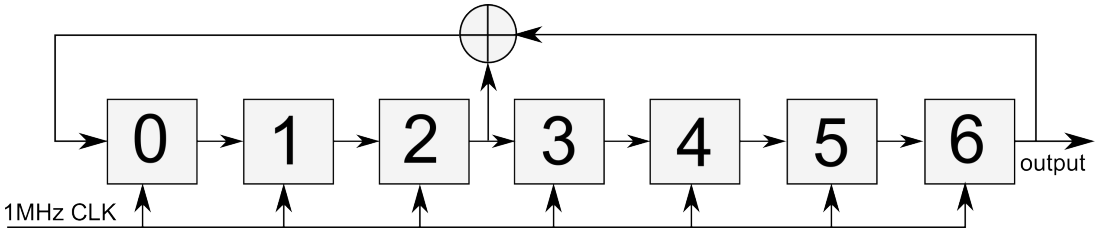

Abbildung 6.2: Blockschaltbild des PN-Sequenz Generator

Während der Implementierungsphase ist es dagegen vom Vorteil, eine bekannte Sequenz schicken zu können. Das bietet einem die Möglichkeit, die Signalverläufe an den bestimmten Punkten prüfen zu können, um festzustellen, wo ein Fehler auftritt. Aus diesem Grund wird ein Schieberegister mit variabler Länge erzeugt. Dieses kann dann mit einer beliebigen Sequenz geladen werden, die sich dann am Ausgang wiederholt.

6.1.3 Seriell/Parallel-Wandler

Die Seriell/Parallel-Wandlung erfolgt über einen Demultiplexer mit zwei Ausgängen. Die Abbildung 6.3 (a) zeigt wie dieser aufgebaut ist.

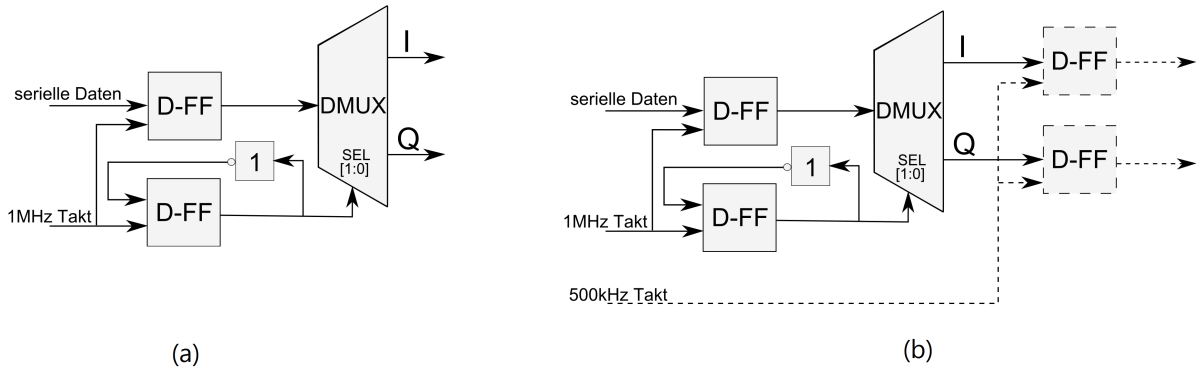

Abbildung 6.3: Demultiplexer

Die ankommenden seriellen Daten werden vor der Parallelisierung erst mal in einem Flipflop zwischengespeichert. Damit wird sichergestellt, dass die Zuweisung der Daten an einem Ausgang nur dann erfolgt, wenn die Daten am Eingang stabil sind. Die Auswahl der zugehörigen Ausgangsleitung erfolgt über ein Kippglied. Da die Daten über dem Seriell/Parallel-Wandler von der 1-MHz-Taktebene in die 500-kHz-Taktebene übergehen, wurden bei dem ersten Entwurf der Schaltung (Abb. 6.3 (b)) die Daten am Ausgang noch mal zwischengespeichert. Somit sollte gewährleistet werden, dass die Daten am Phasenrechner sich nur mit der Symbolfrequenz ändern. Bei der Simulation der Schaltung mit ISim hat diese Vorgehensweise auch einwandfrei funktioniert.(Abb. 6.4)

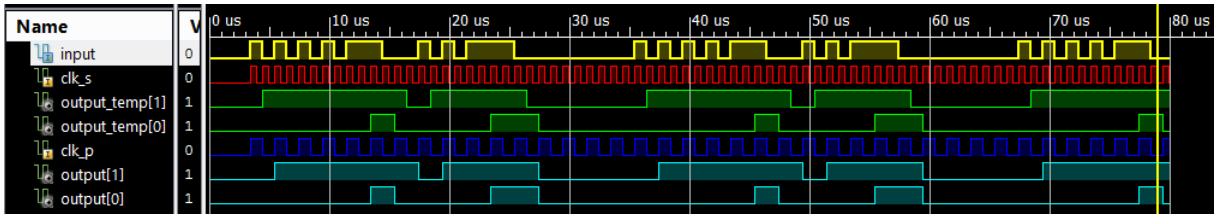

Abbildung 6.4: Demultiplexer Zeitdiagramm in ISim

Wurde die Schaltung aber anschließend synthetisiert und auf dem FPGA gemapt, so erfolgte die Parallelisierung der Daten nicht immer fehlerfrei. Nach einer ausgiebigen Fehlersuche wurde festgestellt das man die Daten in einem FPGA zwischen zwei Taktebenen nur mit FIFO(First-InFirst-Out) sicher übermitteln kann [11]. Durch den Einbau einer FIFO Komponente zwischen dem Seriell/Parallel-Wandler und dem Phasendifferenzrechner ist es nicht mehr nötig, die Daten am Ausgang des Wandlers zwischenzuspeichern. Die beiden Flipflops am Ausgang können somit entfernt werden.

6.1.4 FIFO

Ein FIFO ist ein Pufferspeicher, der einen Übergang von schnellen Übertragungspfaden auf langsamere ermöglicht. Hierfür werden die Eingangsdaten in einem als Ringpufferorganisiertem Dualport-RAM abgelegt. Anschließend können die Daten mit einer geringeren Taktfrequenz ausgelesen werden. Andererseits können die Daten natürlich auch mit einem langsamen Takt abgespeichert werden, um dann mit einer höheren Datenrate ausgelesen zu werden.
Das eingesetzte FIFO erfüllt zwei Aufgaben. Zum einem wird der Übergang der Daten von einer Taktebene in die andere Taktebene sichergestellt und zum anderen werden die parallelen Daten nach dem FIFO auch wirklich synchron ausgegeben. Denn nur so kann man gewährleisten, dass die Bildung der differenziellen Signale in dem Phasendifferenzrechner richtig erfolgt.
Die FIFO Komponente wird als IP-Core von Xilinx™ durch den Core-Generator zur Verfügung gestellt. Es werden folgende Einstellungen vorgenommen:

- Unabhängige Taktsignale für den Ein- und Ausgang des FIFOs
- Eingangs-/Ausgangsbusbreite: je 2 Bit
- Inputs/Outputs: Valid Pin
  - Dieser wird „high“ wenn am Ausgang des FIFOs gültige Daten anliegen und steuert bei den nachfolgenden Elementen den Chip Enable Eingang.

6.1.5 Phasendifferenzrechner

Wie schon in Kapitel 4.2 vorgestellt, werden in dem Phasendifferenzrechner die I- und Q-Signale in differenzcodierte Signale I_D und Q_D umgewandelt. Um die Ausgangswerte in dem Bereich von ± 1 bzw. $\pm j$ festzuhalten, ist eine komplexe Multiplikation mit der Konstante $(0,5 - j0,5)$ notwendig. Da es aber in einem FPGA ohne eine Floating Point Unit nicht möglich ist, Ganze- und Kommazahlen miteinander zu multiplizieren, gilt es einen alternativen und vereinfachten Weg zu finden.
Als Erstes soll mit der Konstante nicht das Ergebnis des Rückführungszweiges multipliziert werden, sondern der komplexe Wert der beiden Eingangssignale I und Q. Beachtet man dabei, dass das komplexe Eingassignal das aus I und Q gebildet wird, nur vier folgende Zustände annehmen kann:

$$s(t) = I(t) + jQ(t)$$

$$\Leftrightarrow -1 - j$$

$$\Leftrightarrow -1 + j$$

$$\Leftrightarrow 1 - j$$

$$\Leftrightarrow 1 + j$$

so kann die Multiplikation mit $(0,5 - j0,5)$ durch eine einfache Zuweisung ersetzt werden.

$$(-1 - j) * (0,5 - j0,5) := -1 + j0$$

$$(-1 + j) * (0,5 - j0,5) := 0 + j$$

$$(1 - j) * (0,5 - j0,5) := 0 - j$$

$$(1 + j) * (0,5 - j0,5) := 1 + j0$$

Der nächste Schritt ist die Implementierung eines komplexen Multiplizierers. Eine Multiplikation bei der zwei komplexe Operatoren $a = a_r + ja_i$ und $b = b_r + jb_i$ miteinander multipliziert werden, kann auch durch vier reelle Multiplikationen erfolgen:

$$\underline{p} = (a_r + ja_i) * (b_r + jb_i) \tag{6.1}$$

$$\underline{p} = a_r b_r + ja_r b_i + ja_i b_r - a_i b_i \tag{6.2}$$

$$\mathrm{Re}(\underline{p}) = a_r b_r - a_i b_i \tag{6.3}$$

$$\mathrm{Im}(\underline{p}) = a_r b_i + a_i b_r \tag{6.4}$$

Um die Multiplikationen in VHDL ausführen zu können benötigt man die Bibliothek „*ieee.numeric_std*“. Die Implementierung des Multiplizierers erfolgt dann wie beschrieben.

Der zugehörige Quellcodeausschnitt ist folgend dargestellt:

```
1        process(clk)
2        begin
3                if rising_edge(clk) then
4                        re_im_int <= re_im;
5                        b_re <= p_re_int( 1 downto 0);
6                        b_im <= p_im_int( 1 downto 0);
7                end if;
8        end process;
9
10       process(re_im_int)
11       begin
12               if re_im_int(1)='1' then
13                       case re_im_int(0) is
14                               when '1' => erg_05_re <="01";
15                                           erg_05_im <= "00";
16                               when others => erg_05_re <="00";
17                                              erg_05_im <= "11";
18                       end case;
19               else
20                       case re_im_int(0) is
21                               when '1' => erg_05_re <="00";
22                                           erg_05_im <= "01";
23                               when others => erg_05_re <="11";
24                                              erg_05_im <= "00";
25                       end case;
26               end if;
27       end process;
28
29   p_re_int <= erg_05_re*b_re-erg_05_im*b_im;
30   p_im_int <= erg_05_re*b_im+erg_05_im*b_re;
```

Code 6.2: Phasendifferenzrechner

Die Ergebnisse der Differenzcodierung liegen anschließend im 2er-Komplement Format vor.

6.1.6 Root-Raised-Cosine-Filter

Die Root-Raised-Cosine-Filter werden über den FIR Compiler v5.0 als ein IP-Cores eingebunden. Damit die Bandbegrenzung ordnungsgemäß funktioniert, werden diese als FIR Filter 101. Ordnung implementiert. Die hohe Filterordnung ist deswegen notwendig, weil bei einer Abtastfrequenz von $f_s = 25MHz$ ein relativ kleiner Durchlassbereich $f_c = 500kHz$ realisiert werden soll. Und solche steilen Flanken sind bei den FIR Filtern nur

mit hoher Ordnung realisierbar. Die Berechnung der dafür notwendigen Filterkoeffizienten erfolgt in Matlab™. Die Koeffizienten liegen zunächst in dem Fließkommazahlenformat vor, dann werden diese in Festkommazahlen umgewandelt und anschließend wird ein „Xilinx Coefficient-File“ erstellt. Die implementierten RRC-Filter haben folgende Parameter:

- Abtastfrequenz: $f_s = 25MHz$
- Cut-Off-Frequenz: $f_c = 500kHz$
- Filterordnung: 101
- Roll-Off-Faktor: 0.22

Im Bild 6.5 ist der dazugehöriger Frequenzgang der RRC-Filer dargestellt.

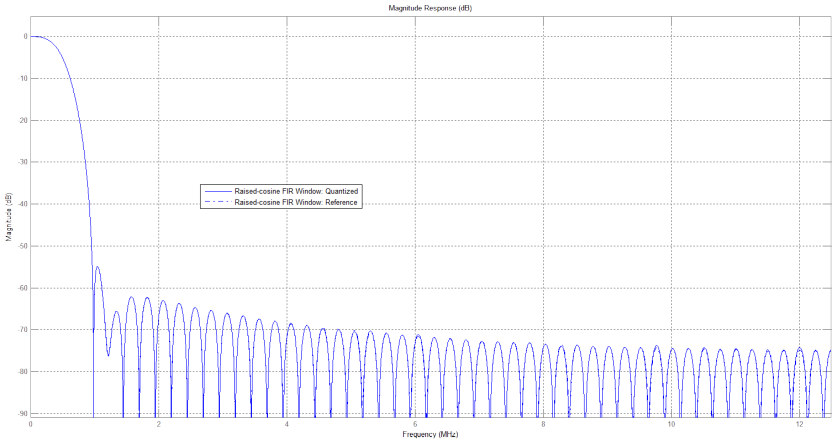

Abbildung 6.5: Frequenzgang Root-Raised-Cosine-Filter

Der verwendete FPGA hat insgesamt 132 DPS-Slices, das ist aber leider nicht ausreichend, um die vier notwendigen RRC-Filter in dem Modem zu implementieren. Aus diesem Grund wurden die RRC-Filter momentan aus dem Design entfernt. Eventuell ist es möglich, die notwendige Ordnung der RRC-Filter durch einen Dezimierungsvorgang oder durch Heruntersetzen der Abtastfrequenz zu reduzieren. Diese Möglichkeiten wurden aus zeitlichen Gründen jedoch nicht genauer untersucht, stellen aber für die Zukunft mögliche Lösungswege dar.

6.1.7 Direkter Digitaler Synthesizer

Die Direkte Digitale Synthese (DDS) ist ein Verfahren zur Erzeugung sinusförmiger Signale mittels digitaler Logik. Die erzeugten $\cos(\omega_T t)$ bzw. $-\sin(\omega_T t)$ Signale werden anschließend bei dem Modulationsvorgang mit I_D und Q_D multipliziert.
Als IP-Core verfügbarer DDS (Abb. 6.6) besteht im Grunde aus einem Phasenakkumulator mit variabler Wortbreite (zwischen 3 und 48bit) und einer Lookup-Tabelle, in der eine

volle Periode Sinusfunktionswerte abgelegt ist. Der Phasenakkumulator an sich besteht aus einem Addierer und einem Register, zu dem bei jedem Abtastpunkt T_S ein vorgegebener Wert des Phaseninkrementes addiert wird. Der Wert, der sich zwischenzeitig in dem Register befindet, dient seinerseits als Zieladresse für die Lookup-Tabelle. Am Ausgang der Tabelle steht wiederum der zugehörige diskrete Amplitudenwert der Cosinus- bzw. Sinusfunktion [12].

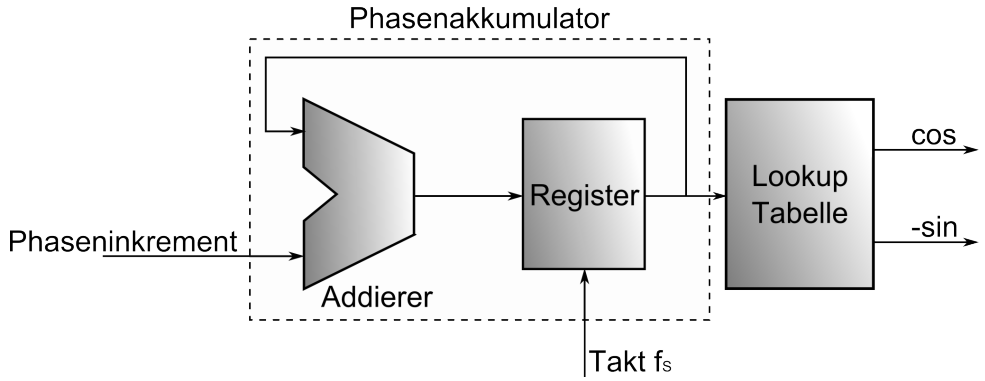

Abbildung 6.6: Blockschaltbild eines DDS

Für die Frequenz des Ausgangssignals gilt:

$$f_{out} = 500kHz = \frac{f_s \Delta\Theta}{2^{B_{\Theta(n)}}} \tag{6.5}$$

dabei ist $\Delta\Theta$ der Phaseninkrement und $B_{\Theta(n)}$ die Wortbreite des Phasenakkumulator in Bit. Um die Quantisierungsfehler so gering wie möglich zu halten, beträgt die Wortbreite 36 Bit. Somit ergibt sich für den Phaseninkrement ein Wert von 1374389535.

$$\Delta\Theta = \frac{f_{out} * 2^{B_{\Theta(n)}}}{f_s} \tag{6.6}$$

$$\Delta\Theta = \frac{500kH * 2^{36bit}}{25MHz} = 1374389535 \tag{6.7}$$

Die Frequenzauflösung bzw. Frequenzabweichung Δf beträgt dabei:

$$\Delta f = \frac{f_s}{2^{B_{\Theta(n)}}} \tag{6.8}$$

$$\Delta f = \frac{25MHz}{2^{36Bit}} = 363,80\mu Hz \tag{6.9}$$

6.1.8 Mixer und Addierer

Mixer und Addierer sind die letzten Komponenten in dem Sender vor der Digital-Analog-Wandlung. Eigentlich ist ein Mixer ein Multiplizierer, welches den Träger mit dem modu-

lierenden Signal multipliziert. Schaut man sich aber die zustande kommenden differenziellen Signale I_D und Q_D an (Abb.6.7), so sieht man das diese nur drei Zustände $(1, 0, -1)$ annehmen können.

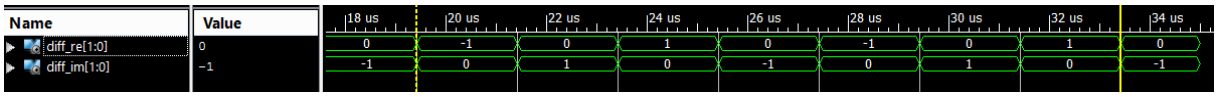

Abbildung 6.7: Zeitdiagramm I_D und Q_D ohne Impulsformungsfilter

Somit kann die Multiplikation zwischen der jeweiligen Trägerschwingung und I_D bzw. Q_D durch einfache Vorgänge wie Zahlennegation und Zuweisung ersetzt werden. Ist das zu modulierende Signal I_D oder $Q_D = 0$, so liegt am Addieren eine Periode ($T = 2\mu s$) lang eine Null an. Entspricht der Wert einer „1“, wird die Trägerwelle unverändert an den Addierer weitergeleitet, ist der Wert aber „-1“, so erfolgt die Invertierung der Trägerwelle. Dieses Vorgehen ist aber nur deswegen möglich, weil zur Zeit noch keine Impulsformungsfilter eingesetzt werden. Da die Planung des Systems auch Verwendung von Impulsformungsfiltern vorsieht (Kapitel 5.2), werden beide Arten des Mixers implementiert (Code. 6.3), wobei einer immer deaktiviert sein muss.

```
1                   -- Mixer --
2  -- Lösung über die Zuweisung --
3          process(diff_re)
4          begin
5                  case diff_re is
6                          when "00" => cos_mal_diff_re <= (others => '0');
7                          when "01" => cos_mal_diff_re <= COS;
8                          when "01" => cos_mal_diff_re <= ((not COS) + 1);
9          end process;
10
11         process(diff_im)
12         begin
13                 case diff_im is
14                         when "00" => sin_mal_diff_im <= (others => '0');
15                         when "01" => sin_mal_diff_im <= SIN;
16                         when "01" => sin_mal_diff_im <= ((not SIN) + 1);
17         end process;
18 -- Lösung über Multiplikation --
19 cos_mal_diff_re <= signed(COS)*diff_re;       --16bit*14bit=> Result 16Bit
20 sin_mal_diff_im <= signed(SIN)*diff_im;       --16bit*14Bit=> Result 16Bit
21                         -- Addierer --
22 cos_plus_sin <= cos_mal_diff_re(15 downto 0) + sin_mal_diff_im(15 downto 0);
```

Code 6.3: Realisierung des Mixer (a) durch Zuweisung und (b) durch Multiplikation

Anschließend wird in dem Addierer die Summe der beiden Signale „cos_mal_diff_re“ und „sin_mal_diff_im“ gebildet. Diese stellt im Endeffekt die informationstragende Schwingung dar, die über das Koaxialkabel übertragen wird.

6.1.9 IP-Core zu Steuerung des Digital-Analog-Wandlers

„DAC-Cotrol“ stellt das letzte Element in dem Modul „Modulator“ dar. Dieser wandelt die zu sendenden Daten, im 2er Komplement Format so um, dass diese über den DAC (Digital-Analog-Wandler) korrekt ausgegeben werden.
Der verwendende DA-Wandler LTC1668 hat einen 16 Bit breiten Datenbus. Die Wandlung der Daten erfolgt mit der Abtastfrequenz $f_s = 25\text{MHz}$. Als Ausgangsgröße steht ein differenzieller Strom zur Verfügung der über zwei $52,3\Omega$ Widerstände eine differentielle Ausgangsspannung von $\pm 1\text{V}$ erzeugt [16].
Der differenzieller Ausgang liefert „$+1\text{V}$“ wenn die Eingangsdaten „$0x0000$“ betragen. Liegt am Eingang hingegen „$0xFFFF$“ so beträgt die Ausgangspannung „-1V“. Es ergibt sich also ein Zusammenhang wie in der Abb. 6.8 (b) dargestellt. Zum verschieben der vorhandenen Daten (a) in den neuen Zahlenbereich, müssen diese über XOR mit dem Bitvektor „0111111111111111“ verknüpft werden. Dadurch wird z.B. aus „$0x3FFF$“ „$0x4000$“ und die Ausgangsspannung nimmt die gewünschten $+0,5\text{V}$ an.

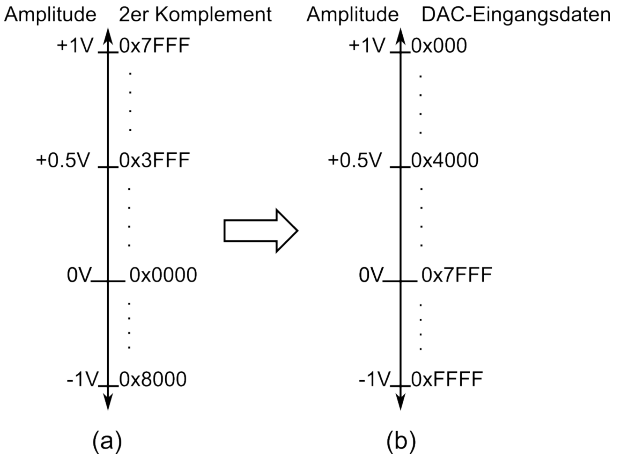

Abbildung 6.8: Zahlenbereich der Daten in 2er Komplement und der Eingangsbereich des AD-Wandlers

6.2 Demodulator

6.2.1 Taktmanagement und DDS

Wie auch bei dem Modulator-Modul bildet der Taktmanagement-Core die Grundlage für alle Elemente in dem Demodulator-Modul. Die Generierung der 50MHz bzw. 25 MHz

Taktsignale erfolgt genauso wie in dem Modulator, deswegen wird in diesem Kapitel darauf nicht weiter eingegangen. Der 1MHz Takt wird auch über einen Zähler erzeugt, aber diesmal muss dieses Taktsignal in Grenzen veränderbar sein, um den genauen Symboltakt einstellen zu können. Die genaue Beschreibung der Symboltaktrückgewinnung erfolgt im Kapitel 6.2.7. Anschließend leitet man von dem gewonnen 1MHz Taktsignal die 500kHz durch ein D-Flipflop ab.
Auch der zum Heruntermischen des DQPSK-Signals eingesetzter direkter digitaler Synthesizer wurde aus dem Modulator-Modul übernommen. (siehe Kapitel 6.1.7)

6.2.2 Analog-Digital-Wandler

Die Abtastung des DQPSK-Signals erfolgt mit dem AD-Wandler LTC2249 der Firma Linear Technology™. Dieser hat einen differenziellen, analogen Eingang, um den Einfluss der Gleichtaktstörungen so gering wie möglich zu halten [17]. Die verwendete Abtastfrequenz beträgt 25MHz. Das Ergebnis der Wandlung wird über einen 14 Bit breiten parallelen Datenbus ausgegeben. Die digitalisierten Werte liegen im 2er-Kompliment vor.

6.2.3 Mixer

Das digitalisierte DQPSK-Signal wird zuerst den beiden Mixern zugeführt. Hier erfolgt die Multiplikation des Eingangssignals mit den lokal erzeugten $-\sin(\omega_T t + \varphi)$ bzw. $\cos(\omega_T t + \varphi)$ Signalen um daraus I_D und Q_D zu bekommen. Da jeder Faktor in der Multiplikation 14 Bit breit ist, ist das Produkt insgesamt 28 Bit breit. Da solche Genauigkeit für die weiteren Berechnungen nicht notwendig ist, werden von dem Ergebnis nur die obersten 16 Bit weiter verwendet.

6.2.4 Tiefpassfilter

Wie schon in dem theoretischen Kapitel 4.3.1 beschrieben, enthält das Ergebnis der Multiplikation auch Signalanteile der doppelten Frequenz. Da man sich aber nur für die Gleichanteile I_D und Q_D interessiert, ist es wichtig, den Rest zu eliminieren. Der dazu notwendiger Tiefpass wird mit dem FIR Compiler erstellt und anschließend in das Design eingebunden. Die Berechnung der Filterkoeffizienten erfolgt wie auch bei den RRC-Filtern mit Matlab™. Um einen optimalen TP-Filter zu entwerfen, wurden unterschiedliche Filterentwürfe in dem Programm Ptolemy II simuliert. Dabei hat sich herausgestellt, dass ein Filter mit 51 Tabs und einer Grenzfrequenz von 50kHz nahezu optimal ist. Dieser hat bei der doppelten Signalfrequenz (1MHz) ca. 40dB Dämpfung und ist mit 51 Tabs ohne Probleme im verwendeten FPGA implementierbar. Die eingestellte Grenzfrequenz liegt unter der Frequenz des Basisbandsignals (500kHz). Bedingt durch die hohe Abtastfrequenz und geringe Filterordnung hat dieser bei 500kHz aber nur eine geringfügige Dämpfung von ca.

5dB. Dadurch funktioniert die Filterung des BB-Signals ohne Probleme. Folgend sind die Filterdaten zusammengefasst:

- Abtastrate f_s: 25MHz
- Grenzfrequenz f_g: 50kHz
- Filterordnung: 50
- Fensterfunktion: Taylor

In der Abbildung 6.9 ist der Frequenzgang des Filters dargestellt.

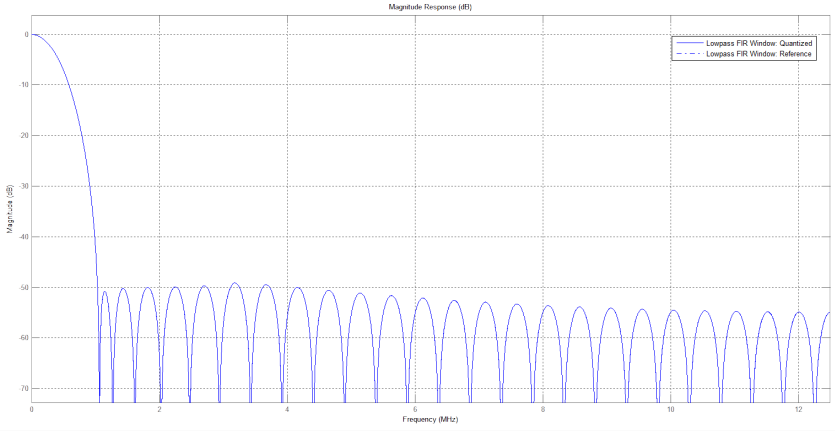

Abbildung 6.9: Frequenzgang des Tiefpassfilters im Demodulator

Da die Impulsantwort des entworfenen Filters zum Mittelpunkt achsensymmetrisch ist (Typ 1 FIR-Filter), sind bei der Implementierung nicht 50 DSP-Slices pro Filter notwendig, sondern nur 25. Diese Einsparung lässt sich auf das Ausnutzen der Achsensymmetrie zurückführen. Dadurch werden die aufwendigen Multiplikationsvorgänge um die Hälfte reduzieren. Dabei werden die Daten aus den jeweiligen Speicherelementen zunächst addiert und anschließend mit identischen Filterkoeffizienten multipliziert (Abb. 6.10) [13].

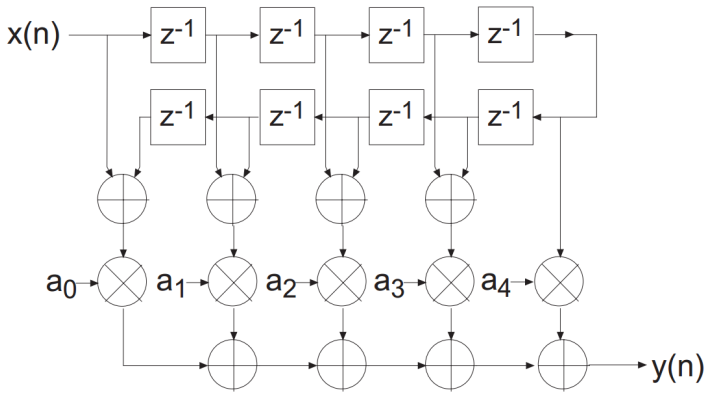

Abbildung 6.10: Filterstruktur bei Ausnutzung der Achsensymmetrie (FIR-Filter 8.Ord)

Die Abbildung 6.12 zeigt die Signale vor und nach der Filterung mit dem implementierten Filter bei der Simulation in ISim.

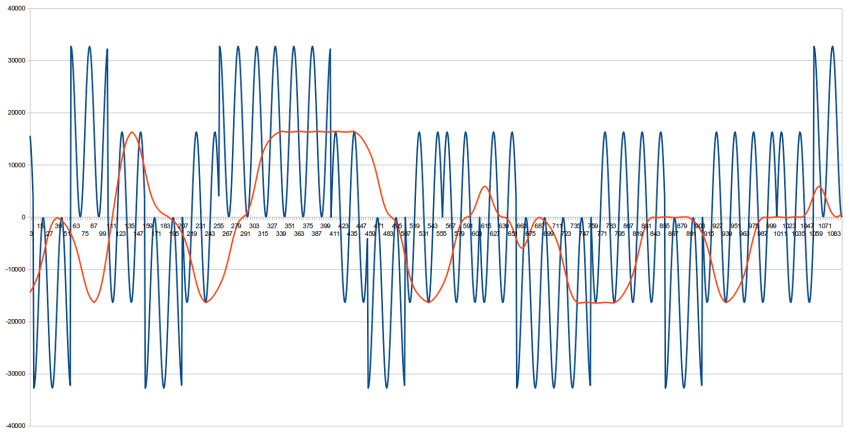

Abbildung 6.11: I_D-Signal vor (blau) und nach (rot) der Tiefpassfilterung

6.2.5 Phasendifferenzdecoder

Als Nächstes werden die empfangenen differenzielle Signale I_D und Q_D in dem Phasendifferenzdecoder wieder in I-bzw.Q-Signale umgewandelt. Dafür wird das aktuelle $(I_{D_n} + jQ_{D_n})$ Signal, durch ein um einen Takt verzögertes Signal $(I_{D_{n-1}} + jQ_{D_{n-1}})$ geteilt. Da in dem Modulator bei der Phasendifferenzbildung das Signal auch mit der Konstante $(0,5 - j0,5)$ multipliziert wurde, muss eine Division des decodierten Signals $(I_D + jQ_D)$ durch $(0,5 - j0,5)$ erfolgen. Bei der Implementierung dieses Vorgehens in VHDL sind zwei grundlegende Probleme zu lösen.

1. Wie teilt man ein ganzzahliges Signal $I_D + jQ_D$ durch eine komplexe Kommazahl $(0,5 - j0,5)$.

2. Teilt man eine Zahl $(10 + j254)$ durch eine andere Zahl $(8 + j256)$ der selben Größenordnung so ist das Ergebnis oft etwas kleiner als Eins $(0,9924 - j0.0080)$. Bei einer Gleitkomma-Umgebung könnte man mit dem Ergebnis ohne Probleme weiter arbeiten, aber bei einer Bitvektordivision ist das Ergebnis immer Null und somit unbrauchbar.

Das erste Problem kann man leicht durch einen mathematischen Handgriff beseitigen. Denn eine Division durch $(0,5 - j0,5)$ kann auch durch Multiplikation mit $(1,0 + j1,0)$ ersetzt werden. Es gilt folgender mathematische Zusammenhang:

$$\frac{a+jb}{0,5-j0,5} = \frac{a+jb}{0,5-j0,5} * \frac{0,5+j0,5}{0,5+j0,5} = \tag{6.10}$$

$$= (a+jb) * \frac{0,5+j0,5}{0,5} = (a+jb) * (1+j1)$$

Der für die Multiplikation mit $(1+j1)$ notwendiger komplexer Multiplizierer wird wie im Kapitel 6.1.5 vorgestellt aufgebaut.
Das zweite Problem kann durch das Verschieben des Dividenden im Zahlenraum beseitigt werden. Nimmt man z.B. die in der Problemstellung vorgestellten Zahlen und multipliziert den Dividenden $(10+j254)$ mit 256 so ergibt sich folgender Zusammenhang:

$$\frac{10+j254}{8+j256} * 256 = \frac{2540+j65024}{8+j256} \approx 254 - j2 \tag{6.11}$$

Somit ist das Ergebnis $\geqslant 1$ und kann ohne Probleme zur Weiterverarbeitung verwendet werden. Für den entworfenen Phasendifferenzdecoder ergibt sich also folgendes Blockschaltbild.

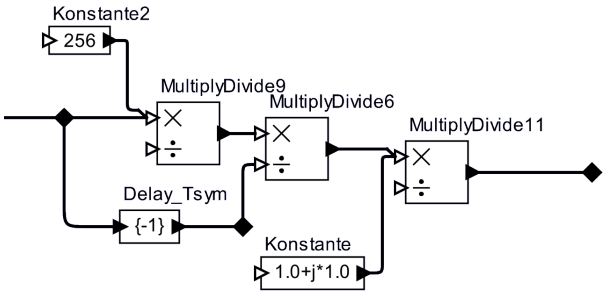

Abbildung 6.12: Blockschaltbild des im VHDL implementierten Phasendifferenzdecoder

Der komplexe Dividierer wird nach dem folgenden mathematischen Zusammenhang durch Multiplikation mit dem komplex konjugierten Divisor und eine reelle Division ersetzt:

$$\frac{a+jb}{c+jd} = \frac{a+jb}{c+jd} * \frac{c-jd}{c-jd} = \frac{(a+jb)(c-jd)}{c^2+d^2} \tag{6.12}$$

6.2.6 Parallel-Seriell-Wandler

Zu erst wurde für die Parallel/Seriell-Wandlung ein Multiplexer implementiert. Dieser hat in der Simulation mit ISim auch einwandfrei funktioniert. Als aber anschließend das Design mit dem ChipScope verifiziert wurde, erfolgte die Wandlung nicht immer fehlerfrei. Die Ursache dieser Fehlfunktion war wieder der Übergang der Daten zwischen zwei unabhängigen Taktebenen. Denn zwischen dem Symboltakt (500kHz), mit dem die parallelen Daten ankommen und dem lokalen 1MHz Takt besteht überhaupt kein Bezug. Wie schon im Kapitel 6.1.4 erläutert, funktioniert dieser Vorgang in einem FPGA nur dann störungsfrei, wenn dazu ein FIFO verwendet wird.
Da das verwendete FIFO IP-Core von Xilinx auch Parallel/Seriell-Wandlung beherrscht, wird der implementierte Multiplexer vollständig aus dem Design entfernt und durch das FIFO ersetzt. Dieser wird wie folgt konfiguriert:

- Unabhäbgige Taktsignale für den Ein- und Ausgang des FIFOs
- Eingangsbus: 2Bit, Ausgangsbus: 1Bit
- Inputs/Outputs: Valid Pin

6.2.7 Early-Late-Gate-Loop

Die implementierte Early-Late-Gate Schleife unterscheidet sich von in dem Kapitel 5.3.2 vorgestellten allgemeinen Aufbau in folgenden Punkten:

1. Anstatt der Integrierer werden Komparatoren eingesetzt.
2. Die Änderung des Symboltaktes erfolgt nicht durch den absoluten Differenzwert der beiden Integrierer sonder additiv in Schritten von ± 1.

Das Bild 6.13 zeigt das dazu gehörige Blockschaltbild.

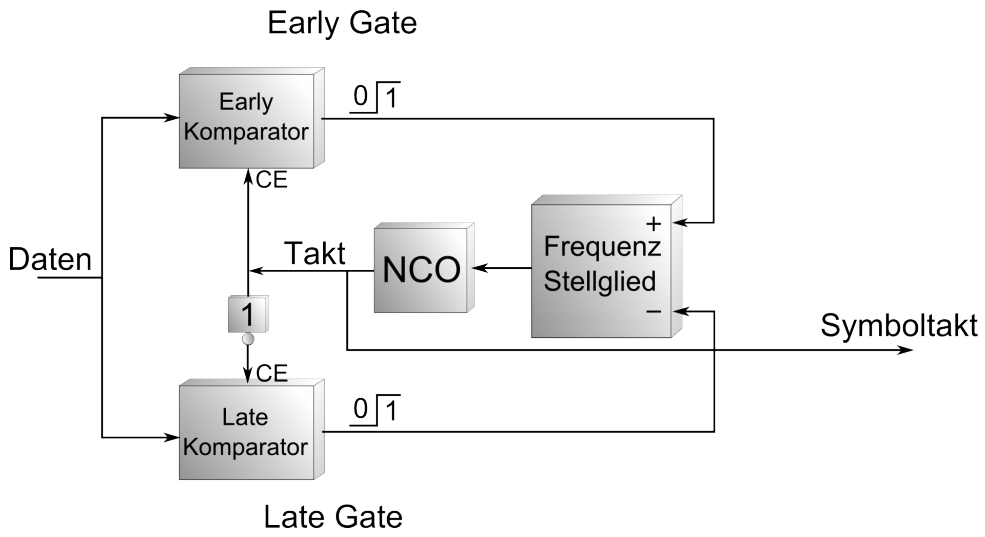

Abbildung 6.13: Blockschaltbild des im VHDL implementierten Early-Late-Gate-Loop

Die Komparatoren detektieren, ob während der jeweiligen positiven Halbtaktperiode eine Zustandsänderung in dem Datenstrom erfolgte. Ist das der Fall, so wird der Ausgang des betroffenen Komparators für kurze Zeit „High“, was zur Erhöhung bzw. Erniedrigung der Frequenz des Symboltaktes um einen Schritt führt. Da der lokale Symboltakt mit einem Zähler von 50MHz Signal abgeleitet wird (Kapitel 6.1.1), erfolgt dessen Beeinflussung durch Anpassen der Zählgrenze (Frequenzstellglied). Daraus folgt, dass bei jedem Schritt die Frequenz des Symboltaktes um ca. 40kHz verändert wird. Diese großen Frequenzsprünge führen also zu ständigem, groben Nachregeln des Symboltaktes. Um diesen Effekt zu minimieren, entschied man sich die Systemfrequenz des Erly-Late-Loop Designs auf 200MHz anzuheben. Dadurch beträgt ein Schritt ca. 10kHz.

Wie man es den folgendem Zeitdiagramm (Abb. 6.14) entnehmen kann, könnte der lokale Symboltakt($f = 500kHz$) zumindest in der Simulation über die entworfene Schleife mit den Daten ($f = 487,83kHz$) synchronisiert werden.

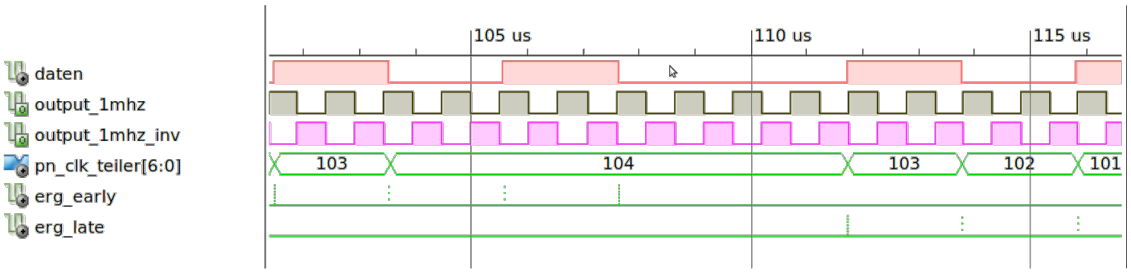

Abbildung 6.14: Zeitdiagramm der Symbolsynchronisation über die Early-Late-Loop

Zum Zeitpunkt $t = 101\mu s$ eilt der 1MHz Takt noch etwas vor. Um den Takt zu verlangsamen, wird die Zählergrenze angehoben. Somit steigt auch die Dauer der Taktperiode von $1,03\mu s$ auf $1,04\mu s$. Eine Zeit lang bleibt der lokale Takt zu den Daten synchron aber irgendwann ist die Abweichung dann wieder so groß, dass nachgeregelt werden muss.

Bedingt durch die notwendigen Systemanpassungen und damit verbundenen Zeitaufwand könnte die implementierte Early-Late-Loop leider nicht mehr in das Gesamtsystem integriert werden.

6.3 Core zur Messung der Bitfehlerrate

6.3.1 BER-Core

Das letzte Element das im Rahmen dieser Bachelorarbeit implementiert wird, ist das BER-Core zur Messung der Bitfehlerrate. Dieses besteht im Grunde aus einem Schieberegister variabler Länge, einem Vergleicher und zwei Zählern (Abb. 6.15). Bedingt durch seine Einfachheit hat dieses Core auch eine Einschränkung: Es können nur Bitfehler in bekannten Sequenzen detektiert werden. Zur Bestimmung der Bitfehlerrate wird das demodulierte serielle Signal zuerst in dem Schieberegister zwischengespeichert. Ist dieses voll, so erfolgt der Vergleich des Schieberegisterinhalts mit der Referenzsequenz. Stimmen die jeweiligen Bits im Schieberegister mit denen aus der Referenzsequenz überein, so wird nur der Gesamtbitzähler hochgezählt. Stimmen Sie aber nicht überein, so wird der Gesamtbitzähler und der Bitfehlerzähler erhöht. Als Alternative zu der Bitfehlerrate kann auch die Sequenzfehlerrate bestimmt werden. Dabei werden die jeweiligen Zähler in der Abhängigkeit von der ganzen Sequenz manipuliert. Die implementierten Gesamtbitzähler und Bitfehlerzähler haben jeweils 64Bit und 32Bit.

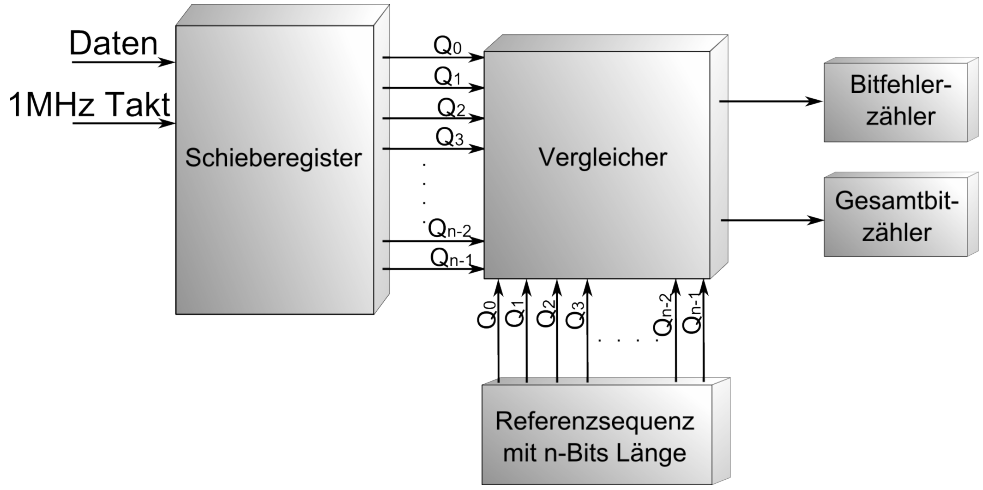

Abbildung 6.15: BER-Core Blockschaltbild

6.3.2 Bestimmung der Bitfehlerrate

Zur Bestimmung der Bitfehlerrate werden die Daten von einer Platine zur anderen übermittelt. Die Verbindung erfolgt über eine Zweidrahtleitung ohne analoge Senden- und Empfangsfilter. Die verwendete Sendesequenz ist 64 Bit lang. Das Bild 6.16 zeigt den zeitlichen Verlauf und das Spektrum des Sendesignals. Die Amplitude der differenziellen Ausgangsspannung beträgt 1V:

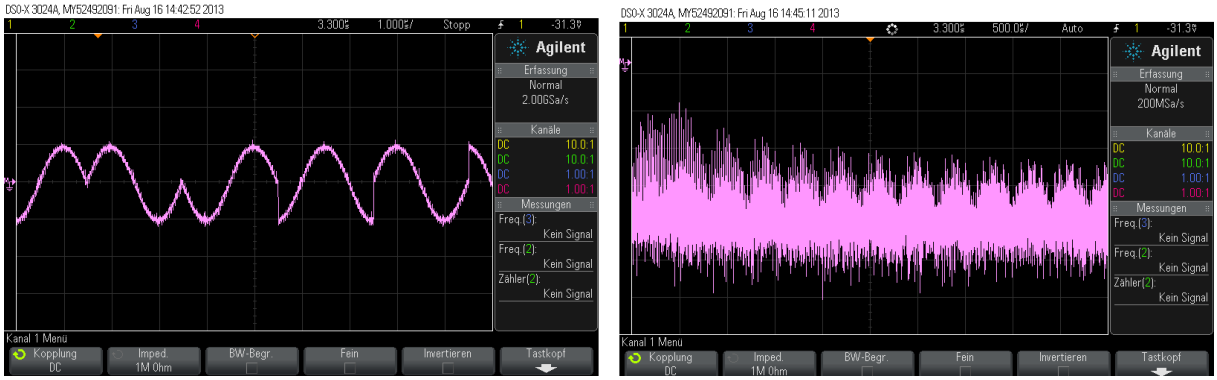

Abbildung 6.16: Zeitfunktion und Spektrum des Sendesignals

Da die zur Symboltaktrückgewinnung notwendige Early-Late-Gate-Loop noch nicht in das Modem integriert ist, wird zur Bestimmung des BER auch der Symboltakt über eine zusätzliche Leitung übergeben. Die Zählerstände von dem Gesamtbitzähler und dem Bitfehlerzähler werden mit ChipScope ausgelesen (Abb. 6.17).

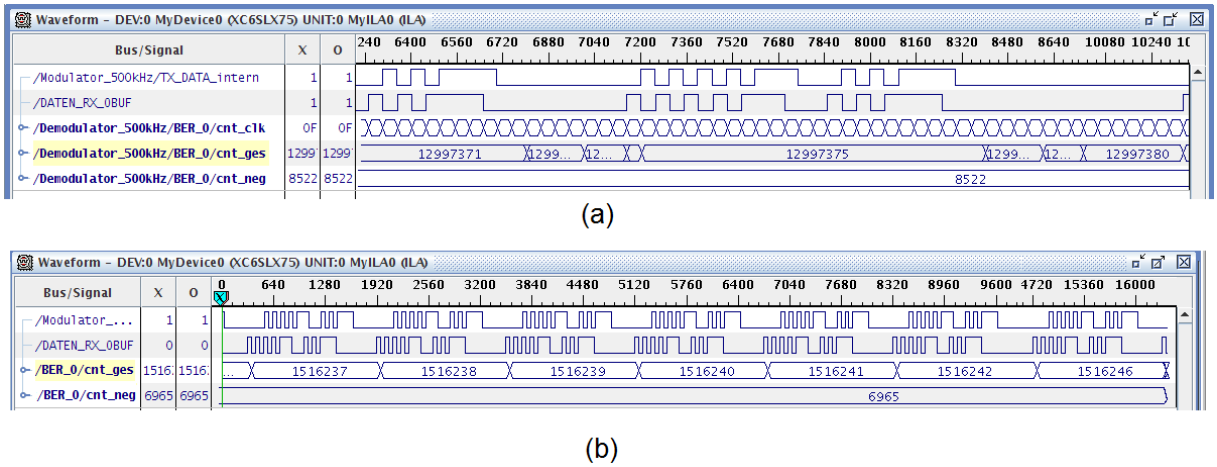

Abbildung 6.17: Messergebnis (a) Bitfehlerrate, (b) Sequenzfehlerrate

Im Bild (a) ist das Ergebnis der Bitfehlerratenmessung und im (b) der Sequenzfehlerratenmessung dargestellt. Die gemessene Bitfehlerrate beträgt ca. 0,0653% und die Sequenzfehlerrate 0,4593%.

7 Fazit und Ausblick

Das Ziel dieser Bachelorthesis ist der Entwurf und die Implementierung eines digitalen Modulationsverfahrens zur Kommunikation über ein Koaxialkabel zwischen der sich in bis zu 4000m Tiefe befindenden Bohrlochsonde und dem Server auf der Oberfläche. Die Kommunikation soll dabei über dieselben Leitungen wie die Stromversorgung erfolgen. Daher wurden zuerst unterschiedliche Modulationsarten miteinander verglichen, um ein passendes Übertragungsverfahren auszuwählen. Die DQPSK wurde wegen ihrer großen Robustheit und vergleichsweise einfachen Aufbau als das optimale Übertragungsverfahren ausgewählt. Für einen Vollduplex-Betrieb musste ein geeignetes Kanalzugriffsverfahren mit den entsprechen Frequenzbändern festgelegt werden.

Vor der eigentlichen Realisierung des DQPSK-Modems in VHDL wurde das Verhalten des Systems untersucht. Dafür wurde zuerst ein Modell der Übertragung, bestehend aus dem Sender, Empfänger und dem Kanal in dem Gleitkomma-Simulator Ptolemy II erstellt. Es wurde untersucht, wie asynchronen Oszillatoren im Sender und Empfänger das System beeinflussen und, wie sich die Bandbegrenzung mithilfe der Impulsformungsfilter auf die notwendige Übertragungsbandbreite auswirkt. Zum Simulieren eines AWGN-Kanals (Additive white Gaussian noise) wurde das Sendesignal mit weißem Rauschen variabler Leistung überlagert. Auch der mögliche Einfluss von Störungen mit selektiver Frequenz könnte gezeigt werden.

Für die Datendekodierung im DQPSK-Empfänger wurden unterschiedliche Verfahren zur Symboltaktrückgewinnung untersucht und bewertet. Dabei hat sich das Early-Late-Gate-Verfahren als die geeignetste Lösung herausgestellt.

Anschließend erfolgte die Implementierung des DQPSK-Modems in VHDL. Außer einigen Ausnahmen, wie z.B. DDS bzw. FIR-Filter, die über den Xilinx®Core Generator erstellt werden können, wurden die meisten Elemente selbst entworfen und implementiert. Die Funktionalität der Elemente bzw. des Modems wurde mit dem VHDL-Simulator ISim simuliert und mit dem VHDL-Logikanalysator ChipScope verifiziert. Außer dem Modem selbst wurde auch noch ein PN-Sequenz-Core implementiert. Dieses erzeugt die benötigten, nahezu zufälligen Sendedaten. Zu Auswertung von Übertragungen wurde ein BER-Core zur Bestimmung der Bitfehlerrate entwickelt.

Als Ergebnis wurden in dieser Arbeit alle Instanzen einer geeigneten Kommunikationssystems mit 1Mbit/s realisiert und mit zwei FPGA-Platinen und einer Zweidrahtverbindung erfolgreich getestet.

Als künftiges Vorgehen sind der Downlink Kanal mit 19,2kbit/s und eventuell ein zweiter Uplinkkanal mit 500 kbit/s als Modifikation der erstellen IP-Cores zu erstellen und das getestete Early-Late-Gate-Loop-Modul zu integrieren. Auch die analogen Elemente der Sende- bzw. Empfangseinheit müssen noch entwickelt werden. Erst dann können Messungen unter realen Bedingungen durchgeführt werden. Zusätzlich kann der erste Uplink-Kanal auf eine 8-DPSK erweitert werden, um somit die zur Verfügung stehende Übertragungsrate bei der vorhandenen Frequenzbandbreite noch weiter zu erhöhen.

Anhang

Für diese Bachelorarbeit relevanten Dateien befinden sich auf dem beigelegten CD-ROM. Diese beinhaltet:

- Filter Koeffizienten Files
- Ptolemy Simulationen
- VHDL Desings
- Xilinx™Compiler Einstellungen für DDS und Tiefpassfilter (FIR)

Literaturverzeichnis

[1] KIT: *Geothermie im Institut für Angewandte Informatik*, `www.iai.fzk.de/www-extern/index.php?id=1805`, [Online] (Stand: August 2012)

[2] Wiegel, Friedrich: *Schaltplan und PCB-Entwurf einer Embedded FPGA-Systemplatine*, Duale Hochschule Baden-Württemberg, Praxisbericht 3. Semester

[3] Wiegel, Friedrich: *Hochtemperatur-Auslegung und Optimierung eines Embedded Systems*, Duale Hochschule Baden-Württemberg, Praxisbericht 5. Semester)

[4] Mäusl, Rudolf: *Digitale Modulationsverfahren*, Hüthig Verlag Heidelberg, 4.Auflage (1995)

[5] Prof. Dr.-Ing. Oberschmidt, Gerald: *Vorlesungsskript zu Grundlagen der Übertragungstechnik*, Duale Hochschule Baden-Württemberg (2013)

[6] Hoffmann, Josef: *Matlab und Simulink in Signalverarbeitung und Kommunikationstechnik*, Addison Wesley Longmann Verlag, 1.Auflage (1999)

[7] Proakis, Wolfgang: *Digitale Fernsehtechnik 2: DVB-S2 - Codierung und Modulation*, Books on Demand GmbH, 1.Auflage (2012)

[8] Eustachi, John G: *Digital communications*, McGraw-Hill Verlag, 5.Auflage (2008)

[9] Xilinx®: *Spartan-6 FPGA Clocking Resources,* `http://www.xilinx.com/support/documentation/user_guides/ug382.pdf`, [Online-PDF] (Stand: August 2013)

[10] Kammeyer, Karl-Dirk: *Nachrichtenübertragung*,Vieweg+Teubner Verlag, 5.Auflage (2011)

[11] Xilinx: *Timing Constraints User Guide,* `http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug612.pdf`, [Online-PDF] (Stand: August 2013)

[12] Xilinx: *LogiCORE IP DDS Compiler v4.0,* `http://www.xilinx.com/support/documentation/ip_documentation/dds_ds558.pdf`, [Online-PDF] (Stand: August 2013)

[13] Xilinx: *LogiCORE FIR Compiler v5.0,* http://www.xilinx.com/support/documentation/ip_documentation/fir_compiler_ds534.pdf, [Online-PDF] (Stand: August 2013)

[14] Ptolemy II: *Ptolemy Project* http://ptolemy.eecs.berkeley.edu/ptolemyII/, [Online] (Stand: August 2013)

[15] Xilinx: *ChipScope Pro and the Serial I/O Toolkit* http://www.xilinx.com/tools/cspro.htm, [Online] (Stand: August 2013)

[16] Linear Technology: *DAC LTC1668 Datasheet,* http://cds.linear.com/docs/en/datasheet/166678f.pdf, [Online] (Stand: August 2013)

[17] Linear Technology: *ADC LTC2249 Datasheet,* http://cds.linear.com/docs/en/datasheet/2249fa.pdf, [Online] (Stand: August 2013)