Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Kim Radmacher

Entwurf, Simulation und Implementierung einer digitalen Audioübertragungsstrecke im Basisband auf einem FPGA

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

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

Kim Radmacher

Entwurf, Simulation und Implementierung einer digitalen Audioübertragungsstrecke im Basisband auf einem FPGA

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Hans Jürgen Micheel
Zweitgutachter : Prof. Dr. rer. nat. Jürgen Reichardt

Abgegeben am 01. Oktober 2013

Kim Radmacher

Thema der Bachelorthesis

Entwurf, Simulation und Implementierung einer digitalen Audioübertragungsstrecke im Basisband auf einem FPGA

Stichworte

Taktrückgewinnung, Synchronisierung, Audiointerface, Kompandierung, Leitungscodierung, Basisband, Aliasing, Phase Locked Loop, ISDN, lineare Verzerrungen, Intersymbol-Interferenz, weißes Rauschen, Entzerrung, Bitfehlerrate, FPGA

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Taktrückgewinnung bei der digitalen Basisbandübertragung. Zudem ist ein Audiointerface für die Übertragung von Audio in ISDN Qualität entwickelt worden. Implementiert ist dieses System auf einer FPGA-Plattform. Verschiedene Parameter des Systems können mittels einer MATLAB GUI eingestellt werden. Des Weiteren ist eine grafische Auswertung des Systemverhaltens durch diese GUI ermöglicht.

Kim Radmacher

Title of the paper

Design, Simulation and Implementation of a Digital Audio Transmission Path in Baseband on a FPGA

Keywords

Timing Recovery, Synchronisation, Audiointerface, Companding, Line coding, Basisband, Aliasing, Phase Locked Loop, ISDN, Lineare Distortion, Intersymbol-Interferenz, White Noise, Equalization, Bit Error Rate, FPGA

Abstract

This thesis deals with the timing recovery for digital baseband transmission systems. An audio interface is developed to transmit audio in ISDN quality. This system is implemented on a FPGA-board. A MATLAB GUI was programmed to set the different parameters of the transmission system and to allow a convenient analysis of the system’s behaviour.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Fernsprechnetz (oder auch Telefonnetz) ist weltweit das größte Kommunikationsnetz. Dieses wurde ursprünglich für die Übertragung analoger Basisbandsignale mit einer Bandbreite von 3,1 kHz konzipiert. Die Deutsche Bundespost entschied jedoch 1979, dass die Ortsvermittlungsstellen des Telefonnetzes in Deutschland digitalisiert werden sollten. Versuche haben gezeigt, dass zwei unabhängige Vollduplex[1]-Kanäle simultan übertragen werden konnten. 1989 begann schließlich der offizielle Betrieb des digitalen diensteintegrierenden Universalnetz ISDN (Integrated Services Digital Network). Damit war die Deutsche Bundespost Vorreiter für ISDN in Europa [1].

Mit Einführung dieses Dienstes wird eine wesentlich breitbandigere Übertragung gefordert, für welche das Kommunikationsnetz nicht ausgelegt wurde. Ein ISDN Primäranschluss beispielsweise hat 30 Nutzkanäle mit einer Datenrate von je 64 kbit/s. Dies führt dazu, dass das Signal durch den Übertragungsweg verzerrt wird. Dabei ist die sogenannte Letzte-Meile hauptsächlich für Verzerrungen verantwortlich. Diese bezeichnet bei Kommunikationsnetzen wie dem Telefonnetz den letzten Abschnitt der Leitung, die zum Teilnehmeranschluss führt. Da heutzutage nahezu alle Vermittlungsstellen über LWL-Kabel[2] verbunden sind, stellt die Letzte-Meile mit Kupferleitungen den Flaschenhals der Datenübertragung dar. Denn hier ist die Geschwindigkeit viel stärker von der Dämpfung und der Entfernung abhängig. Zusätzlich wird dem Signal während der Übertragung auch Rauschen überlagert.

Kann das digitale Signal im Empfänger vollständig rekonstruiert werden, so weist dieses keinen Qualitätsverlust gegenüber dem gesendeten Signal auf. Bei analoger Signalübertragung hingegen besteht durch überlagertes Rauschen stets Qualitätsverlust. Da jede Übertragungsstrecke andere Eigenschaften besitzt, müssen für die Rekonstruktion der digitalen Signal adaptive Entzerrer eingesetzt werden. Diese arbeiten jedoch nur optimal, wenn zuvor eine Taktrückgewinnung stattgefunden hat.

[1] Die Übertragung von Nachrichten in einem bidirektionalen Kommunikationsnetz kann zeitgleich in beide Richtungen stattfinden

[2] Lichtwellenleiter (LWL), Lichtleitkabel (LLK) oder auch häufig Glasfaserkabel genannt

1.1 Spezifikation und Zielsetzung

Mit Anlehnung an ISDN soll ein Simplex-Übertragungssystem auf dem ML507-Board von Xilinx realisiert werden. Ein Kanalmodell sowie die Entzerrung ist hierfür bereits durch eine Bachelor-Thesis eines früheren Studenten der HAW Hamburg gegeben [2]. Diese soll durch eine Taktrückgewinnungseinheit sowie ein Audio-Interface ergänzt werden. Im Weiteren soll eine Spezifikation des zu realisierenden Gesamtsystems aufgestellt werden. Eine Spezifikation ist oftmals Teil des Pflichtenhefts[3] und liefert eine exakte, vollständige und für eine Überprüfung geeignete Beschreibung eines Systems. Während der Entwicklungsphase und nach Fertigstellung wird dieses gegen die Anforderungen der Spezifikation geprüft. Diese gelten als erfüllt wenn sich eine Gegenüberstellung des Systems mit der Spezifikation im Einklang befindet. Werden Unstimmigkeiten festgestellt, so muss eine Nacharbeitung des Systems erfolgen. Abbildung 1.1 zeigt das zu realisierende Gesamtsystem.

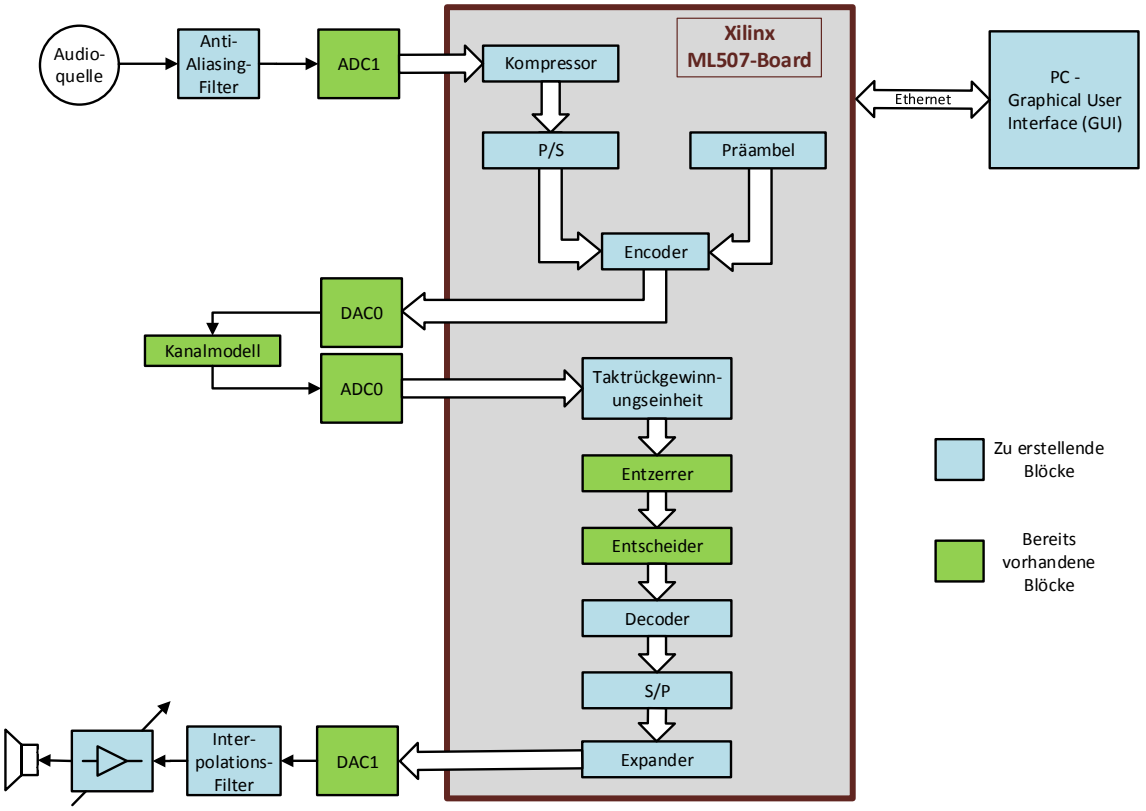


Abbildung 1.1: Blockschaltbild des zu realisierenden Gesamtsystems

[3] Im Pflichtenheft sind nach DIN 69901-5 die vom Auftragnehmer erarbeiteten Realisierungsvorgaben niedergelegt. Diese beschreiben die Umsetzung des vom Auftraggeber vorgegebenen Lastenheftes [3]

Das ML507-Board bietet mit dem AC97 Codec eine Möglichkeit das Audio-Interface zu realisieren [4]. Damit wären ADC und DAC, Anti-Aliasing- und Interpolations-Filter sowie Seriell-/Parallel- und Parallel-/Seriell-Umsetzer vorgegeben. Der Codec ist mit einer Auflösung von 16 Bit und einer Abtastfrequnez von 44,1 kHz weit über dem, was bei ISDN gefordert wird. Über eine Kompandertechnik verfügt der Codec nicht. Daher werden die Komponenten des Audio-Interface selbst erstellt.

- **Audioquelle**:

  Die Quelle soll durch einen 3,5 mm Klinkenanschluss verfügen,
  sodass verschiedene Audiogeräte, wie beispielsweise ein MP3-Player,
  an diesen angeschlossen werden können.

- **Anti-Aliasing-Filter (AAF) und Interpolationsfilter:**
  Grenzfrequenz: $f_g = 3,4\ kHz$
  Sperrfrequenz: $f_c = 4\ kHz$
  Mindest-Sperrdämpfung mit 12 Bit ADC: $a_{min} = 72,2\ dB$

- **Analog-Digital-Umsetzer (ADC0+ADC1):**
  Typ: ADS7947
  Bitauflösung: 12 Bit

- **Digital-Analog-Umsetzer (DAC0+DAC1):**
  Typ: DAC8801
  Bitauflösung: 14 Bit

- **Kompressor/Expander:**
  Verbesserung des Signal-zu-Rausch Verhältnisses durch Kompandertechnik, bzw. Komprimierung von 12 Bit auf 8 Bit und Expandierung von 8 Bit auf 12 Bit

- **Parallel-/Seriell- (P/S) und Seriell-/Parallel- (S/P) Umsetzer:**
  Benötigte Umsetzer für serielle Datenübertragung

- **Encoder/Decoder:**
  Auswahl des bipolar NRZ- oder HDB3-Leitungscodes

- **Kanalmodell:**
  Tief- und Hochpasskanalmodell (5RC-Glieder, 6RC-Glieder, 7RC-Glieder, CR-Glied)

- **Taktrückgewinnung:**
  Taktrückgewinnungseinheit für vorhandene Kanalmodelle

- **Entzerrer:**
  Adaptive Kanalentzerrung mittels Lattice-Entzerrer 3. Ordnung incl. Decision Directed Mode

- **Entscheider:**
  Schwellwertentscheider - abhängig vom eingestellten Leitungscode
- **Präambel:**
  Sender- und empfängerseitig bekannte Trainingssequenzen für die Taktrückgewinnung und den Entzerrer
- **Ausgabe:**
  Ausgabe am Lautsprecher und einstellbarer Verstärkung für Lautstärkeregulierung
- **Graphical User Interface (GUI):**
  Grafische Signalauswertung sowie Systemsteuerung durch eine Matlab-GUI

1.2 Aufbau und Methodik

Damit dem Leser der Inhalt dieser Thesis leicht verständlich gemacht wird, ist der Aufbau klar gegliedert. Es werden jedoch Grundlagen aus unterschiedlichen Gebieten der Informations- und Elektrotechnik, wie beispielsweise der Signalverarbeitung vorausgesetzt. Eine kurze Einführung in die Thematik ist im Kapitel 1 *„Einleitung*“gegeben. Die Anforderungen an das zu realisierende System sind im Abschnitt 1.1 *„Spezifikation“* aufgelistet. In den darauf folgenden Kapiteln 2 *„Digitale Basisbandsignalübertragung“*, 3 *„Taktrückgewinnung“* und *4„Audio-Interface“* werden zunächst die theoretischen Grundlagen erläutert. Im Anschluss folgt dann jeweils ein Entwurf, welcher durch Simulation verifiziert wird. Im Kapitel 5 *„Konzept und Realisierung“* werden schließlich die simulierten Modelle zu einem Gesamtsystem zusammengefasst und in Hardware implementiert, welches in Kapitel 6 *„Inbetriebnahme“* getestet wird. Eine Schlussbetrachtung und einen Vergleich mit der Spezifikation bietet das letzte Kapitel 7 *„Schlussbetrachtung“*.

2 Digitale Basisbandsignalübertragung

2.1 Digitale Signale

Ein Digitalsignal besteht aus einer Folge von Zahlen $a[k]$, welche häufig aus dem Bereich der Dualzahlen $a[k]\epsilon\{0,1\}$ sind. In der digitalen Signalverarbeitung ist ein digitales Signal eine zeit- und amplitudendiskrete Abbildung eines analogen Signals. Das analoge, zeitkontinuierliche Signal wird zunächst zeitdiskret mit f_A abgetastet. Anschließend erfolgt eine Amplitudendiskretisierung durch einen Quantisierer. Diese beiden Vorgänge vereinigt der Analog-Digital-Converter (ADC), vgl. Abb. 2.1. Das entstehende Digitalsignal ist Grundlage für die digitale Signalverarbeitung, welche unterschiedliche Aufgaben erfüllen kann.

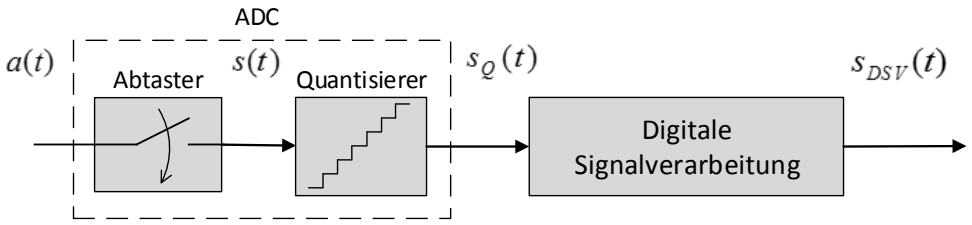


Abbildung 2.1: Modell zur digitalen Signalverarbeitung

Für ein mathematisches Modell des Abtastvorganges eines analogen Signals $a(t)$ wird der Dirac-Impuls $\delta(t)$ mit äquidistanten Abtastwerten im Abstand $1/f_A = T_A$ gewichtet.

$$s(t) = a(t) \cdot \sum_{k=-\infty}^{\infty} T_A \cdot \delta(t - kT_A) = T_A \cdot \sum_{k=-\infty}^{\infty} a(kT_A) \cdot \delta(t - kT_A) \tag{2.1}$$

Die Gewichtung des Dirac-Impulses stellt lediglich die abgetasteten Analogwerte dar, welche die Zahlenfolge $a[k]$ repräsentieren. Somit ergibt sich die zeitdiskrete Abbildung des zeitkontinuierlichen Signals zu

$$s(t) = T_A \cdot \sum_{k=-\infty}^{\infty} a[k] \cdot \delta(t - kT_A). \tag{2.2}$$

Gleichung 2.2 stellt eine ideale Abtastung dar. Die Erzeugung eines Dirac-Impulses ist aus physikalischen Gegebenheiten nicht realisierbar. Daher ist das mathematische Modell des Abtasters durch einen Impulsformer bzw. Formfilter zu ergänzen.

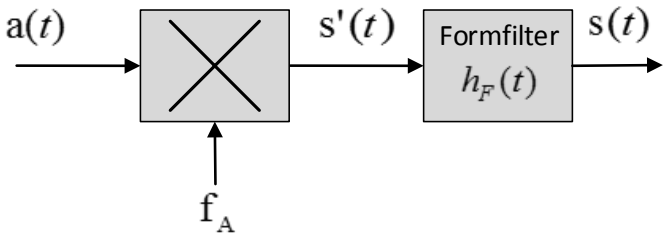


Abbildung 2.2: Modell eines realen Abtaster

Das Formfilter besitzt die Impulsantwort

$$h_F(t) = \frac{1}{T_i} \cdot rect\left(\frac{t - T_i/2}{T_i}\right). \tag{2.3}$$

Dabei steht T_i für die Impulsbreite, welche für den Fall $T_i = T_A$ einen S&H[1] über die gesamte Abtastdauer bedeutet. Damit ergibt sich das Abtasterausgangssignal zu

$$s(t) = s'(t) * h_F(t) = \sum_{k=-\infty}^{\infty} a[k] \cdot rect\left(\frac{t - T_A/2 - kT_A}{T_A}\right). \tag{2.4}$$

Das abgetastete Signal $s(t)$, für das die Annahme einer bipolaren Aussteuerung

$$-1 \leq s(t) \leq 1$$

getroffen wird, wird nun durch einen Quantisierer mit m-Bit codiert. Die Amplitudenquantisierung ergibt sich dann zu 2^m Stufen. Abbildung 2.3 zeigt zwei unterschiedliche Quantisierungskennlinien $s_Q(k)$ für $m = 4$ Bit. Das Quantisierungsintervall ergibt sich bei beiden zu

$$QI = 2^{(-m+1)} = 0.125. \tag{2.5}$$

Im unteren Teil ist die Quantisierungsabweichung $e_Q(k)$ abgebildet.

[1] Ein Sample & Hold (S&H) hält eine analoge Eingangsspannung kurzzeitig auf einen definierten Wert

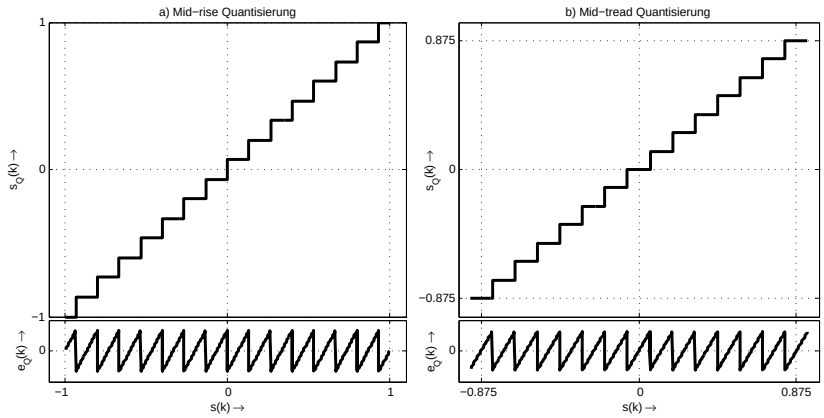


Abbildung 2.3: Zwei unterschiedliche, lineare 4 Bit Quantisierungskennlinien

Bei der linken Kennlinie handelt es sich um eine sogenannte Mid-rise Quantisierungskennlinie. Das Ausgangssignal

$$-1 \leq s_Q(t) \leq 1$$

wird symmetrisch ausgesteuert. Nachteilig hierbei ist, dass für Eingangssignale der Größe Null mit gering überlagertem Rauschen, das quantisierte Ausgangssignal um $\pm 1/(2^m - 1)$ hin- und her springt (granulares Rauschen) [5]. Dies wird mit der rechten Kennlinie, der Mid-tread Quantisierungskennlinie, vermieden. Nachteilig ist, dass eine unterschiedliche Anzahl an Quantisierungsstufen vorliegt. Für symmetrische eine Aussteuerung wird dann einfach auf eine Quantisierungsstufe verzichtet. Für die Mid-tread Kennlinie ergibt sich das Ausgangssignal

$$-1 + 2^{-m+1} \leq s_Q(t) \leq 1 - 2^{-m+1}.$$

Ein Qualitätsmaß der Quantisierung liefert der Signal- zu Störabstand, der auch einfach SNR (*engl. Signal-to-noise ratio*) genannt wird.

$$SNR_Q = 20 \cdot log_{10}(\sqrt{3} \cdot 2^m \cdot s_{eff})\ dB = 10 \cdot log_{10}(3 \cdot 2^{2m} \cdot s_{eff}^2)\ dB \qquad (2.6)$$

Für ein auf eins normiertes Eingangssignal mit dem Effektivwert $s_{eff} = 1/\sqrt{2}$ und 4 Bit ergibt sich ein SNR von 26 dB. Mit dem von der ITU-T[2] festgelegem Wert von 8 Bit für ISDN, ergibt sich ein SNR von 49.8 dB. Dieser Störabstand ist für große Signalpegel völlig ausreichend. Versuche haben gezeigt, dass bei einem Störabstand kleiner 22 dB die Quantisierungsabweichungen als deutlich störend wahrgenommen werden [6]. Deshalb wird bei der PCM-Sprachübertragung ein Störabstand von mindestens 30 dB gefordert [6] [7]. Dieser Störabstand wird nach Gleichung 2.6 ab dem 0,1-fachem des Eingangspegel schon unterschritten, vgl. Abbildung 2.4.

[2]International Telecommunication Union (Telecommunication Standardization Sector))

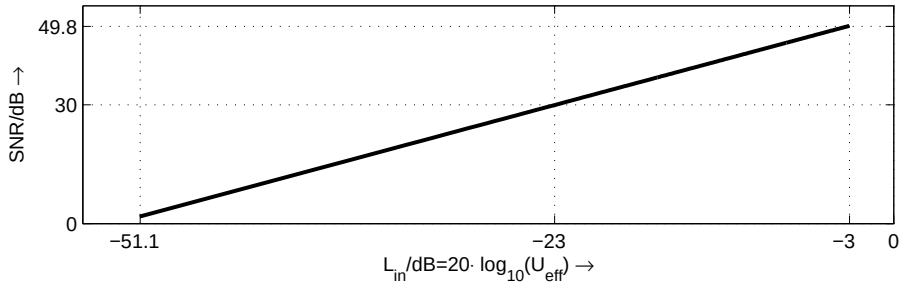


Abbildung 2.4: SNR als Funktion des Eingangspegels bei 8 Bit

Abhilfe könnte ein kleineres Quantisierungsintervall durch Erhöhung der Bitanzahl schaffen. Dies behält jedoch einen pegelabhängigen SNR. Eine andere und bei PCM-Übertragung angewandte Methode bietet die nichtlineare Quantisierung (Kompandierung), auf die im Kapitel 4.3 näher eingegangen wird.

2.2 Systemmodell

In diesem Abschnitt wird das Systemmodell in drei Hauptkomponenten unterteilt und im einzelnen systemtheoretisch analysiert und modelliert (vgl. Abbildung 2.5).

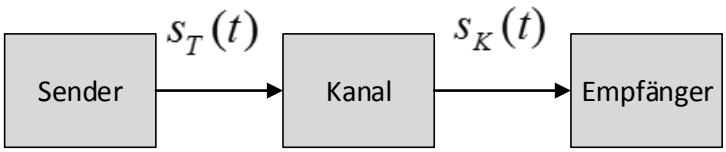


Abbildung 2.5: Blockschaltbild des Übertragungssystems

2.2.1 Sender

In dem Sender geht es darum, eine analoge Signalquelle geeignet über den Kanal an den Empfänger zu übertragen. Bei der Signalquelle handelt es sich hierbei um ein Audiosignal bzw. Sprachsignal. Da die Übertragung digital stattfinden soll, muss das Audiosignal zunächst digitalisiert werden. Dies geschieht wie bereits in Abschnitt 2.1 und Abbildung 2.1 dargestellt.

Beim Abtastvorgang entsteht jedoch im Frequenzbereich eine mit der Abtastfrequenz periodische Fortsetzung des kontinuierlichen Signals. Dies liegt daran, dass aus einer Multiplikation im Zeitbereich, eine Faltung im Frequenzbereich wird. Abbildung 2.6 veranschaulicht das Prinzip. Dabei zeigt $|A(f)|$ das Spektrum des abzutastenden Analogsignals, welches offensichtlich eine Überlagerung zweier gleichgroßer Sinusschwingungen (mit $f_1 = 1kHz$ und $f_2 = 2kHz$) darstellt.

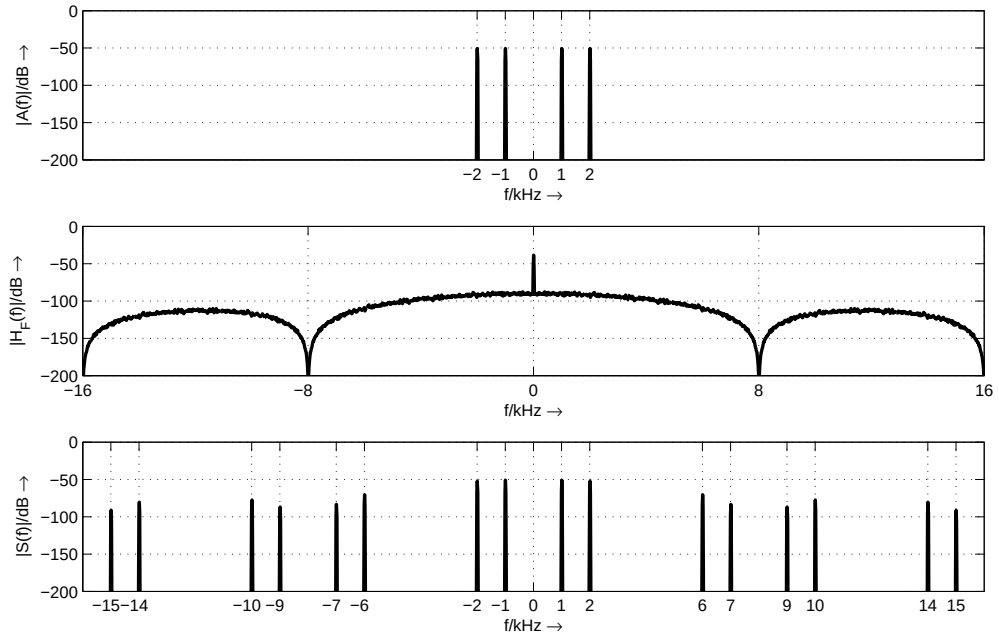


Abbildung 2.6: Spektrale Darstellung der Abtastung ohne Aliasing

$|H_F(f)|$ zeigt den Amplitudengang des Formfilters, welcher sich nach Fourier Transformation aus Gleichung 2.3 ergibt. Diesem ist eine Abtastfrequenz von $f_A = 8kHz$ zu entnehmen. Die darunterliegende Abbildung stellt das Ausgangsspektrum des Abtasters dar. Deutlich zu erkennen ist die Dämpfung durch den Formfilter Amplitudengang, sowie die periodische Fortsetzung von $|S(f)|$ um die Abtastfrequenz. Dies bleibt für beliebig-frequente abzutastende Eingangssignale nicht ohne Folgen. Das Nyquist-Shannon-Abtasttheorem besagt, dass die Abtastfrequenz größer als das Doppelte der im abzutastenden Signal vorkommenden Frequenz sein muss. Diese Bedingung wird in Abbildung 2.6 eingehalten.

In Abbildung 2.7 enthält das abzutastende Signal einen Sinus mit der Frequenz von $f_2 = 5kHz$. Somit wird das Abtasttheorem verletzt.

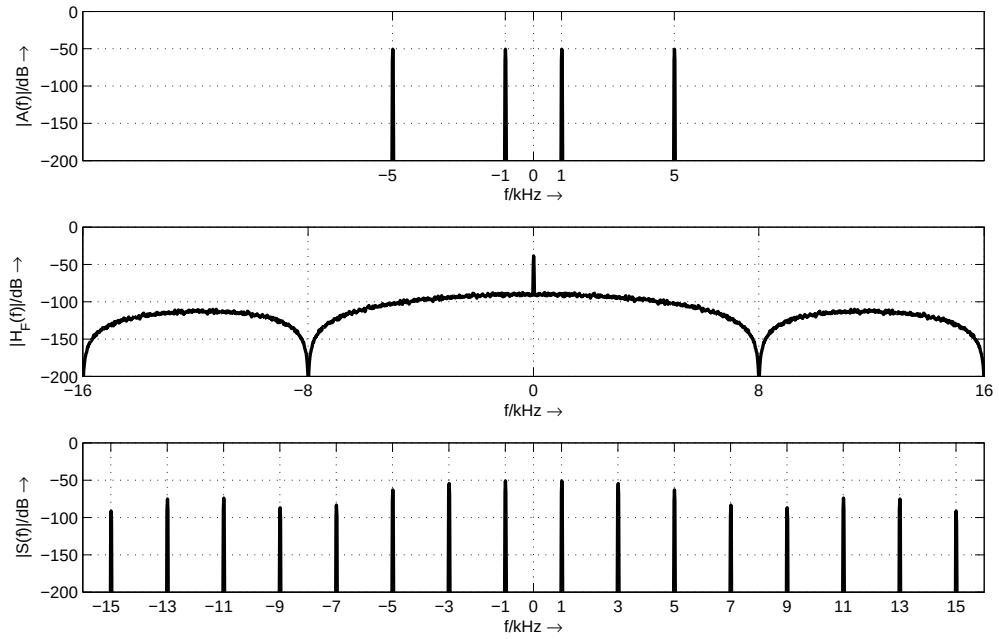


Abbildung 2.7: Spektrale Darstellung der Abtastung mit Aliasing

Im Ausgangsspektrum $|S(f)|$ des Abtasters entsteht der sogenannte Alias-Effekt (kurz Aliasing). Das 5 kHz Signal kann hierraus nicht mehr extrahiert werden. Die entstehenden Alias-Frequenzen lassen sich durch Tabelle 2.1 errechnen, wobei f mit der zu lokalisierenden Eingangsfrequenz zu substituieren ist.

pos. Frequenzen	neg. Frequenzen
$-f$	$+f$
$-f_A \pm f$	$f_A \pm f$
$-2f_A \pm f$	$2f_A \pm f$
...	...

Tabelle 2.1: Aliasing-Frequenzen

Um Aliasing zu vermeiden, muss offensichtlich ein Tiefpassfilter vor den Abtaster geschaltet werden. Daher wird dieser Filter auch einfach Anti-Aliasing-Filter (AAF) genannt. Aus dem Nyquist-Shannon-Abtasttheorem ergibt sich die Nyquist-Frequenz

$$f_N = f_A/2 \tag{2.7}$$

bei der das Anti-Aliasing-Filter genügend Dämpfung erreichen muss, um Aliasing zu vermeiden. Ein Entwurf des AAF folgt im Kapitel 4. Das Systemmodell zur Digitalisierung von Analogsignalen ist nun beschrieben und kann durch Abbildung 2.8 zusammengefasst werden.

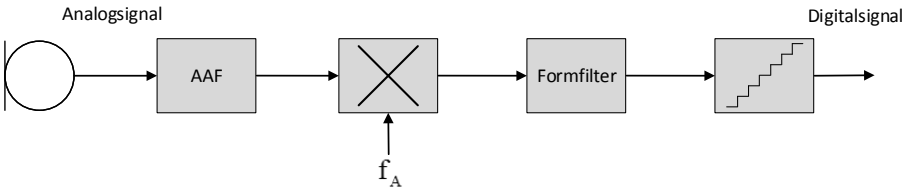


Abbildung 2.8: Modell zur Digitalisierung eines Analogsignals

Eine weitere Funktion, die der Sender beinhaltet, ist die Leitungscodierung, welche Einfluss auf die zu übertragende Pulsfolge nimmt. Die Motivation dabei ist, ein an das Übertragungsmedium angepasstes Sendesignal und genügend Synchronisationsinformation für den Taktregenerator im Empfänger bereitzustellen. Im Rahmen dieser Thesis fällt die Betrachtung der Leitungscodes ausschließlich auf den bipolaren Nonreturn-to-Zero Code (NRZ) und den Bipolarcode 1. Ordnung, oder auch Alternate Mark Inversion Code (AMI) genannt. Telefonkanäle, wie sich im Abschnitt 2.2.2 zeigt, besitzen neben Tiefpass- ebenso Hochpasscharakter. Deswegen ist es von Vorteil ein gleichanteilfreies Sendesignal zu erzeugen. Für ein bipolares NRZ Signal gilt Gleichung 2.4 mit $a[k]\epsilon\{-1, 1\}$. Hier kann fälschlicherweise der Eindruck entstehen, es bestehe keine Gleichleistung im Spektrum.

An dieser Stelle muss auf die Eigenschaft des zu übertragenden Signals eingegangen werden. Die Datenfolge $a[k]$ geht aus dem analogen Sprachsignal hervor. Dieses ist aus Sicht der Informationstheorie ein Zufallssignal. Somit ist die zu übermittelnde Datenfolge ebenso ein Zufallssignal bzw. stochastisches Signal. Von diesem wird vorausgesetzt es sei statistisch unabhängig (unkorreliert) und gleichverteilt [6]. Um eine Aussage über die spektrale Leistungsverteilung des Signals treffen zu können, muss das Leistungsdichtespektrum (LDS) berechnet werden. Dieses erhält man durch die Wiener-Khintchine Beziehung, welche besagt, dass die spektrale Leistungsdichte eines stationären Zufallsprozesses die Fourier-Transformierte der korrespondierenden Autokorrelationsfunktion ist [8]. Die Autokorrelationsfolge (AKF) lässt sich nach der Vorschrift

$$r_{aa}[l] = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{k=-N}^{N} a[k]a[k+l] \tag{2.8}$$

berechnen [9]. Abbildung 2.9a zeigt die AKF einer bipolaren Zufallsdatenfolge. Zu sehen ist ein Dirac an der Stelle $l = 0$. Die Fouriertransformierte der AKF ergibt somit ein konstantes

Leistungsdichtespektrum

$$S_{aa}(f') \bullet\!\!-\!\!\circ\, r_{aa}[l] \tag{2.9}$$

(vgl. Abbildung 2.9b).

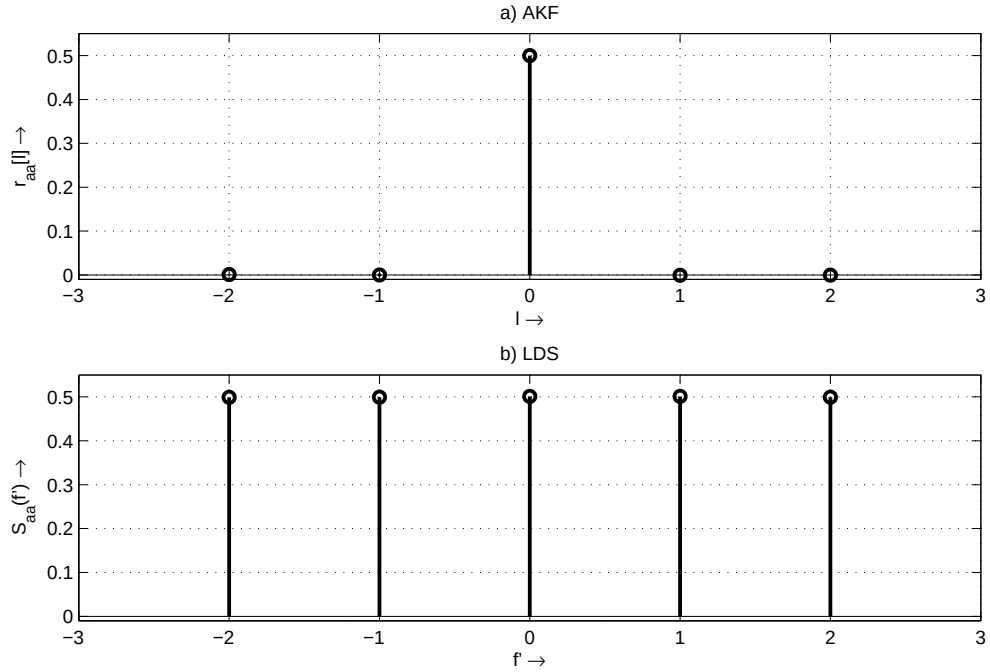


Abbildung 2.9: Autokorrelationsfolge und LDS einer bipolaren Binärfolge

Das Leistungsdichtespektrum ist auf normierter Frequenzachse f' dargestellt. Eine absolute Darstellung wird erlangt, indem die AKF auf die absolute Zeitachse τ bezogen und anschließend fourier-transformiert wird. Es kann gezeigt werden, dass dies zu einer Gewichtung der AKF sowie des LDS, durch T_0 bewirkt [9]. Mit der Fourier-transformierten Impulsantwort des Formfilter nach Gleichung 2.3

$$h_F(t) \circ\!\!-\!\!\bullet\, \underline{H}_F(f) = si(\pi f T_0) \cdot e^{-j\pi f T_0}, \tag{2.10}$$

erhält man schließlich das Leistungsdichtespektrum des Sendesignals.

$$S_{ss}(f) = |H_F(f)|^2 \cdot S_{aa}(f) \tag{2.11}$$

Für die Messung von Leistungsdichtespektren wird im Weiteren die Welch-Methode verwendet und kurz erläutert. Da es sich um stochastische Signale handelt, kann im Gegensatz zu deterministischen Signalen nur eine Spektralschätzung erfolgen. Die Welch-Methode bietet

die am häufigsten angewandte Form der Spektralschätzung [8]. Diese zerlegt eine Eingangsfolge, aus der das Leistungsdichtespektrum geschätz werden soll, in mehrere Teilblöcke. Jeder Teilblock wird mit einer Fensterfunktion[3] bewertet. Ziel dabei ist es, eine Periodizität zu erzeugen. Dies vermindert die Auswirkungen des Leck-Effekts[4]. Dieser tritt entweder bei nichtperiodischen Signalen auf oder bei Signalen, deren Periodendauer nicht mit dem Fenster übereinstimmt. Aus den gefensterten Teilfolgen werden Periodogramme gebildet, dessen Mittelwerte als Schätzgröße der spektralen Leistungsdichte gelten [8].

Abbildung 2.10 zeigt das gemessene, amplitudennormierte Leistungsdichtespektrum einer bipolaren NRZ Zufallsfolge von 2^{18} Bits. Die Parameter der Welch-Methode wurden empirisch ermittelt, sodass eine möglichst erwartungsgetreue Schätzung erreicht wird. Als Fenster wurde ein Blackman-Fenster verwendet.

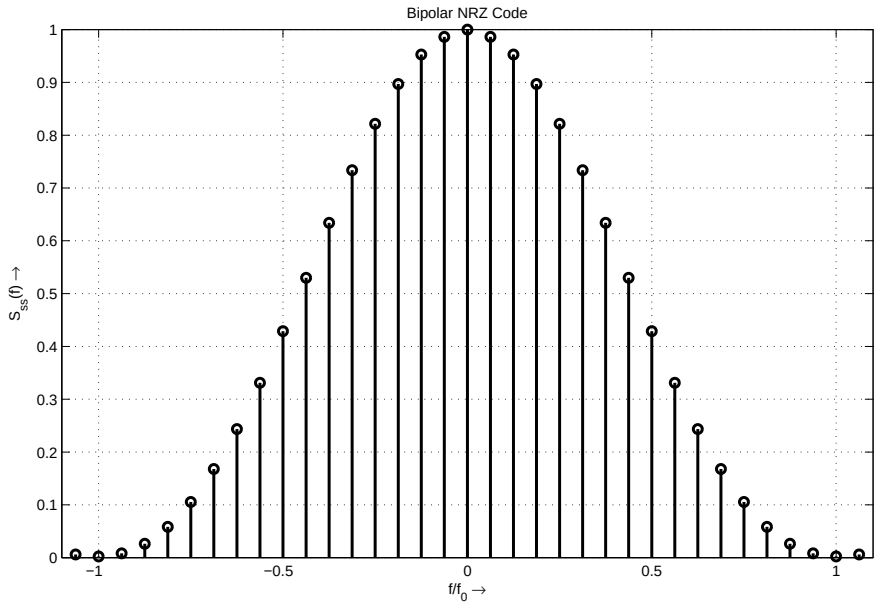


Abbildung 2.10: Spektrale Leistungsdichte des bipolar NRZ-Codes

Wie das Spektrum zeigt, ist dieses nicht gleichanteilfrei. Deutlich zu erkennen ist der Formfilteramplitudengang, welcher eine si^2 Gewichtung verursacht. Bei der Taktfrequenz f_0 ist keinerlei Spektralinformation vorhanden. Dies macht eine Taktrückgewinnung aus dem bipolaren NRZ Signal ohne weitere Signalverarbeitung nicht möglich.

[3] Eine Fensterfunktion gewichtet eine Datenfolge innerhalb eines Ausschnittes (Fenster), welche in die nachfolgende Berechnungen eingehen

[4] Der Leck-Effekt beschreibt das Phänomen bei der Spektralanalyse, dass Frequenzanteile vorkommen, die bei einer theoretisch unendlich langen Beobachtungszeitfolge nicht enthalten sind

Der AMI-Encoder besitzt ein Ausgangssignalalphabet $c[k]\epsilon\{-1, 0, 1\}$. Da für die Darstellung von binären Daten drei Zustände verwendet werden, handelt es sich beim AMI-Code um einen sogenannten pseudoternär-Code. Das Blockschaltbild des Encoders ist in Abbildung 2.11 dargestellt.

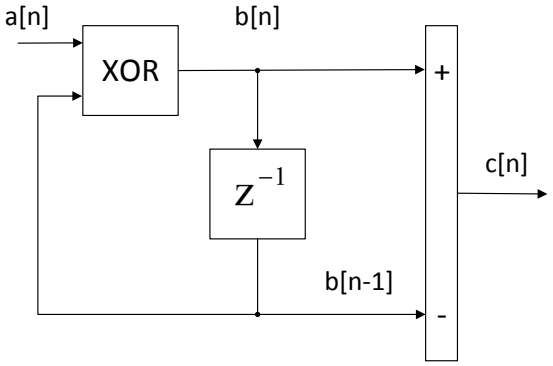


Abbildung 2.11: AMI-Encoder

Aus dem Blockschaltbild kann die Kodiererausgangsgleichung

$$c[n] = b[n] - b[n-1] \tag{2.12}$$

abgeleitet werden. Diese ist der Vorkodierergleichung

$$b[n] = a[n] \oplus b[n-1] \tag{2.13}$$

übergeordnet. Um den Frequenzgang des Encoders zu erhalten wird aus Gleichung 2.12 die Impulsantwort

$$h_c(t) = \delta(t) - \delta(t - T_0) \tag{2.14}$$

erzeugt. Nach Fourier-Transformation und Umformungen erhält man den Kodieramplitudengang.

$$|H_c(f)|^2 = sin(\pi f T_0)^2 \tag{2.15}$$

Dieser hat offensichtlich einen sinusförmigen Verlauf. Somit ergibt sich das Leistungsdichtespektrum des Sendesignals zu

$$S_{ss}(f) = |H_F(f)|^2 \cdot |H_c(f)|^2 \tag{2.16}$$

Wie am Leistungsdichtespektrum in Abbildung 2.12 zu sehen ist, ist dieses im Gegensatz zum bipolaren NRZ-Code gleichanteilsfrei. Jedoch ist auch hier die Taktrückgewinnung ohne Signalverarbeitung nicht möglich, da bei f_0 keine Spektralinformation vorhanden ist.

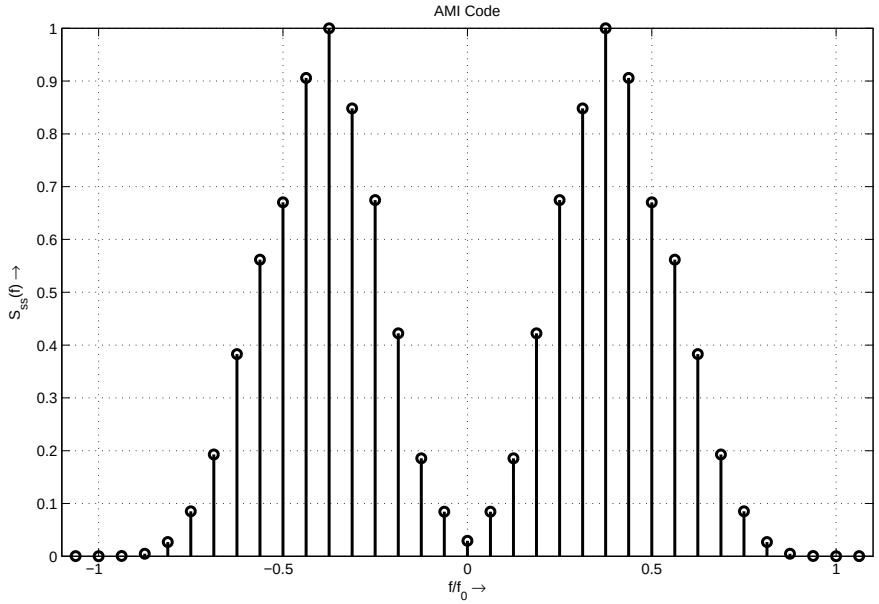


Abbildung 2.12: Spektrale Leistungsdichte des AMI-Codes

2.2.2 Kanal

Der Kanal bildet die Verbindungsstrecke zwischen Sender und Empfänger. Üblicherweise ist dieser in seinen Eigenschaften vorgegeben, sodass sich Sender und Empfänger auf diesen abstimmen müssen. Wie es bei klassischen Fernsprech-Übertragungssystemen üblich ist, wird ausschließlich eine kabelgebundene Übertragungsstrecke als Kanal behandelt. Aus der Leitungstheorie ist bekannt, dass es bei der Übertragung von Signalen, abhängig von Frequenz- sowie Leitungslänge, zu Dämpfungs- und Phasenänderungen kommt. Diese werden im Allgemeinen als Verzerrungen bezeichnet. Ein einfaches Beispiel für Verzerrungen eines Übertragungskanals bietet ein Tief- oder Hochpassfilter 1. Ordnung, bestehend aus einem Kondensator und einem Widerstand. Die Grenzfrequenz $f_g = \frac{1}{2\pi T}$ wird von der Zeitkonstante $T = R \cdot C$ bestimmt. Die Übertragungsfunktion des RC-Tiefpasses im Frequenz- und Laplacebereich lauten

$$\mathcal{F}\{h_{TP}(t)\} = \underline{H}_{TP}(j\omega) = \frac{1}{j\omega T + 1} \tag{2.17}$$

$$\mathcal{L}\{h_{TP}(t)\} = H_{TP}(s) = \frac{1}{sT + 1} \tag{2.18}$$

Für den CR-Hochpass

$$\mathcal{F}\{h_{HP}(t)\} = \underline{H}_{HP}(j\omega) = \frac{j\omega T}{j\omega T + 1} \tag{2.19}$$

$$\mathcal{L}\{h_{HP}(t)\} = \ H_{TP}(s) = \frac{sT}{sT + 1} \tag{2.20}$$

Um die Verzerrungen im Zeitbereich darzustellen, wird ein Rechteckimpuls der Breite T_0 auf den Tiefpass gegeben. Die Antwort wird durch Faltung des Eingangssignals mit der Impulsantwort erhalten. Mit der Laplace-Transformation wird aus der Faltung eine Multiplikation.

$$s_E(t) = h_{TP}(t) * s_{rect}(t) \circ\!\!-\!\!\bullet\, S_E(s) = H_{TP}(s) \cdot S_{rect}(s) \tag{2.21}$$

Mit dem Rechteckimpuls im Laplacebereich

$$S_{rect}(s) = \frac{1}{s}(1 - e^{-T_0 s}) \tag{2.22}$$

ergibt sich das Ausgangssignal nach Gleichung 2.21 zu

$$S_E(s) = \frac{\frac{1}{T}}{\frac{1}{T} + s} \cdot \frac{1}{s}(1 - e^{-T_0 s}). \tag{2.23}$$

Die Laplace-Rücktransformation ergibt die Filterantwort im Zeitbereich

$$S_E(s) \bullet\!\!-\!\!\circ\, s_E(t) = \begin{cases} 1 - e^{-\frac{1}{T}}, & t \leq T_0 \\ e^{-\frac{(t-T_0)}{T}}, & t > T_0 \end{cases} . \tag{2.24}$$

Abbildung 2.13 a) zeigt die Filterantwort für eine Zeitkonstante $T = 1.59\mu s$ und $T_0 = \frac{1}{64kHz} = 15.62\mu s$. Die Verzerrungen sind noch nicht besonders stark ausgeprägt. Dies ist daran zu erkennen, dass die Filterantwort bei $2T_0$ bereits vollständig abgeklungen ist.

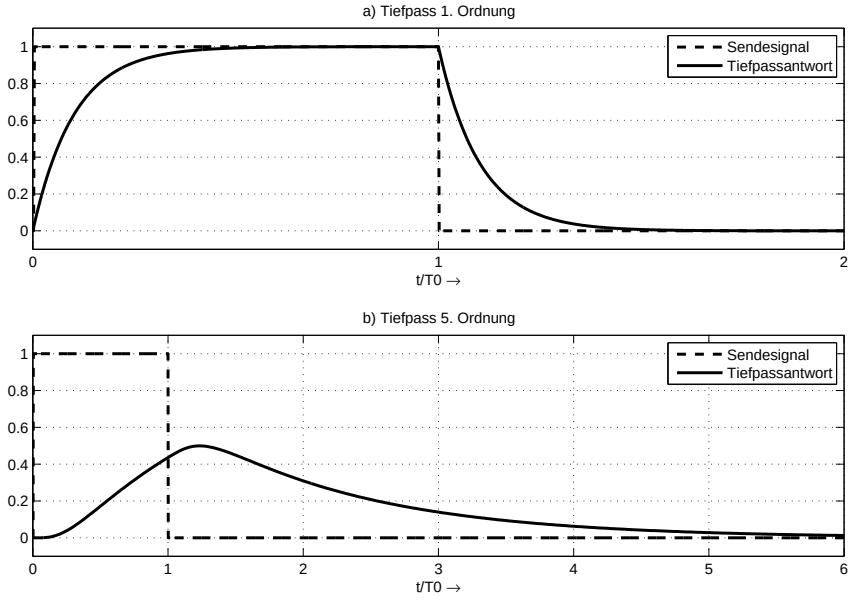


Abbildung 2.13: Antwort eines Tiefpasses 1. und 5. Ordnung auf einen Rechteckimpuls der Breite T_0

Stärkere Verzerrungen zeigen sich bei Filter höherer Ordnung, wie es in Abbildung 2.13 b) dargestellt ist. Hierbei handelt es sich um eine Kaskadierung von fünf RC-Tiefpässen 1. Ordnung. Deutlich zu erkennen ist eine Filterantwort, die erst nach $6T_0$ abgeklungen ist. Die Rede ist hierbei vom sogenannten Symbolübersprechen oder auch Intersymbolinterferenz (ISI). Dieser im Allgemeinen unerwünschte Effekt entsteht, wenn die 1. Nyquistbedingung nicht erfüllt wird. Diese besagt, dass die Impulsantwort eines Systems mit der Abtastrate T_A, Nullstellen zu den Zeitpunkten $k \cdot T_A$ für $k \, \epsilon \, \mathbb{N}^*$ aufweisen muss.

Fernmeldeleitungen mit einer Länge von mehrere Kilometern verursachen deutlich stärkere Verzerrungen als es ein Tiefpass 1. Ordnung verursacht. Jedoch variiert die Länge der Fernmeldeleitungen und damit auch die Verzerrungen. Als Kanal wird daher ein Modell herangezogen, welches unterschiedlich starke Verzerrungen erzeugen kann. An der Hochschule für Angewandte Wissenschaften Hamburg besteht bereits ein Kanalmodell. Dieses wird für Versuche im Bereich der digitalen Basisbandsignalübertragung genutzt. Die Parameter des Kanalmodells wurden bereits in der Bachelor-Thesis [2] genutzt um einen Kanal zu modellieren. Dieses Kanalmodell findet auch hier Anwendung. Der Kanal besteht aus einer Tiefpass-Kaskadierung von wahlweise fünf, sechs oder sieben RC-Gliedern, welche über Relais hinzu- oder weggeschaltet werden können. Für den Einfluss der Hochpasscharakteristik kann ein zusätzliches CR-Hochpassfilter 1. Ordnung hinzugeschaltet werden. Die unterschiedlichen Amplitudengänge des Kanalmodells sind in Abbildung 2.14 dargestellt.

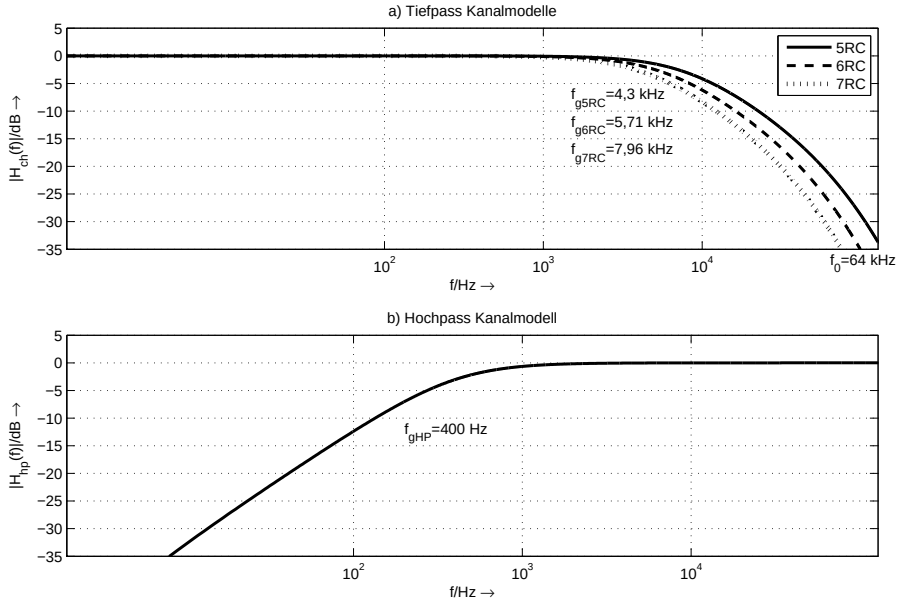


Abbildung 2.14: Amplitudengang der Kanalmodelle

Bei der Übertragung im Fernsprechnetz besteht grundsätzlich eine Bandbegrenzung auf den Frequenzbereich [5]

$$300\ Hz \leq f \leq 3400\ Hz \tag{2.25}$$

Die Grenzfrequenz des Hochpasskanalmodells ist mit 400 Hz etwas darüber. Das Kanalmodell soll dennoch für die Übertragung genutzt werden. Erwartungsgemäß werden die Auswirkungen bei bipolarem NRZ-Code wegen des Gleichanteils zu mehr Verzerrungen führen. Der Phasenfrequenzgang

$$\varphi(j\omega) = arctan\left(\frac{Im\{H_{ch}(j\omega)\}}{Re\{H_{ch}(j\omega)\}}\right) \tag{2.26}$$

gibt Ausschluss über die Phasenänderung. Die Gruppenlaufzeit

$$\tau_{Gr}(j\omega) = -\frac{d\varphi(j\omega)}{d\omega}. \tag{2.27}$$

zeigt die frequenzabhängige Verzögerungszeit, die ein Signal durch einen Filter bzw. Übertragungskanal erfährt (vgl. Abbildung 2.15).

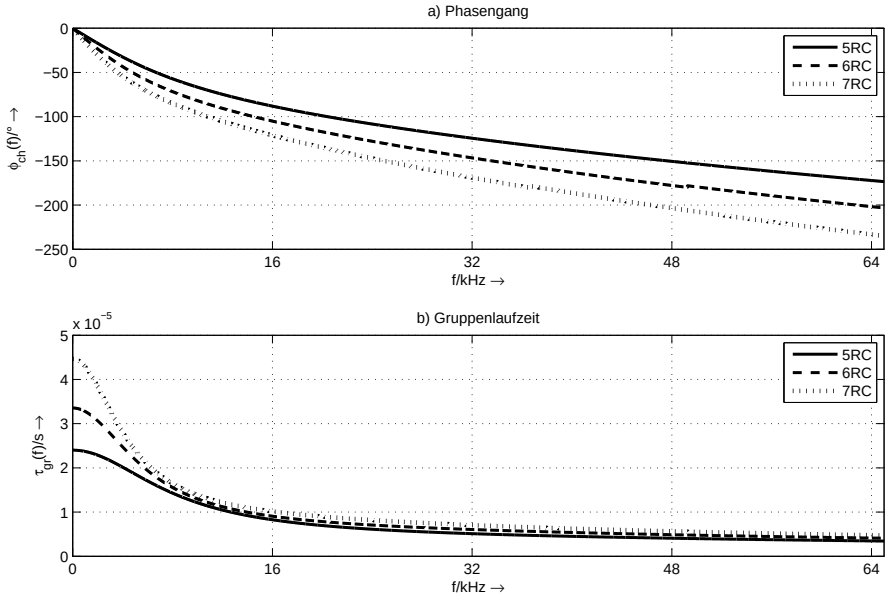


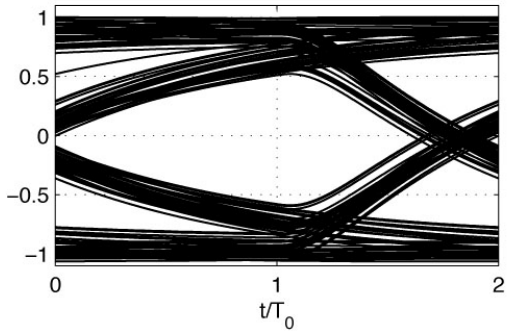
Abbildung 2.15: Phasengang und Gruppenlaufzeit der Kanalmodelle

Wie stark die Verzerrungen sind, kann anhand eines Augendiagramms dargestellt werden. Dies zeigt die Antwort eines Systems auf eine Zufallsfolge, überlagert im Abstand der Periodendauer für eine gewisse Anzahl an gesendeten Symbolen. In Abbildung 2.16 sind einige amplitudennormierte Augendiagramme dargestellt.

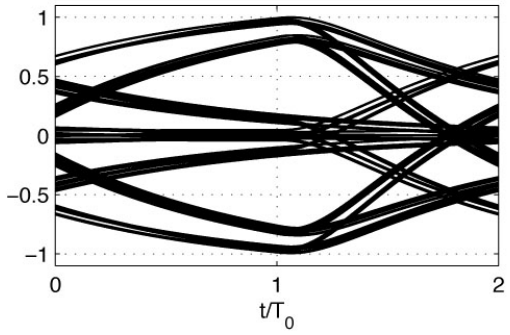
Die Augenöffnung wird durch zunehmende Dämpfung vermindert, was zu mehr ISI führt. Dem Tiefpass 3. Ordnung mit Hochpassanbindung ist nur eine leicht verminderte vertikale Augenöffnung zu entnehmen. Im Gegensatz dazu bringt der Tiefpass 5. Ordnung deutlich ISI mit sich (vgl. Abbildung 2.16 a),c).

Der Vor- und Nachteil der zwei betrachteten Leitungscodes des Senders kann auch am Augendiagramm eingesehen werden. Hier bestätigt sich, dass die Gleichanteilsfreiheit des AMI-Codes zu weniger Verzerrungen führt, als beim bipolaren NRZ-Code (vgl. Abbildung 2.16 a),b)).

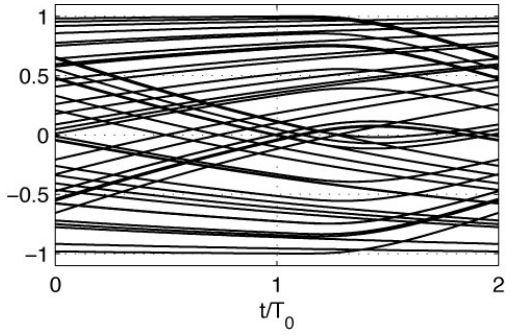
Neben Verzerrungen wird dem Signal während der Übertragung Rauschen überlagert. Die Ursache dieser additiven Störungen liegt üblicherweise im Rauschen passiver und aktiver Bauelemente, in den Einstreuungen durch Rundfunksender, Nebensprechen und durch Quantisierungsrauschen. Der Störabstand kann variieren, liegt jedoch typischerweise oberhalb von 30 dB [5].



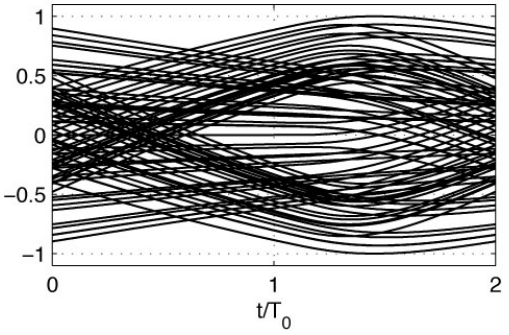
(a) Bipolar NRZ, 3RC- und HP-Kanal



(b) AMI-Code, 3RC- und HP-Kanal



(c) Bipolar NRZ, 5RC-Kanal



(d) AMI-Code, 5RC-Kanal

Abbildung 2.16: Augendiagramm verschiedener Kanäle, sowie Leitungscodes

2.2.3 Empfänger

Die Einflüsse des Kanals wurden dargestellt. Nun ist es Aufgabe des Empfängers, aus dem übermittelten Signal die korrekte Nachricht bzw. Symbolfolge zu rekonstruieren. Starke Intersymbolinterferenz führt dazu, dass im Empfänger eine einfache Schwellwertentscheidung nicht ausreicht, um entscheiden zu können, ob eine ’1’, ’0’ oder ’-1’ gesendet worden ist. Für diesen Fall werden Entzerrer eingesetzt. Diese besitzen idealerweise die Eigenschaft, die Tiefpasscharakteristik des Kanals durch inverses Übertragungsverhalten auszugleichen.

$$\underline{H}_E(f) = \underline{H}_{ch}(f)^{-1} \tag{2.28}$$

Wäre dies der Fall, würde am Entzerrerausgang das exakte Sendesignal rekonstruiert. Der Aufwand dafür wäre immens und steht in keinem Verhätnis zur praktischen Anwendung. Ein Entzerrerausgang muss für einen optimalen Fall lediglich die 1. Nyquistbedingung, welche bereits im Abschnitt 2.2.2 erläutert wurde, aufweisen. Ein nachgeschalteter Schwellwertentscheider sorgt dann für die korrekte leitungscodierte Symbolfolge, sodass schließlich dem

Decoderausgang die Digitalfolge $a[n]$ entnommen werden kann. Abbildung 2.17 zeigt das Prinzipblockschaltbild eines Empfängers.

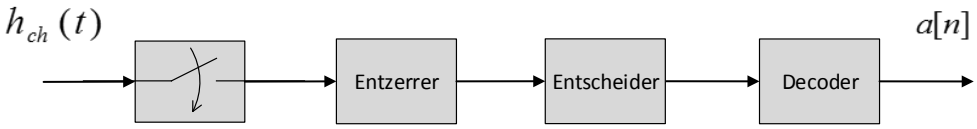


Abbildung 2.17: Prinzipblockschaltbild eines Empfängers

Bei bekannter Kanalübertragungsfunktion kann im einfachsten Fall ein FIR-Filter (Finite-Impulse-Response-Filter) als Entzerrer eingesetzt werden. Die Koeffizienten können mit gegebener Rechteckimpulsantwort des Kanals durch die Rekursionsformel

$$a_n = \begin{cases} \dfrac{1}{h_{ch}(T_0)}, & n = 0 \\ -\sum_{i=0}^{n-1} \dfrac{a_i \cdot h_{ch}([n-i+1]T_0)}{h_{ch}(T_0)}, & n > 0 \end{cases} \tag{2.29}$$

berechnet werden [10]. In der Bachelor-Thesis [2] ist bereits ein Entzerrer entwickelt, welcher für die Realisierung dieses Systems genutzt werden soll. Dieser ist ein sogenannter Lattice-Entzerrer, welcher auf dem Least-Mean-Square-Algorithmus[5] (LMS-Algorithmus) basiert. Da der LMS-Algorithmus jedoch nur langsam konvergiert, wird diesem ein Lattice-Prädiktor(dekorreliert die Eingangsdatenfolge) vorgeschaltet. um die Konvergenzgeschwindigkeit zu erhöhen.

Um die Arbeitsweise eines einfachen FIR-Entzerrers nach Gleichung 2.29 zu veranschaulichen, ist ein Ausgangssignal eines Entzerrers 4. Ordnung in Abbildung 2.18 b) abgebildet.

[5]Der LMS-Algorithmus beruht auf der Methode des steilsten Abstiegs (auch Gradientenverfahren genannt) um allgemeine Optimierungsprobleme lösen zu können

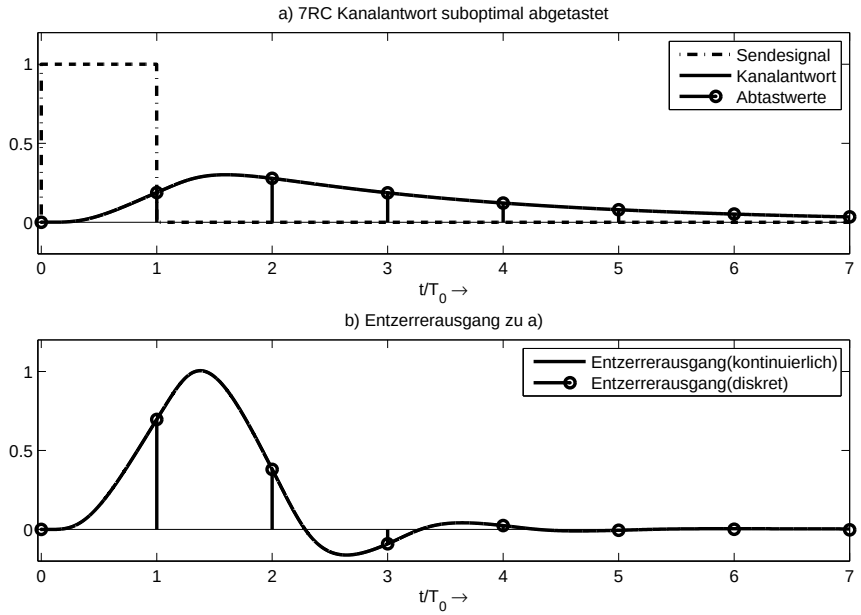


Abbildung 2.18: 7RC Kanalausgang und Entzerrerausgang bei suboptimaler Abtastung

Das FIR-Filter ist ein digitales Filter und arbeitet nicht zeitkontinuierlich, sondern mit diskreten Abtastwerten. Hier entsteht die Problematik des Abtastzeitpunktes. Wird die Kanalantwort ohne Ausgleich der Gruppenlaufzeit bzw. Phasenverschiebung abgetastet, ergeben sich die in Abbildung 2.18 b) diskreten Entzerrerausgangswerte. Wie bereits erwähnt ist es nötig, die 1. Nyquistbedingung am Entzerrerausgang zu erzeugen, was hier offensichtlich nicht der Fall ist. ISI wird demnach nicht beseitigt. Ohne das Angleichen an die Kanallaufzeit ist eine Entzerrung daher wenig effektiv. Abbildung 2.19 a) zeigt die Kanalantwort mit optimaler Abtastung. Der Entzerrerausgang weist somit Nullstellen im Taktraster auf, vgl. Abbildung 2.19 b).

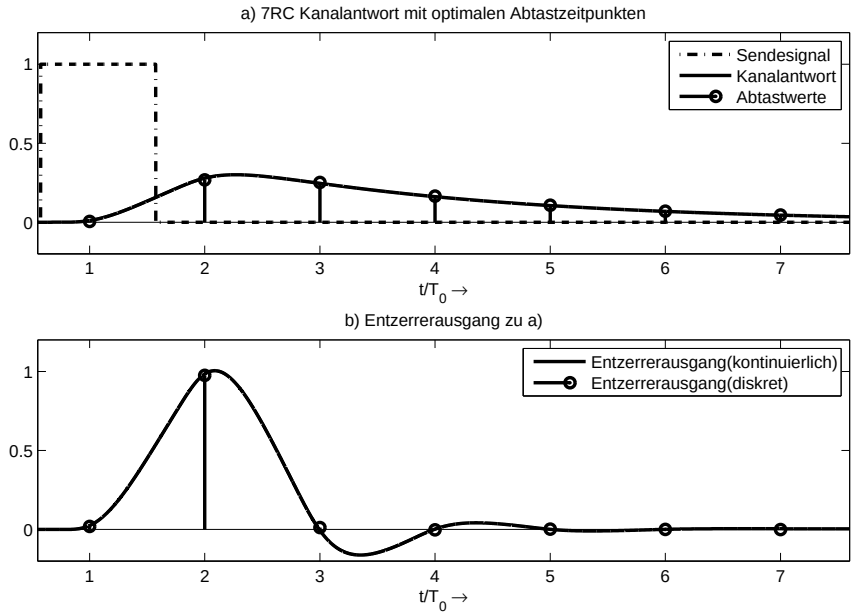


Abbildung 2.19: 7RC Kanalausgang und Entzerrerausgang bei optimaler Abtastung

Betrachtet man die Gruppenlaufzeit der Kanalmodelle aus Abbildung 2.15 b) etwas genauer, so ist diesen bei 64 kHz die auszugleichende Laufzeit zu entnehmen. Tabelle 2.2 listet die kanalspezifischen Gruppenlaufzeiten auf.

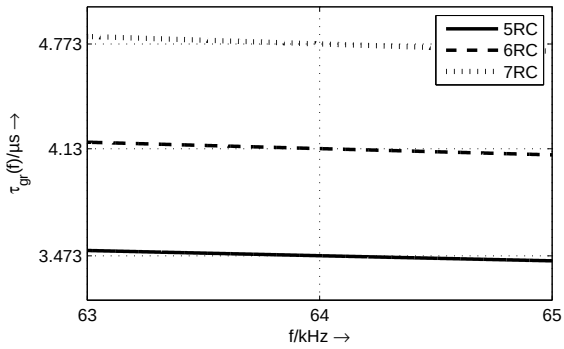


Abbildung 2.20: Gruppenlaufzeiten

Kanalmodell	$\tau_{CH}[\mu s]$
5RC	3,473
6RC	4,130
7RC	4,773

Tabelle 2.2: Gruppenlaufzeiten bei 64 kHz

Abbildung 2.21 zeigt die Abtastung für die drei Tiefpasskanalmodelle mit den Takt-Verzögerungszeiten nach Tabelle 2.2. Folgend gilt dies als optimale Abtastung.

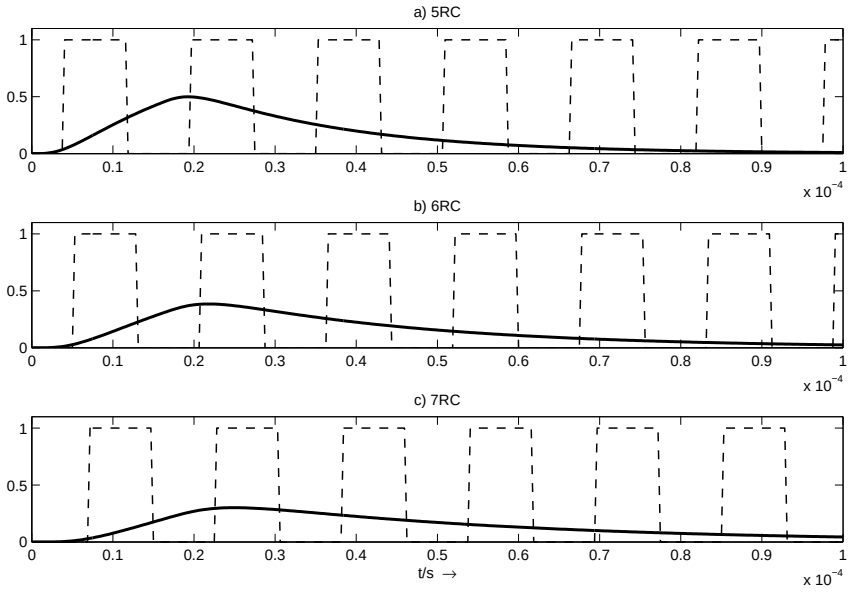
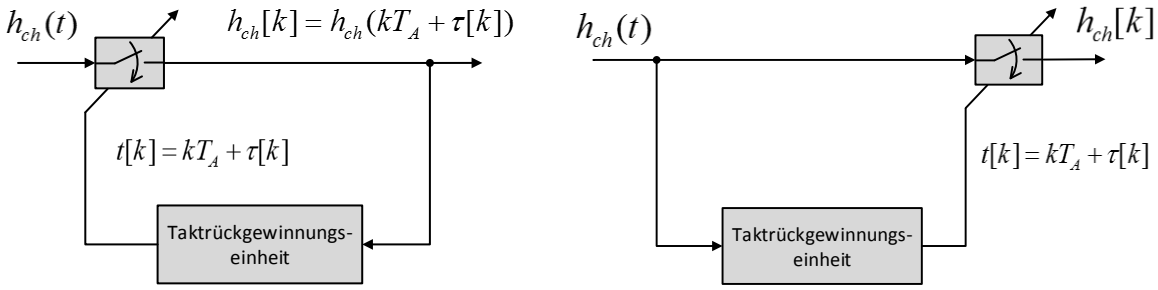


Abbildung 2.21: Ideale Abtastzeitpunkte für die steigende Flanke eines Rechtecktaktsignals

Die kanalspezifische Laufzeit wird in der Praxis nicht durch eine fixe Verzögerung ausgeglichen. Im Empfänger wird eine Taktrückgewinnungseinheit dem Entzerrer vorgeschaltet, welche sich auf das Empfangssignal synchronisiert.

3 Taktrückgewinnung

Taktrückgewinnung, oder auch Taktsynchronisierung, ist ein Prozess des Empfängers, welcher aus einem Empfangssignal den Sendetakt bestimmt, um somit zu den korrekten Zeitpunkten abzutasten. Wird zu den falschen Zeitpunkten abgetastet, so hat dies verherende Auswirkungen auf die Performance. Im vorigen Kapitel wurde beschrieben, dass die Gruppenlaufzeit des Kanals ausgeglichen werden muss. Hinzu kommt jedoch noch, dass das Taktsignal des Senders und des Empfängers in der Regel nicht zeitgleich beginnen. Daher der Begriff Synchronisierung. Es gibt vielzählige Möglichkeiten der Taktsynchronisation, welche in zwei Kategorien unterschieden werden können.



(a) Entscheidungsrückgekoppelte Methode (b) Entscheidungsvorwärtsgekoppelte Methode

Abbildung 3.1: Kategorien der Taktrückgewinnung

Entscheidungsrückgekoppelte Systeme arbeiten häufig mit verschiedenen Algorithmen zur Abtastfehler-Detektierung, welche eine schnelle Konvergenz des Abtastfehlers $\tau[k]$ aufweisen [11]. Eine Möglichkeit bietet Beispielsweise der Müller-Muller-Timing-Error-Detector (MM-TED). Dieser synchronisiert sich auf die Nullstellen der Kanalantwort $h_{ch}(t)$. Dies ist jedoch nur möglich, wenn die 1. Nyquist Bedingung erfüllt ist. Somit muss der Empfänger einen bereits voreingestellten Entzerrer beinhalten [5]. Aus diesem Grund wird in dieser Thesis die entscheidungsvorwärtsgekoppelte Methode genutzt, dessen Vorgehensweise im Weiteren erläutert wird.

3.1 Nichtlineare Spektralmethode

Aus Abschnitt 2.2.1 ging durch das Leistungsdichtespektrum des bipolaren NRZ-Codes sowie des AMI-Codes hervor, dass dort keine Sychronisationsinformation für die Taktrückgewinnung enthalten ist. Daher erfordert dies eine empfängerseitige Signalverarbeitung, um Spektralanteile bei der Taktfrequenz f_0 zu erhalten. Das dabei enstehende Signal könnte Ähnlichkeit mit dem unipolaren RZ-Code (return-to-zero) besitzen. Dieses besitzt durch das Formfilter einen Recheckimpuls welcher die Hälfte der Pulsbreite des NRZ-Codes besitzt, vgl. Abbildung 3.2 b).

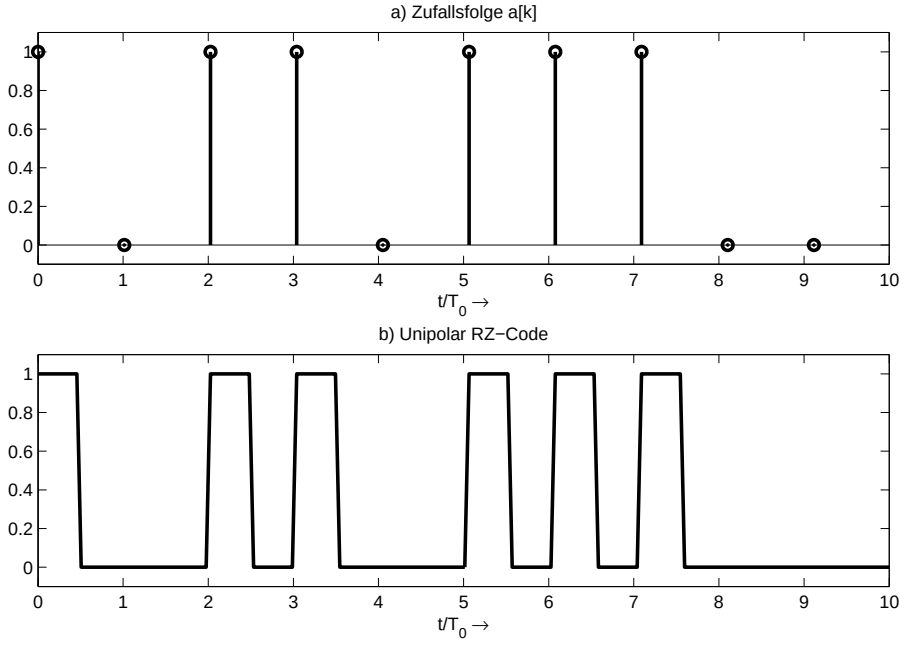


Abbildung 3.2: Unipolar RZ-Code

Die Impulsantwort des RZ-Formfilters ergibt sich zu

$$h_{FRZ}(t) = \frac{1}{2T_0} \cdot rect\left(\frac{t - T_0/4}{T_0/2}\right). \tag{3.1}$$

Dem Amplitudengang kann ein Durchlassbereich bei der Taktfrequenz f_0 entnommen werden.

$$|H_{FRZ}(f)| = |si(\pi f T_0/2)|\Big|_{f=f_0} = si(\pi/2) = 0,6366 \tag{3.2}$$

Dies spiegelt das mit der Welch-Methode geschätzte Leistungsdichtespektrum des unipolaren RZ-Codes, in Abbildung 3.3, wieder. Die Eingangsfolge wurde mit einem Hamming-Fenster gewichtet.

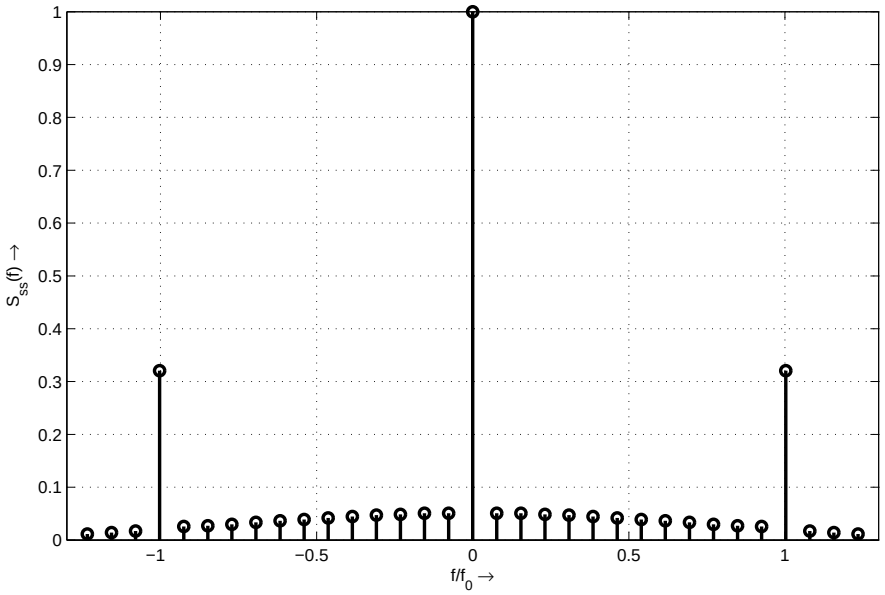


Abbildung 3.3: Leistungsdichtespektrum des unipolaren RZ-Codes

Auf den RZ-Code wird nicht genauer eingegangen. Es sollte lediglich gezeigt werden, dass eine solche Signalform zu einem diskreten Spektralanteil bei der Taktfrequenz führt. Das NRZ codierte Signal gilt es also so zu verändern, dass es dem des unipolaren RZ-Codes ähnelt. Dies kann in der Digitaltechnik mit einem einfachen Flankendetektor erreicht werden.

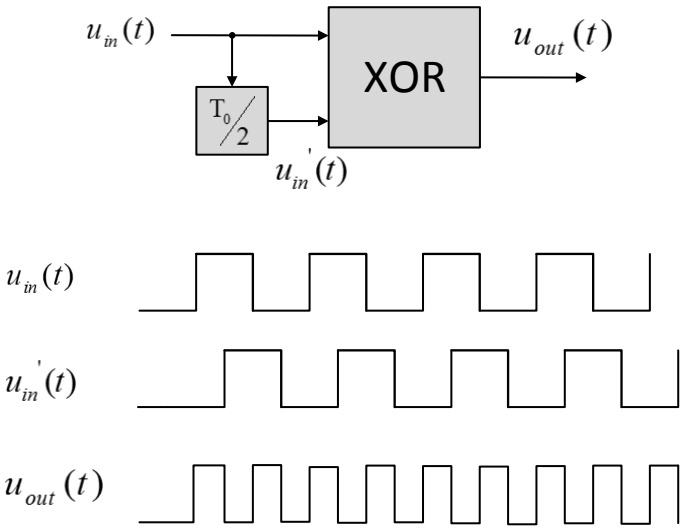


Abbildung 3.4: Flankendetektor mit einem xor-Logikgatter

Da die Kanalantwort jedoch starke Verzerrungen aufweist, muss eine andere Lösung gefunden werden. Eine Alternative bietet ein Hochpassfilter mit der Zeitkonstante $T = CR = \frac{1}{2\pi f_0}$. Dessen Rechteckantwort reagiert auf Änderungen der Eingangsflanke, vgl. Abbildung 3.5 a) und b). Da es sich bei einem Hochpass um ein LTI-System[1] handelt, können am Ausgang jedoch keine Frequenzen erscheinen, die nicht im Eingangssignal enthalten sind. Daher erfordert es eine nichtlineare Signalverarbeitung. Dies kann eine Betragsbildung oder Quadrierung sein. In Abbildung 3.5 c) ist die nichtlineare Operation mittels Quadrierung demonstriert. Die Amplituden sind auf eins normiert.

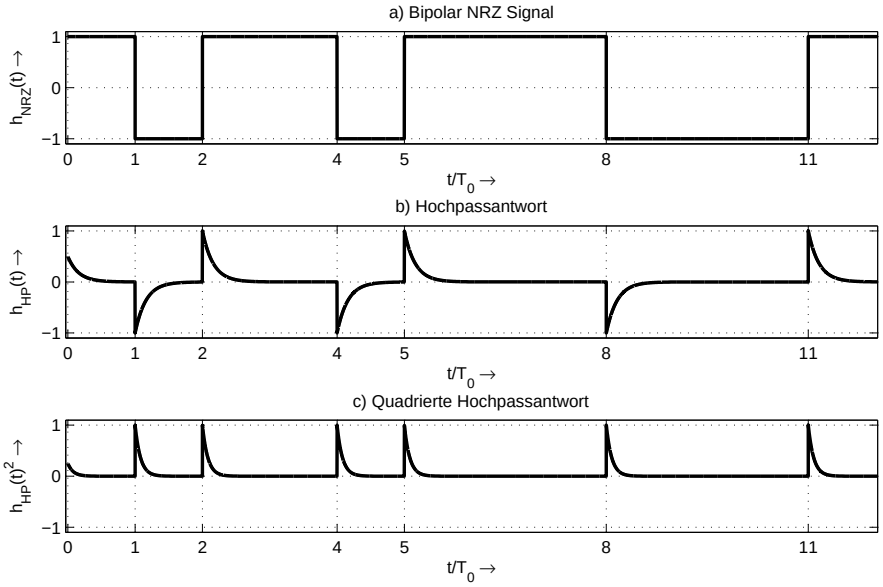


Abbildung 3.5: Nichtlineare Spektralmethode im Zeitbereich

Äquivalent zu Abbildung 3.5 sind die Leistungsdichtespektren in Abbildung 3.6 dargestellt. Diese sind auf die Frequenz f_0 und der Amplitude eins normiert. Deutlich zu erkennen ist die Wirkung des Hochpasses durch den herausgefilterten Gleichanteil. Abbildung 3.6 c) zeigt das Leistungsdichtespektrum der quadrierten Hochpassantwort. Diese stellt eine Faltung im Frequenzbereich mit sich selber dar. Grafisch kann dies verdeutlicht werden, indem das Spektrum aus Abbildung 3.6 b) schrittweise mit sich selbst verglichen wird. Ähneln sich die Spektren, so wird dies durch ein Peak gekennzeichnet. In Abbildung 3.6 c) ist ein Maximum bei $f = 0$, da hier das Spektrum, mit sich selbst verglichen, zu 100 % identisch ist. Mit Verschiebung um f_0 nach rechts oder links, ist die Ähnlichkeit nur noch teilweise gegeben, bei verschieben um $2f_0$ noch weniger. Dies ist nur eine grobe Umschreibung der Faltung

[1] In der Systemtheorie werden vornehmlich Systeme behandelt, die durch Linearität und Zeitinvarianz gekennzeichnet sind. Man spricht dabei von LTI-Systemen (engl. linear time-invariant) [5]

einer Funktion mit sich selbst, welche die Autokorrelation beschreibt. Erwartungsgemäß müsste in Abbildung 3.6 c) ein Ein- und Ausblendvorgang an den Spektrallinien erscheinen. Durch die verminderte Anzahl an FFT-Punken wird dies jedoch nicht dargestellt.

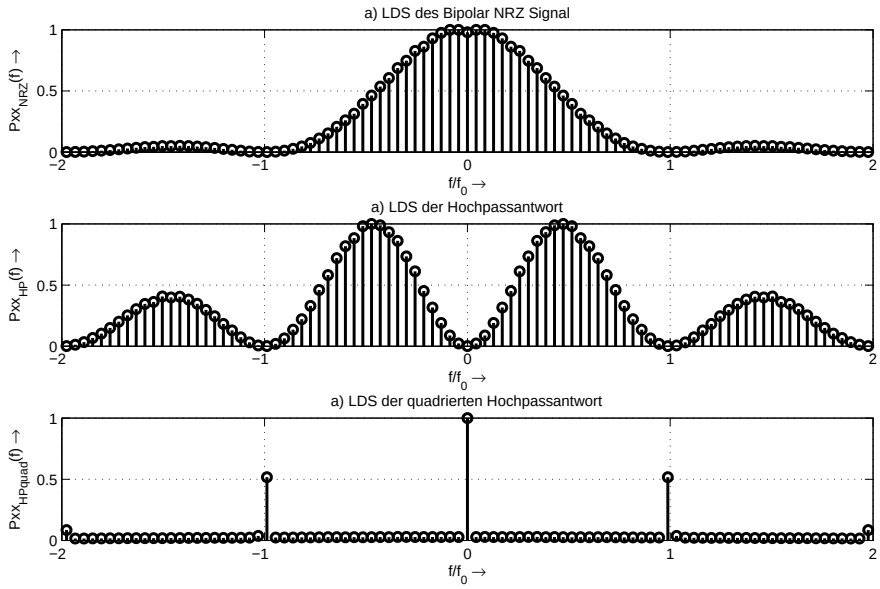


Abbildung 3.6: Nichtlineare Spektralmethode im Frequenzbereich

Da ein Hochpassfilter im Allgemeinen auch als Differenzierer bekannt ist, kann dieser als solcher zeitdiskret durch eine simple Differenzenbildung approximiert werden [12].

$$y(t) = \frac{dx(t)}{dt} \rightarrow y[n] = x[n] - x[n-1]) \tag{3.3}$$

Mit der z-Transformation ergibt sich die zeitdiskrete Übertragungsfunktion

$$H_{diff}(z) = \frac{Y(z)}{X(z)} = 1 - z^{-1}. \tag{3.4}$$

An diesem Punkt muss auf die Abtastfrequenz $T_A = 1/f_A$ des ADC-Boards eingegangen werden. Das verwendete ADC-Board, welches im Kapitel 5 näher beschrieben wird, besitzt zwei ADC. Ein ADC wird benötigt um den Kanalausgang zu digitalisieren, der andere wird für das Audiosignal benötigt (in Kapitel 4 mehr dazu). Diese arbeiten hardwarebedingt jedoch beide mit der selben Taktdomain. Dies erzwingt, dass die höchste beider erforderlichen Taktfrequenzen als einheitlicher Takt genutzt werden muss. Diese beträgt $f_A = 11 \cdot f_0$, welche im Kapitel 4 erläutert wird. Die erhöhte Abtastfrequenz führt dazu, dass die Implementierung

des Differenzierers nach Gleichung 3.3 nicht ratsam wäre. Ist dem Eingangssignal $x[n]$ um die Abtastfrequenz $f_A/2$ bandbegrenztes weißes Rauschen überlagert, so geht dieses nahezu vollständig mit ins Ausgangssignal $y[n]$ ein. Dies wird bei Betrachtung des Differenzierer-Amplitudenganges $|H_{diff}(f)|$ in Abbildung 3.7 a) deutlich.

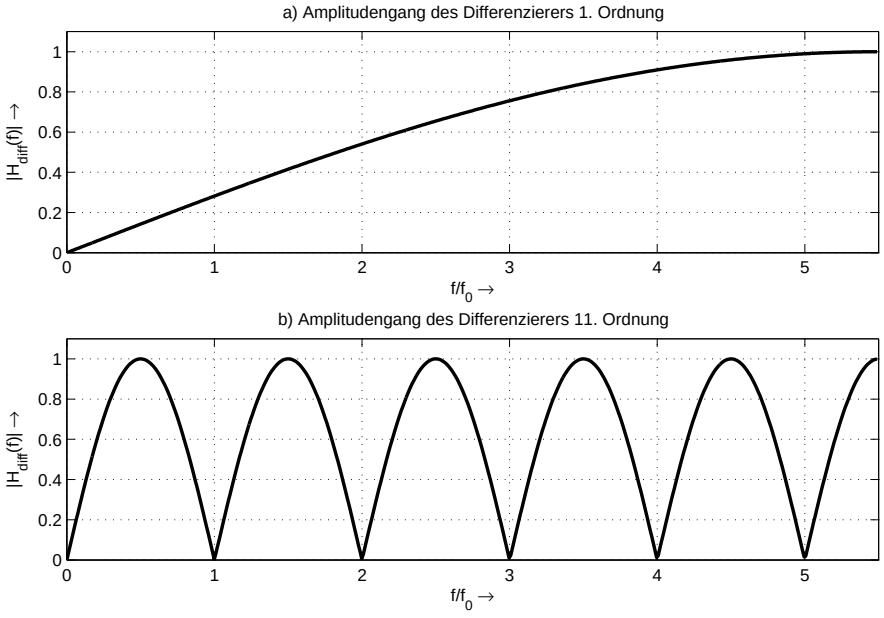


Abbildung 3.7: Differenzierer Amplitudengänge

Durch einen Differenzierer höherer Ordnung kann der Amplitudengang einfach beeinflusst werden, vgl. Abbildung 3.7 b). Dieser ergibt sich, wenn in Gleichung 3.3 die Differenz statt über $1 \cdot T_A$, über $11 \cdot T_A$ gebildet wird. So entsteht eine Art Kammfilter, welcher Nullstellen im Abstand f_0 besitzt und somit eine verbesserte Störunterdrückung bietet.

Im Leistungsdichtespektrum des Empfangssignals wurde damit Taktinformation erzeugt. Nun ist es weitere Aufgabe des Empfängers, sich auf diese Taktinformation zu synchronisieren. Eine einfache Methode scheint eine schmalbandige Bandpassfilterung zu sein, um die Taktinformation zu isolieren. Dies bietet jedoch keine geeignete Lösung, da bei langen Null- oder Einsfolgen das Filter auschwingt und somit der Takt komplett wegfällt. Auf Grund dessen bedarf es einer anderen Methode.

3.2 Phase Locked Loop (PLL)

Eine Phasenregelschleife, im Weiteren Phase Locked Loop oder kurz PLL genannt, bietet eine in der Praxis häufig angewandte Methode der Taktrückgewinnung [6]. Das Prinzipblockschaltbild einer Phase Locked Loop ist in Abbildung 3.8 dargestellt.

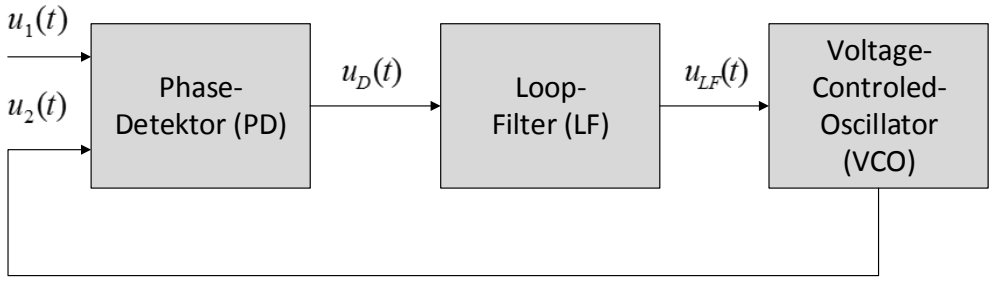


Abbildung 3.8: Blockschaltbild des Phase Locked Loop

Eine Phase Locked Loop besteht aus einem Phasendetektor (auch Phasendiskriminator genannt), dessen Ausgangssignal $u_D(t)$ im wesentlichen proportional zur Phasendifferenz $\varphi_1(t) - \varphi_2(t)$ ist. Dem Schleifenfilter (engl. Loopfilter) und einem spannungsgesteuerten Oszillator (engl. Voltage Controlled Oscillator), dessen Frequenz vom Loopfilter Ausgangssingal $u_{LF}(t)$ gesteuert wird. Um das Verhalten der PLL zu erläutern werden für die Eingangsignale harmonischen Schwingungen

$$u_1(t) = \hat{U}_1 \cdot sin(\omega_0 t + \varphi_1(t)) \tag{3.5}$$

$$u_2(t) = \hat{U}_2 \cdot cos(\omega_0 t + \varphi_2(t)) \tag{3.6}$$

angenommen. $u_1(t)$ stellt dabei das Referenzsignal (regelungstech. Führungsgröße) dar, auf welches sich das Signal $u_2(t)$ (regelungstech. Regelgröße), vom VCO, synchronisieren soll. Eine Möglichkeit, einen Phasendetektor zu realisieren, ist ein Multiplizierer. Mit der Produktregel der trigonometrischen Funktionen, ergibt sich das Phasendetektorsignal zu

$$u_D(t) = u_1(t) \cdot u_2(t) = \frac{\hat{U}_1 \hat{U}_2}{2} \cdot \{sin[\varphi_1(t) - \varphi_2(t)] + sin[2\omega_0 t(\varphi_1(t) + \varphi_2(t))]\}. \tag{3.7}$$

Der zweite Sinus-Therm, aus Gleichung 3.7, ist eine reine Störgröße, welche mit dem Loopfilter herausgefiltert werden muss, um eine phasenabhängige Ansteuerung des VCO zu erhalten. Das Loopfilter Ausgangssignal stellt die Stellgröße dar, welche dann idealerweise aus dem ersten Therm aus Gleichung 3.7 besteht. Der sogenannte eingerastete Zustand der

PLL ergibt sich für eine Phasendifferenz von $\varphi_1(t) - \varphi_2(t) = 0$. Hier wird der Sinus zu Null. Das Regelverhalten des Phasendetektors zeigt Abbildung 3.9.

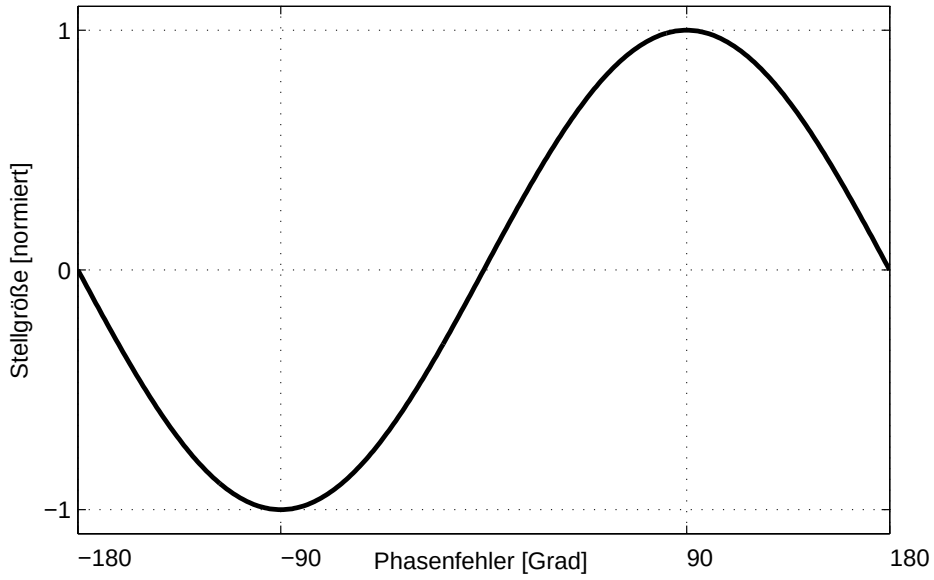


Abbildung 3.9: Regelverhalten eines Multiplizierers als Phasendetektor

Die Nichtlinearität des Regelverhaltens ist nachteilig. Denn auf große Phasenfehler soll mit einer großen Stellgröße reagiert werden. Daher beschränkt sich die Betrachtung dieses Phasendetektor nur auf den linearen Bereich von $\pm 90°$. Man spricht deshalb auch vom linearen PLL [13]. Ist die Stellgröße $u_{LF}(t)$ gleich Null, so schwingt der VCO auf seiner Mittenfrequenz f_0. Hier ergibt sich bereits der Vorteil gegenüber einer Bandpassfilterung zur Taktrückgewinnung. Während das Bandpassfilter bei Ausbleiben der Taktinformation auschwingt, schwingt der VCO weiter auf seiner Mittenfrequenz. Für weitere Betrachtungen wird das mathematische Modell eines PLL im Laplacebereich aufgestellt, vgl. Abbildung 3.10.

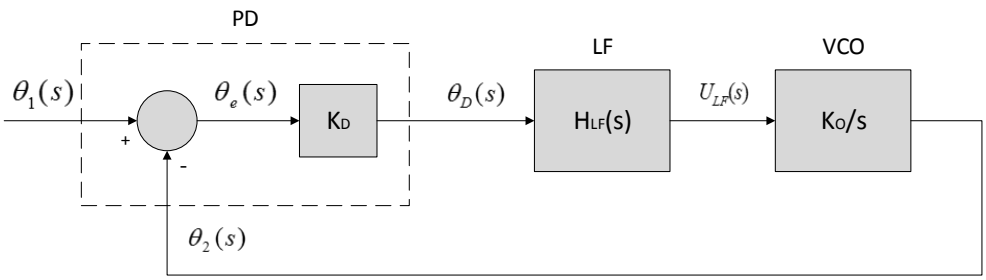


Abbildung 3.10: Mathematisches Modell eines Phase Locked Loop

Hieraus geht das Verhalten eines Integrierers für den VCO hervor. Dies ist dadurch begründet, dass für die Phase gilt

$$\varphi_2(t) = K_O \int_0^t u_{LF}(\tau) d\tau. \tag{3.8}$$

Mit der Laplacetransformation erhält man daraus

$$\varphi_2(t) \circ\!\!-\!\!\bullet\, \Theta_2(s) = U_{LF}(s) \frac{K_O}{s} \tag{3.9}$$

Das Simulinkmodell des VCO zeigt Abbildung 3.11.

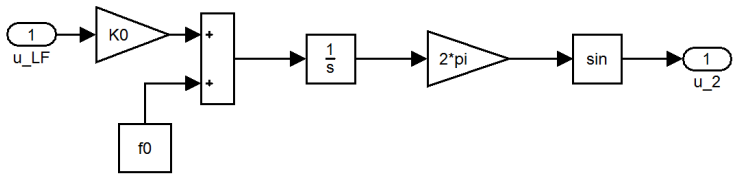


Abbildung 3.11: Simulinkmodell des VCO

Aus dem mathematischen Modell heraus können die Übertragungsfunktion des offenen Regelkreises

$$H_o(s) = \frac{K_D K_O}{s} \cdot H_{LF}(s) \tag{3.10}$$

und des geschlossenen Regelkreises

$$H_C(s) = \frac{\Theta_2(s)}{\Theta_1(s)} = \frac{K_D K_O \cdot H_{LF}(s)}{s + K_D K_O \cdot H_{LF}(s)} \tag{3.11}$$

definiert werden. K_D und K_O stellen dabei einfache Verstärkungsfaktoren dar, welche im Folgenden als $K = K_D \cdot K_O$ zusammengefasst werden. Da der Phasendetektor eine im Vergleich zur Stellgröße höherfrequente Störgröße verursacht, muss das Loopfilter tiefpasscharakter haben. Zunächst wird von einem passivem Tiefpass 1. Ordnung ausgegangen.

$$H_{LF}(s) = \frac{1 + T_2 s}{1 + T_1 s} \tag{3.12}$$

Das Loopfilter enthält eine Nullstelle im Zähler. Diese bestimmt den Dämpfungsfaktor der PLL und trägt damit maßgeblich zum Regelverhalten bei. Dies wird bei Betrachtung des

geschlossenen Regelkreises deutlich. Mit Gleichung 3.12 und Gleichung 3.11 ergibt sich

$$H_C(s) = \frac{\frac{KT_2}{T_1}s + \frac{K}{T_1}}{s^2 + \left(\frac{K}{T_1T_2} + \frac{1}{T_1}\right)s + \frac{K}{T_1}}. \tag{3.13}$$

Gleichung 3.13 stellt eine PLL 2. Ordnung dar. Generell kann gesagt werden, dass die Ordnung einer PLL, die des Loopfilters um eins inkrementiert ist. In der Regelungstechnik ist es üblich, Übertragungsfunktionen in eine normalisierte Form zu bringen, sodass der Nenner die Form $s^2 + 2\zeta\omega_r s + \omega_r^2$ aufweist [13]. Der Parameter ω_r ist dabei die Eigenfrequenz und ζ der Dämpfungsfaktor.

$$H_C(s) = \frac{\left(2\zeta\omega_r - \frac{1}{K}\right)s + \omega_r^2}{s^2 + 2\zeta\omega_r s + \omega_r^2} \tag{3.14}$$

Mit

$$\omega_r = \sqrt{\frac{K}{T_1}} \tag{3.15}$$

$$\zeta = \frac{\omega_r}{2}\left(T_2 + \frac{1}{K}\right) \tag{3.16}$$

Für sogenannte high-gain loops, für die $K >> \omega_r$ gilt, kann die Vereinfachung

$$H_C(s) = \frac{2\zeta\omega_r s + \omega_r^2}{s^2 + 2\zeta\omega_r s + \omega_r^2} \tag{3.17}$$

für den geschlossenen Regelkreis gemacht werden [13]. Abbildung 3.12 zeigt den Phasenfrequenzgang für unterschiedliche Dämpfungfaktoren. Dieser hat großen Einfluss auf das Einschwingverhalten einer PLL. Bei zu kleinem ζ entstehen bei Phasensprüngen starke Überschwinger, hingegen ist bei zu großem ζ die Reaktionszeit des Systems sehr langsam. In der Regel wird ein Dämpfungsfaktor von $\zeta = 0.707$ gewünscht, da dieser Wert einen guten Kompromiss vom Überschwingen zur Einschwingdauer darstellt.

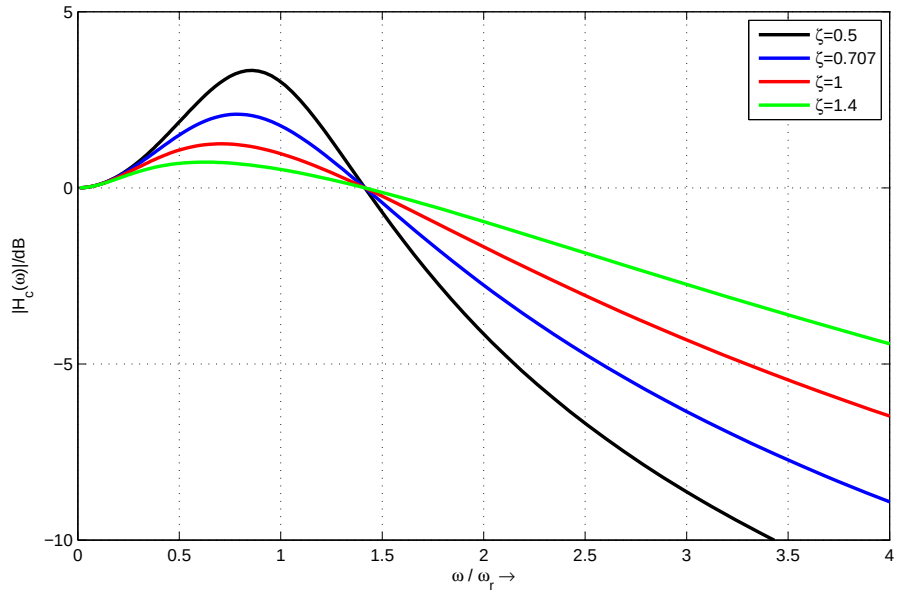


Abbildung 3.12: Phasenfrequenzgang der linearen PLL mit Variieren des Dämpfungsfaktor

Im Folgenden wird die PLL dimensioniert. Hierfür wird nach einer Methode von Roland Best vorgegangen und darauf referenziert [13]. Ausschlaggebend ist das Loopfilter, von denen vier unterschiedliche betrachtet werden. Zunächst muss die Kreisfrequenz bestimmt werden, bei welcher der Regelkreis eine Dämpfung von 3 dB aufweist. Diese wird häufig um den Faktor $1/20$ der Taktkreisfrequenz ω_0 gewählt. Für eine PLL 2. Ordnung kann gezeigt werden, dass für einen Dämpfungsfaktor von $\zeta = 0.707$ die 3 dB Grenzkreisfrequenz

$$\omega_{3dB} \approx 2,06 \cdot \omega_r \tag{3.18}$$

beträgt. Es kann weiter gezeigt werden, dass

$$\omega_2 \approx 1,55 \cdot \omega_r \tag{3.19}$$

ist. Dieses kennzeichnet die Kreisfrequenz bei welcher der Amplitudengang des offenen Regelkreises die 0 dB Grenze passiert. Das kennzeichnet eine Nullstelle im System. Nach Umstellung kann mit Gleichung 3.18 und 3.19, für

$$\omega_2 \approx \frac{\omega_{3dB}}{1,33} \tag{3.20}$$

festgelegt werden. Damit kann die Zeitkonstante T_2 eines Loopfilters 1. Ordnung berechnet werden zu

$$T_2 = \frac{1}{\omega_2}. \tag{3.21}$$

Die Zeitkonstante T_1 wird nach der Vorschrift

$$T_1 = \frac{1}{\omega_1} \tag{3.22}$$

berechnet, wobei gilt

$$\omega_1 = \frac{\omega_2^2}{K}. \tag{3.23}$$

Der Verstärkungsfaktor K wird dabei so eingestellt, dass weiterhin $\zeta = 0.707$ eingehalten wird, da K, nach Gleichung 3.16, Einfluss auf den Dämpfungsfaktor hat. Loopfilter n-ter Ordnung weisen n-Polstellen auf. Somit ergibt sich die Übertragungsfunktion zu

$$H_{LF}(s) = \frac{T_2 s + 1}{(T_1 s + 1)(T_3 s + 1)(T_4 s + 1)...(T_n + 1)} \tag{3.24}$$

Die Parameter für Loopfilter höherer Ordnung werden nach der Vorschrift

$$T_n = \frac{1}{\omega_n}, n = 3, 4, 5... \tag{3.25}$$

berechnet, wobei gilt

$$\omega_n = 5 \cdot \omega_{n-1}, n = 3, 4, 5... \tag{3.26}$$

Da jedoch bereits zu niedrigeren Frequenzen eine höhe Dämpfung gefordert ist, wird $\omega_n = \omega_{n-1}$ gewählt. Nach diesen Vorschriften werden die Loopfilter dimensioniert. Tabelle 3.1 listet die Parameter der PLL auf.

Ordnung	T_1	T_2	T_3	T_4	T_5	K
1.	$0,0017s$	$66,15\mu s$				380000
2.	$61,88\mu s$	$47,25\mu s$	$47,25\mu s$			27720
3.	$49,41\mu s$	$47,25\mu s$	$23,62\mu s$	$23,62\mu s$		22134
4.	$26,75\mu s$	$37,80\mu s$	$18,90\mu s$	$18,90\mu s$	$18,90\mu s$	18720

Tabelle 3.1: PLL Parameter

In Abbildung 3.13 sind die Amplitudengänge verschiedener Loopfilter dargestellt. Dabei weist das Loopfilter 1. Ordnung gegenüber den anderen eine deutlich niedrigere Grenzfrequenz auf. Dies liegt daran, dass ein hoher Verstärkungsfaktor benötigt wird um eine Dämpfung von $\zeta = 0.707$ einzuhalten. Wird dem Filter eine Polstelle hinzugefügt, so steigt die Resonanzfrequenz der PLL und der Dämpfungsfaktor sinkt. Daraus folgt, dass der Verstärkungsfaktor der PLL kleiner werden muss. Dies führt schließlich zu einer höheren Grenzfrequenz des Loopfilters.

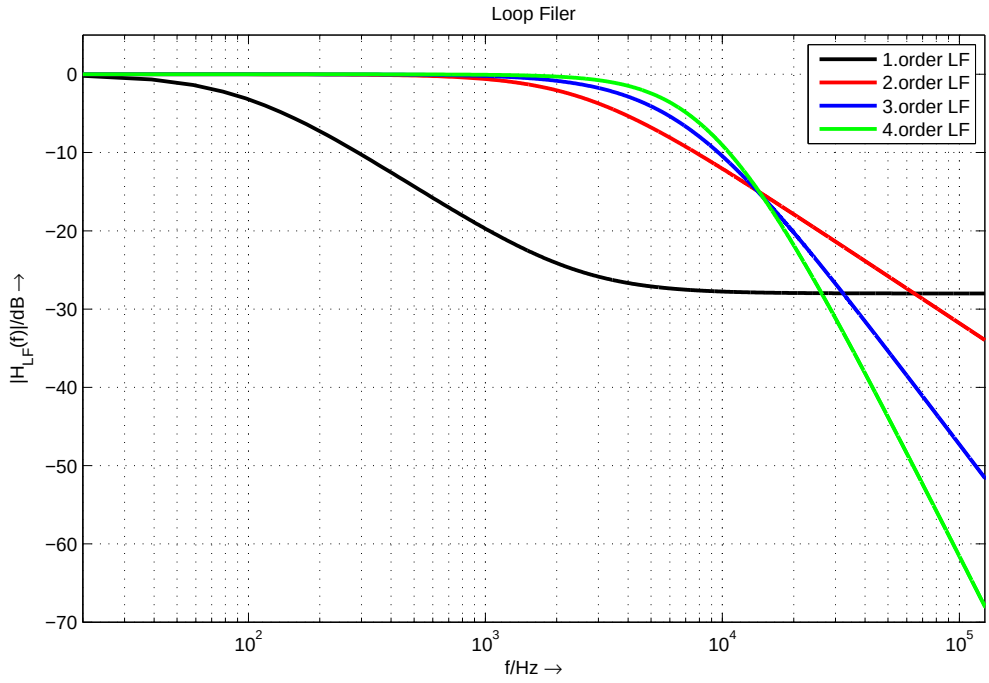


Abbildung 3.13: Amplitudengang verschiedener Loopfilter

Das Loopfilter soll eine möglichst hohe Dämpfung der Spektralanteile bei der doppelten Taktfrequenz ($2f_0 = 128\ kHz$) besitzen. Das Loopfilter 1. Ordnung hat durch die Nullstelle eine begrenzte Dämpfung. Dies ist zudem nachteilig, wenn dem Referenzsignal $u_1(t)$ zusätzlich Rauschen überlagert ist. Denn durch das Loopfilter muss dieses möglichst gut unterdrückt werden. Mit dem Hinzufügen einer Polstelle wird eine Dämpfung von 20 dB/Dekade weiterhin gewährleistet. Wird das Filter um eine weitere Polstelle erweitert, so steigt die Dämpfung auf 40 dB/Dekade. Das Loopfilter 4. Ordnung erzielt somit eine Dämpfung von 60 dB/Dekade.

Abbildung 3.14 zeigt die Amplitudengänge des Phasenübertragungsverhalten der PLL mit unterschiedlichen Loopfiltern. Wie zuvor bereits erwähnt, wird ein Dämpfungsfaktor von $\zeta = 0.707$ stets eingehalten. Der geschlossene Regelkreis besitzt eine zusätliche Polstelle, welche zu einem stärkerem Abfall von zusätzlich 20 dB/Dekade führt.

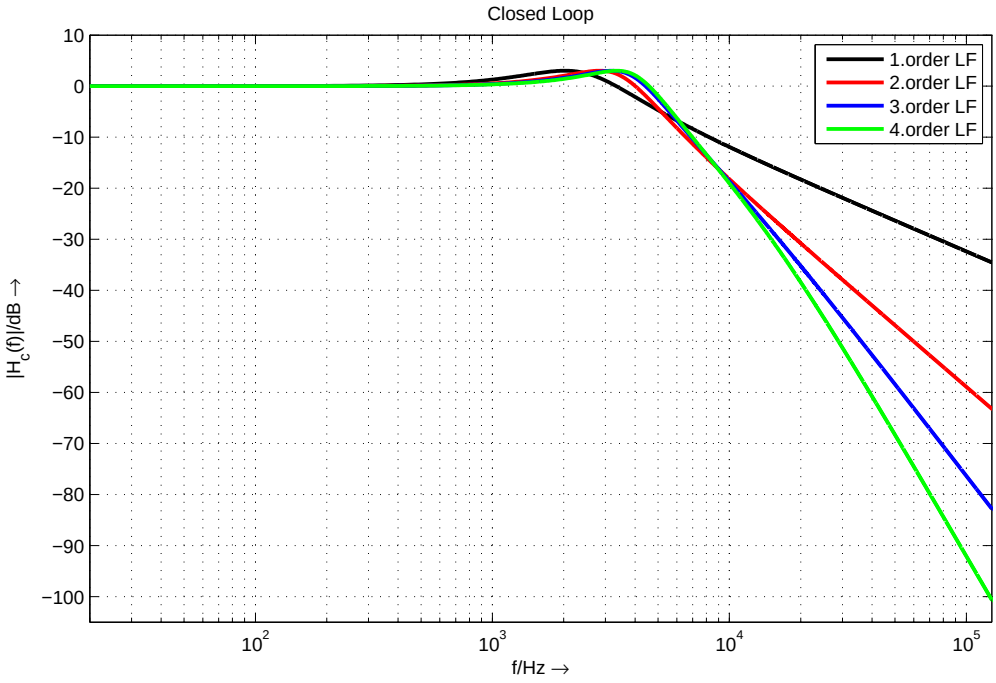


Abbildung 3.14: Phasenfrequenzgänge der geschlossenen Regelkreise

Die dimensionierten PLL unterschiedlicher Ordnung sollen nun in Simulink simuliert werden. Um die Loopfilter zeitdiskret realisieren zu können, bedarf es einer Transformation der zeitkontinuierlichen Übertragungsfunktion. Hierfür wird die Methode der bilinearen Transformation verwendet. Diese bietet anderen Methoden gegenüber den Vorteil, Alias-Effekte im zeitdiskreten System zu vermeiden. Die Systemfunktion $H_{LF}(s)$ korrespondiert mit der zeitdiskreten Systemfunktion $H_{LF}(z)$ durch Substitution der Variablen

$$s = \frac{2}{T_A}\frac{1 - z^{-1}}{1 + z^{-1}}. \tag{3.27}$$

Mit Gleichung 3.27 in Gleichung 3.12 ergibt sich ein diskretes Loopfilter 1. Ordnung zu

$$H_{LF}(z) = \frac{2T_2 + T_A + (T_A - 2T_2)z^{-1}}{2T_1 + T_A + (T_A + 2T_2)z^{-1}.} \tag{3.28}$$

Eine bilinear Transformation in Matlab bietet die Funktion *„bilinear“*. Das Modell des Senders, des Kanalmodelles sowie der Abtaster des Empfängers, ist in Abbildung 3.15 abgebildet. Simuliert wird mit einem 5RC-Kanalmodell und dem AMI-Leitungscode.

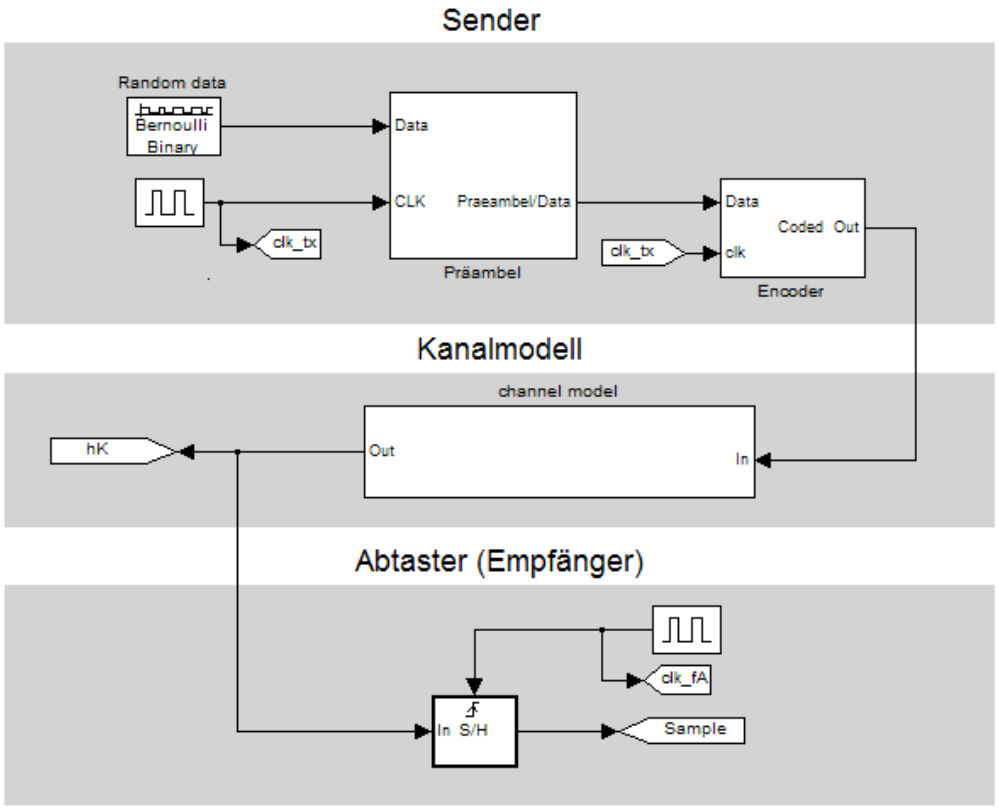


Abbildung 3.15: Simulink Modell des Senders, Kanalmodell und Abtaster

Hierbei ist zu erwähnen, dass zu Beginn der Übertragung eine Präambel[2] gesendet wird. Bei welcher sich auf das Ethernet-Protokoll bezogen wird. Diesem wird eine Präambel der Form "101010...1011", und der Länge von 8 Byte angefügt [14]. Dies soll für ein schnelles Einrasten der PLL sorgen. Im Anschluss an die Präambel folgen Zufallszahlen. Die Taktrückgewinnungseinheit ist in Abbildung 3.16 abgebildet.

[2]Eine Präambel ist in der Nachrichtentechnik eine Einleitung vor den eigentlichen Informationsdaten, in Form einer senderseitig festgelegten Trainingsfolge

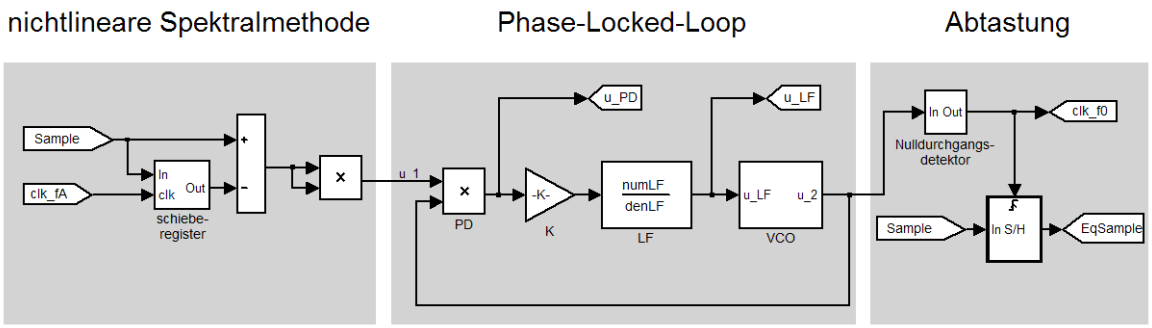


Abbildung 3.16: Simulink Modell der Taktrückgewinnungseinheit

Zunächst wird das Phasendetektor-Ausgangssignal $u_{PD}(t)$ analysiert. Dies ist mit einer PLL 2. Ordnung simuliert, vgl. Abbildung 3.17 a). Für die Dauer der Trainingssequenz von $t/T_0 = 0...64$ kann dem Phasendetektorausgang ein Einschwingvorgang der PLL entnommen werden. Da nach der Trainingssequenz $(t/T_0 > 64)$ die Zufallsfolge beginnt, besitzt der Phasendetektorausgang bei längeren Nullfolgen Unregelmäßigkeiten.

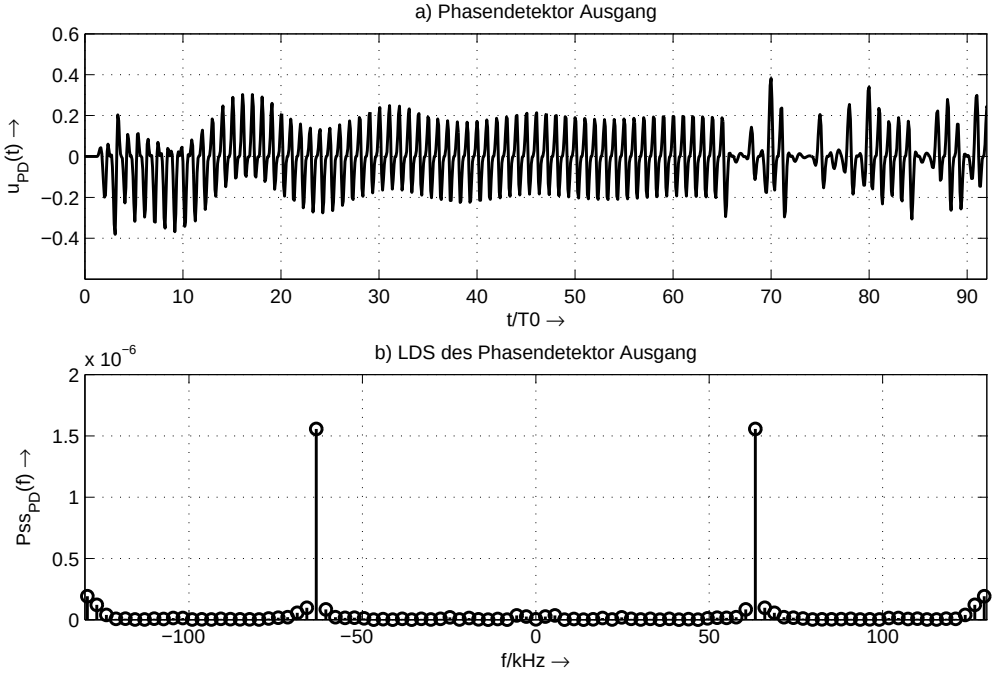


Abbildung 3.17: Ausgangssignal und LDS des Phasendetektors

Das Leistungsdichtespektrum zeigt Spektralanteile an der Taktfrequenz f_0 sowie $2f_0$. Diese stellen Störgrößen dar, welche herausgefiltert werden müssen. Die Stellgröße muss idealerweise nur aus einem Gleichanteil bestehen, womit der VCO angesteuert wird.

In Abbildung 3.18 ist das Ausgangssignal des Loopfilters 1. Ordnung, sowie dessen Leistungsdichtespektrum dargestellt. Dadurch dass der Verstärkungsfaktor K vor dem Loopfilter geschaltet ist, bestitzt der Loopfilterausgang die Einheit V/Hz bzw. normiert auf 1 V die Einheit Hz^{-1}. Dieses stellt somit die Frequenzauslenkung, bzw. den Frequenzhub des VCO dar.

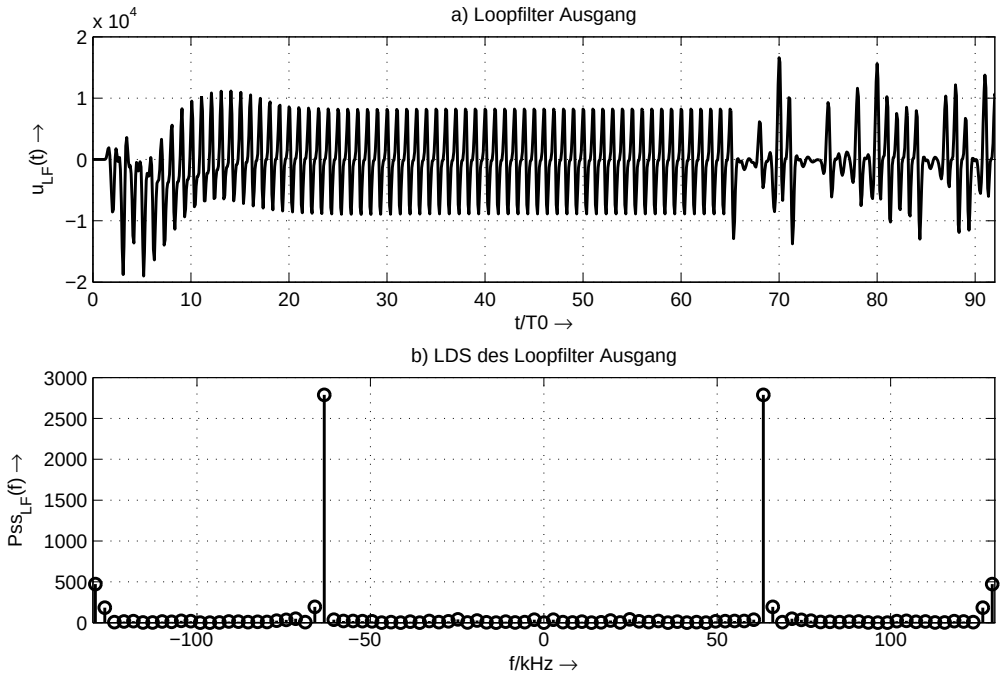


Abbildung 3.18: Ausgangssignal und LDS des Loopfilters 1. Ordnung

Zu erkennen ist, dass der Spitze-Spitze Wert ziemlich groß ist. Dieser führt mit Werten weit über $\pm 1,5 \cdot 10^4 Hz$ enorm zum Jittern[3] des Taktes bei. Das Leistungsdichtespektrum zeigt eine ungenügende Filterung der Störfrequenzen. Eine Verbesserung zeigt das Loopfilter 2. Ordnung, vgl. Abbildung 3.19.

[3] Jitter (engl. für Schwankung) bezeichnet zeitliches schwanken/zittern eines Taktes bei der Übertragung von Digitalsignalen

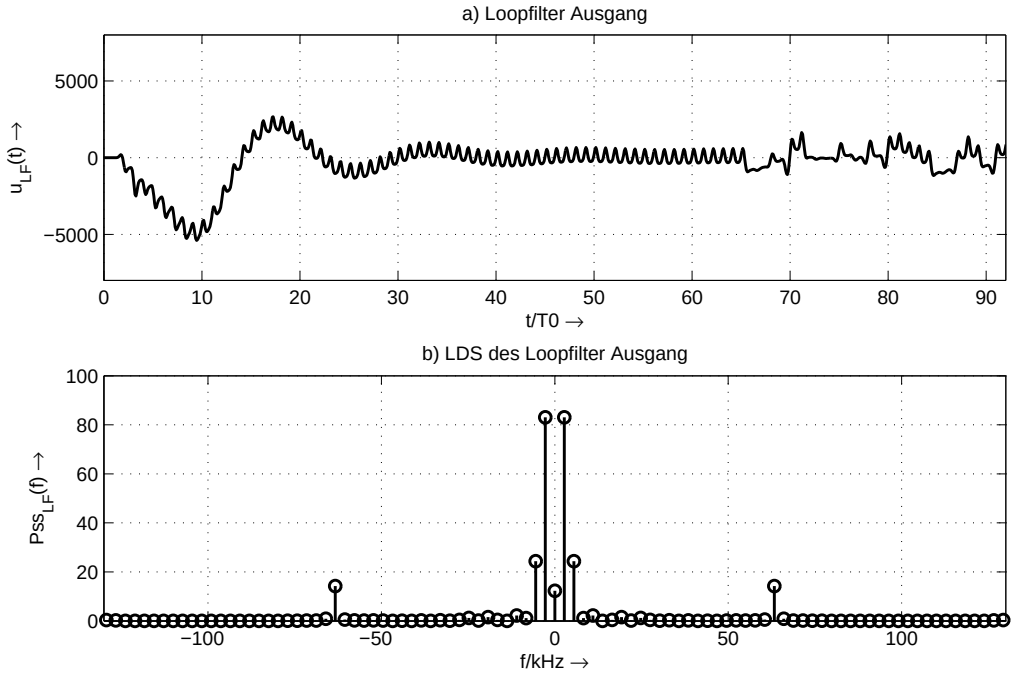


Abbildung 3.19: Ausgangssignal und LDS des Loopfilters 2. Ordnung

Dem Ausgangssignal u_{LF} ist bereits ein deutliches Regelverhalten mit der Eigenfrequenz f_r der PLL zu entnehmen. Der Spitze-Spitze Wert und damit auch der Frequenzhub des VCO ist deutlich kleiner gegenüber dem des Loopfilter 1. Ordnung. Das Leistungsdichtespektum zeigt zudem, dass die Störfrequenz bei $2f_0$ nicht mehr vorhanden zu sein scheint. Frequenzanteile bei f_0 sind ebenso merklich gedämpft. Zu niedrigen Frequenzen hin sind die stärksten Spektralanteile vorhanden. Diese resultieren aus dem Einschwingvorgang mit der Eigenfrequenz der PLL.

Das Loopfilter 3. Ordnung führt zu einem noch besseren Ergebnis, welches auch das Leistungsdichtespektrum wiederspiegelt.

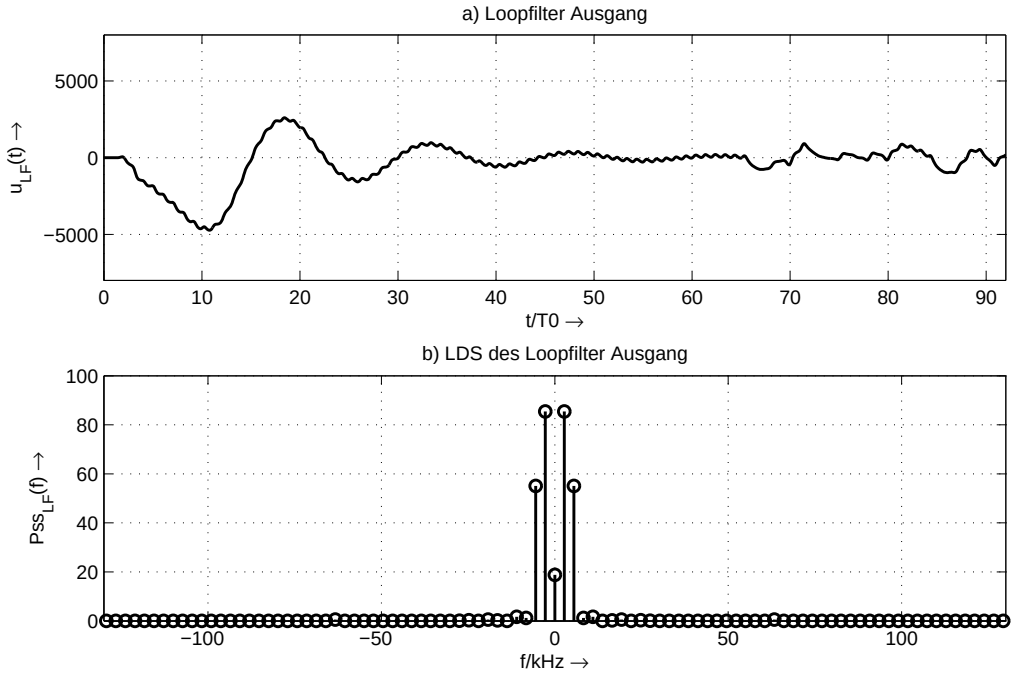


Abbildung 3.20: Ausgangssignal und LDS des Loopfilters 3. Ordnung

Hier ist lediglich die Störfrequenz f_0 noch zu erahnen. Im Bezug auf den Frequenzhub kann gesagt werden, dass es nur noch eine minimale Verbesserung zu geben scheint. Der zeitliche Verlauf zeigt dass die Unregelmäßigkeiten nach der Trainingssequenz sich nicht vollständig beseitigen lassen. Demnach muss ein konstantes Jittern in Kauf genommen werden.

Das Loopfilter 4. Ordnung scheint schließlich kaum noch eine Verbesserung zu bewirken, vgl. Abbildung 3.21. Aus diesem Grund werden nachfolgende Untersuchungen mit dem Loopfilter 3. Ordnung durchgeführt.

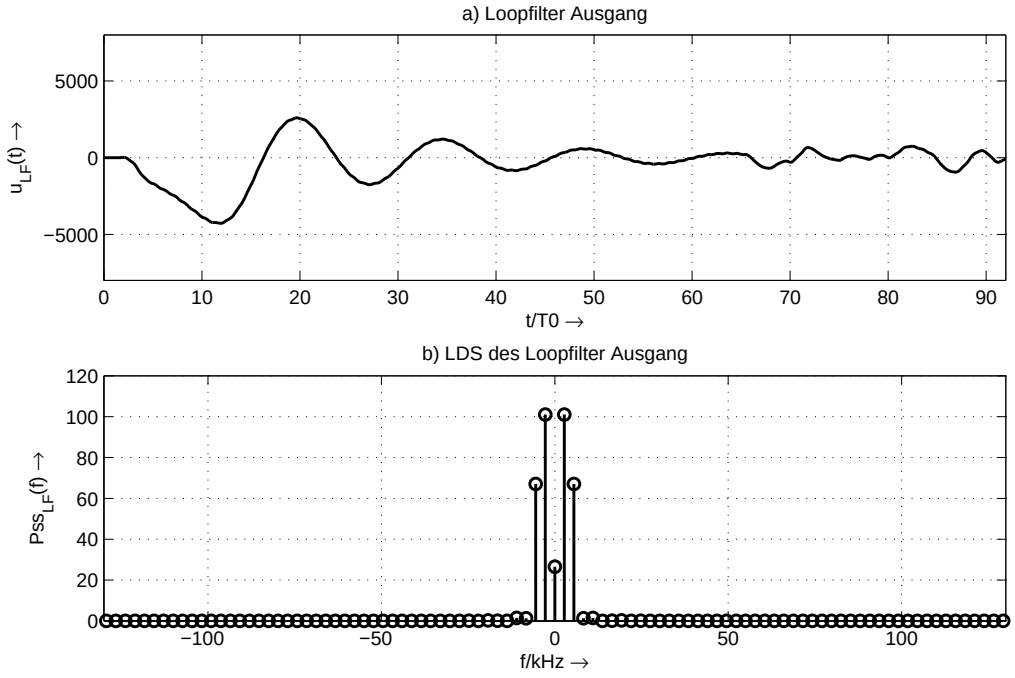


Abbildung 3.21: Ausgangssignal und LDS des Loopfilters 4. Ordnung

Zu den Simulationen der Loopfilter ist noch zu erwähnen, dass der Verstärkungsfaktor K zuvor um einen weiteren Faktor $K = K \cdot 2.8$ ergänzt wurde. Dieser Wert wurde empirisch ermittelt und hat den Grund, dass das Regelverhalten für verschiedene Kanalmodelle konvergieren soll. Da die Simulationen am 5RC-Kanalmodell durchgeführt wurden, welcher weniger Dämpfung im Vergleich zum 7RC-Kanalmodell hat, weist hier das Regelverhalten längere Einschwingphasen auf. Der Dämpfungsfaktor ζ ist demnach kleiner. Abbildung 3.22 zeigt die Simulation mit dem 7RC Tiefpass- und Hochpass-Kanalmodell.

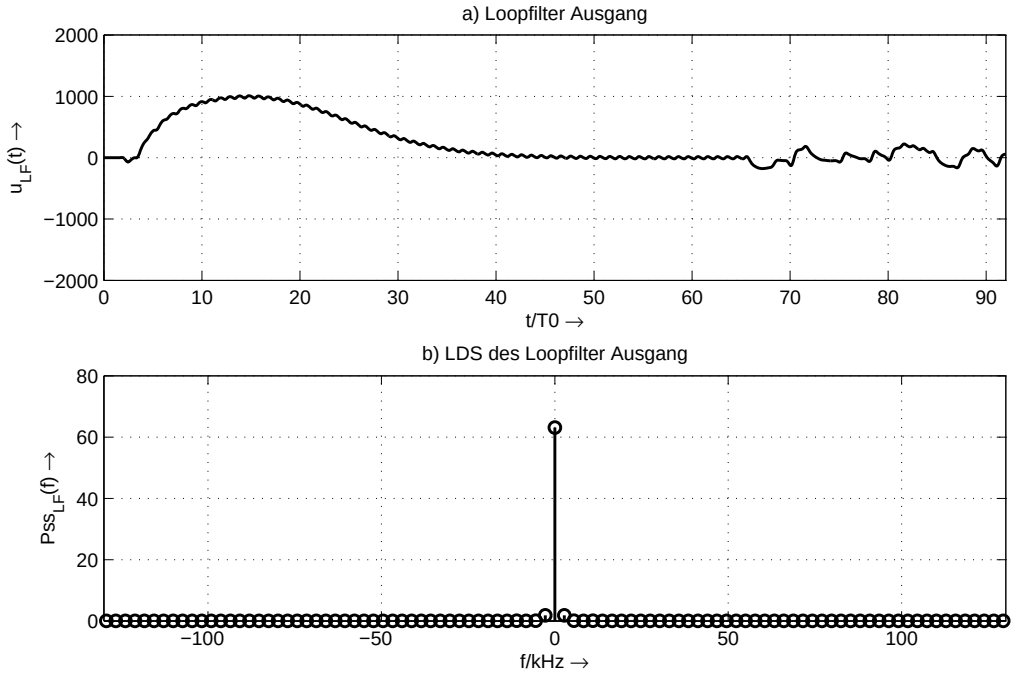


Abbildung 3.22: Ausgangssignal und LDS des Loopfilters 3. Ordnung bei einem 7RC- mit HP-Kanalmodell

Das Loopfilterausgangssignal konvergiert langsamer, was auf einen größeren Dämpfungsfaktor ζ hinweist. Die Unregelmäßigkeiten bei längeren Nullfolgen für $t/T_0 > 64$ ist deutlich geringer als beim 5RC-Kanalmodell. Daraus lässt sich schließen, dass das 7RC-Kanalmodell weniger konstanten Jitter erzeugt.

Da bei einer PLL die Stabilität eine maßgebliche Rolle spielt, wird diese untersucht. Dieses kann auf unterschiedliche Weise geprüft werden. Das Pol-Nullstellendiagramm von Regelkreisen bietet eine Möglichkeit. Befinden sich die Polstellen in der linken s-Ebene, so gilt ein Regelkreis im Allgemeinen als stabil [15]. In der z-Ebene hingegen müssen die Polstellen innerhalb des Einheitskreises liegen.

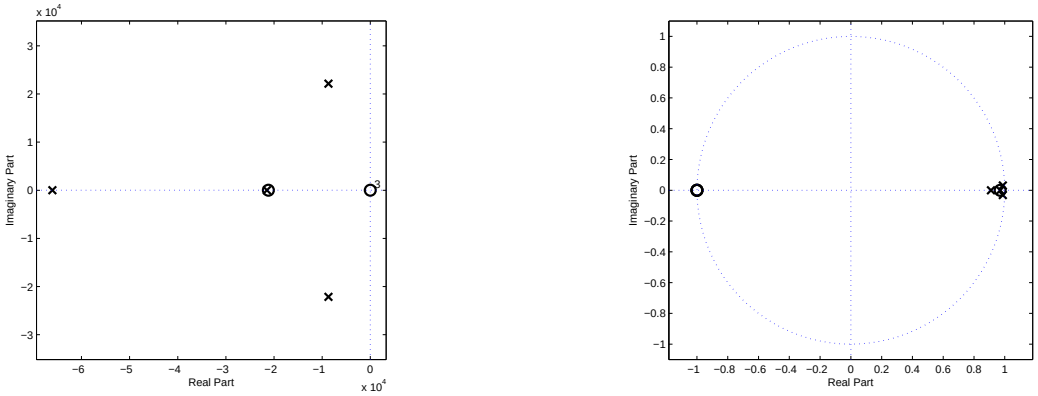
Die Übertragungsfunktion des geschlossenen Regelkreises 4. Ordnung ist gegeben durch

$$H_C(s) = \frac{Kb_1 s + Kb_0}{s^4 + a_2 s^3 + a_1 s^2 + (K \cdot b_1 + a0)s + K \cdot b_0} \tag{3.29}$$

mit den Koeffizienten

$$\begin{aligned} b_0 &= 1/(T_1T_3T_4) \\ b_1 &= T_2/(T_1T_3T_4) \\ a_0 &= 1/(T_4T_1T_3) \\ a_1 &= (T_4+T_1+T_3)/(T_4T_1T_3) \\ a_2 &= (T_4(T_1+T_3)+(T_1T_3))/(T_4T_1T_3) \end{aligned} \tag{3.30}$$

Ein Pol-Nulstellendiagramm mittels Matlab zeigt, dass sich alle vier Polstellen im negativen Bereich der s-Ebene befinden (vgl. Abbildung 3.23 a)). Ein Pol-Nullstellendiagramm in der z-Ebene wird durch bilineare Transformation der Gleichung 3.29 erreicht. Hier befinden sich die Polstellen innerhalb des Einheitskreises (vgl. Abbildung 3.23 b)).



(a) s-Ebene (b) z-Ebene

Abbildung 3.23: Pol- Nullstellendiagramme der PLL 4. Ordnung

Eine weitere Methode der Stabilitätsprüfung von geschlossenen Regelkreisen liefert das Hurwitz-Kriterium [15]. Dafür wird zunächst die $n \times n$ Hurwitzmatrix aufgestellt, für die im Allgemeinen gilt

$$H = \begin{bmatrix} a_{n-1} & a_{n-3} & a_{n-5} & a_{n-7}\dots \\ a_{n-0} & a_{n-2} & a_{n-4} & a_{n-6}\dots \\ 0 & a_{n-1} & a_{n-3} & a_{n-5}\dots \\ 0 & a_{n-0} & a_{n-2} & a_{n-4}\dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \tag{3.31}$$

Die jeweiligen Koeffizienten a_n stellen dabei die Nennerkoeffizienten des geschlossenen Regelkreises nach Gleichung 3.29 dar. Im Weiteren werden Untermatrizen für das System 4. Ordnung gebildet.

$$H_1 = \begin{bmatrix} a_{n-1} \end{bmatrix} = \begin{bmatrix} a_2 \end{bmatrix} \tag{3.32}$$

$$H_2 = \begin{bmatrix} a_{n-1} & a_{n-3} \\ a_{n-0} & a_{n-2} \end{bmatrix} = \begin{bmatrix} a_2 & a_0 \\ a_3 & a_1 \end{bmatrix} \tag{3.33}$$

$$H_3 = \begin{bmatrix} a_{n-1} & a_{n-3} & a_{n-5} \\ a_{n-0} & a_{n-2} & a_{n-4} \\ 0 & a_{n-1} & a_{n-3} \end{bmatrix} = \begin{bmatrix} a_2 & a_0 & 0 \\ a_3 & a_1 & 0 \\ 0 & a_2 & a_0 \end{bmatrix} \tag{3.34}$$

Sind die Unterdeterminanten größer Null, so ist die Hurwitzmatrix positiv definit und der Regelkreis stabil. Mit den Koeffizienten aus Gleichung 3.30 und den Zeitkonstanten aus Tabelle 3.1 ergibt sich

$$\begin{aligned} det(H_1) &= 7.418 \cdot 10^{13} \\ det(H_2) &= 1.758 \cdot 10^{23} \\ det(H_3) &= 1.294 \cdot 10^{28} \end{aligned} \tag{3.35}$$

und damit ein stabiler Regelkreis. Damit wurde die Stabilität geprüft und die PLL dimensioniert.

3.3 High Density Bipolar Code (HDB-Code)

Der Ausgang des Loopfilters zeigte nach den Trainingssequenzen Unregelmäßigkeiten, welche konstanten Jitter verursachen. Dieses kann jedoch reduziert werden indem beim AMI-Code lange Nullfolgen vermieden werden. Lange Nullfolgen führen dazu, dass das Signal keine Synchronisationsinformation mehr besitzt. Dieses ist bei dem HDB-Code nicht der Fall. Der HDB-Code ist im Bezug auf die Impulsformung identisch mit dem AMI-Code. Der wesentliche Unterschied liegt jedoch in den sogenannten Codeverletzungsregeln. Es werden beim HDB3-Code nur maximal drei Nullen in Folge zugelassen. Sollte eine vierte Null folgen, so werden die vier Nullen durch eine Codeverletzung ersetzt. Dabei wird die Gleichanteilfreiheit des Leistungsdichtespektrums

		Polarität des vorangegangenen Symbols	
		'+'	'-'
Polarität der letzten Codeverletzung	'+'	"-00-"	"000-"
	'-'	"000+"	"+00+"

Tabelle 3.2: HDB3-Code Ersetzungsfolgen nach vier detektierten Nullen

Im Nachfolgenden soll ein HDB3-Encoder sowie -Decoder entwickelt werden. Der Encoder muss zunächst einmal vier Nullen in Folge detektieren können. Hierfür wird ein 4-Bit Schieberegister eingesetzt. Ein darauf folgendes Verknüpfungsschaltnetz ermittelt aus den 4 Bit drei Ausgangssignale (*H,L,E*), mit welchen ein Moore-Automat das HDB3 codierte Ausgangssignal erzeugt.

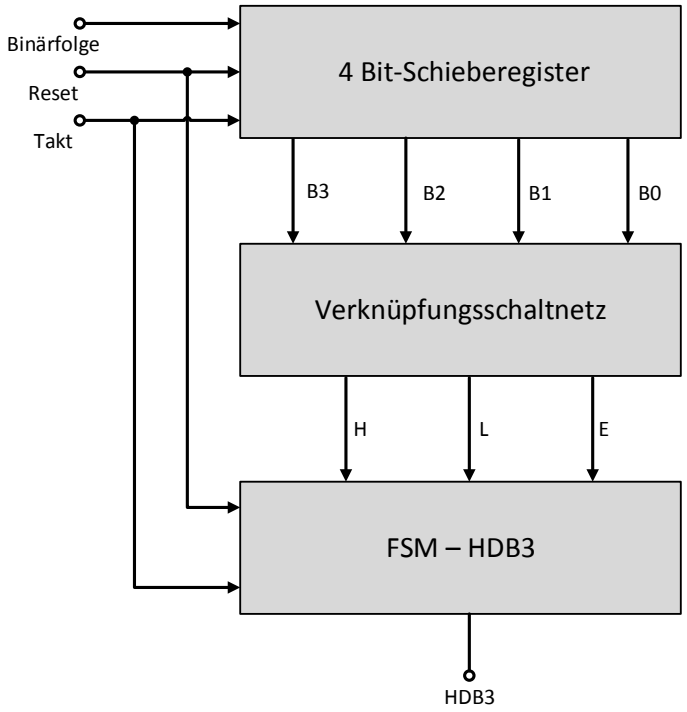


Abbildung 3.24: HDB3-Encoder Blockschaltbild

Sind alle vier Bits des Schieberegisters auf Null, so wird das Signal *E='1'*. Dies kennzeichnet, dass eine Ersetzung folgen muss. Das Signal *H* gibt das letzte Bit des Schieberegisters(*B0*) aus. Ist *H=B0='1'*, so muss der Moore-Automat das Verhalten des AMI-Coders aufweisen und das Ausgangssignal alternieren. Für den Fall, dass *B0≠ '1'*, oder *B0, B1, B2, B3* nicht alle *'0'* sind, muss der Automat eine *'0'* ausgeben. Demnach folgt

$$L = \overline{B0} \cdot (B0 + B1 + B2 + B3) = B1 \cdot \overline{B0} + B2 \cdot \overline{B0} + B3 \cdot \overline{B0}). \tag{3.36}$$

Wie es in der boolschen Algebra üblich ist, bezeichnet '+' eine 'Oder'-Operation und '$\cdot$' eine 'Und'-Operation. Mit diesen Kenntnissen kann das Verknüpfungsschaltnetz aufgebaut werden. Abbildung 3.24 zeigt das Simulink Modell des HDB3-Encoders.

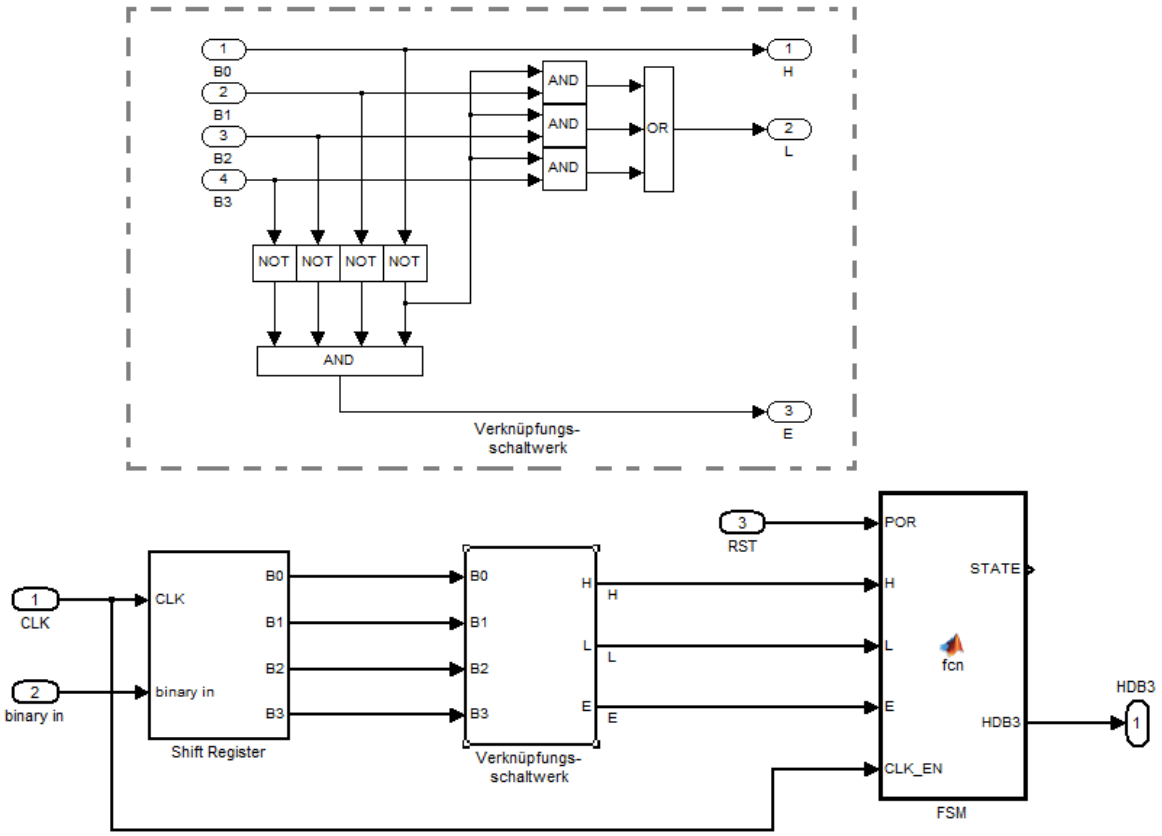


Abbildung 3.25: Simulink Modell des HDB3-Encoders

Mit den Signalen *H,L,E* kann nun der Moore-Automat erstellt werden. Dafür müssen lediglich die Signale des Verknüpfungsschaltwerks und die Ersetzungsfolgen nach Tabelle 3.2 in Einklang gebracht werden. Abbildung 3.26 zeigt den Moore-Automatengraph. Im Anhang A, Listing 7.2, ist der Quellcode des Moore-Automaten zu finden, so wie dieser auch in Hardware implementiert ist.

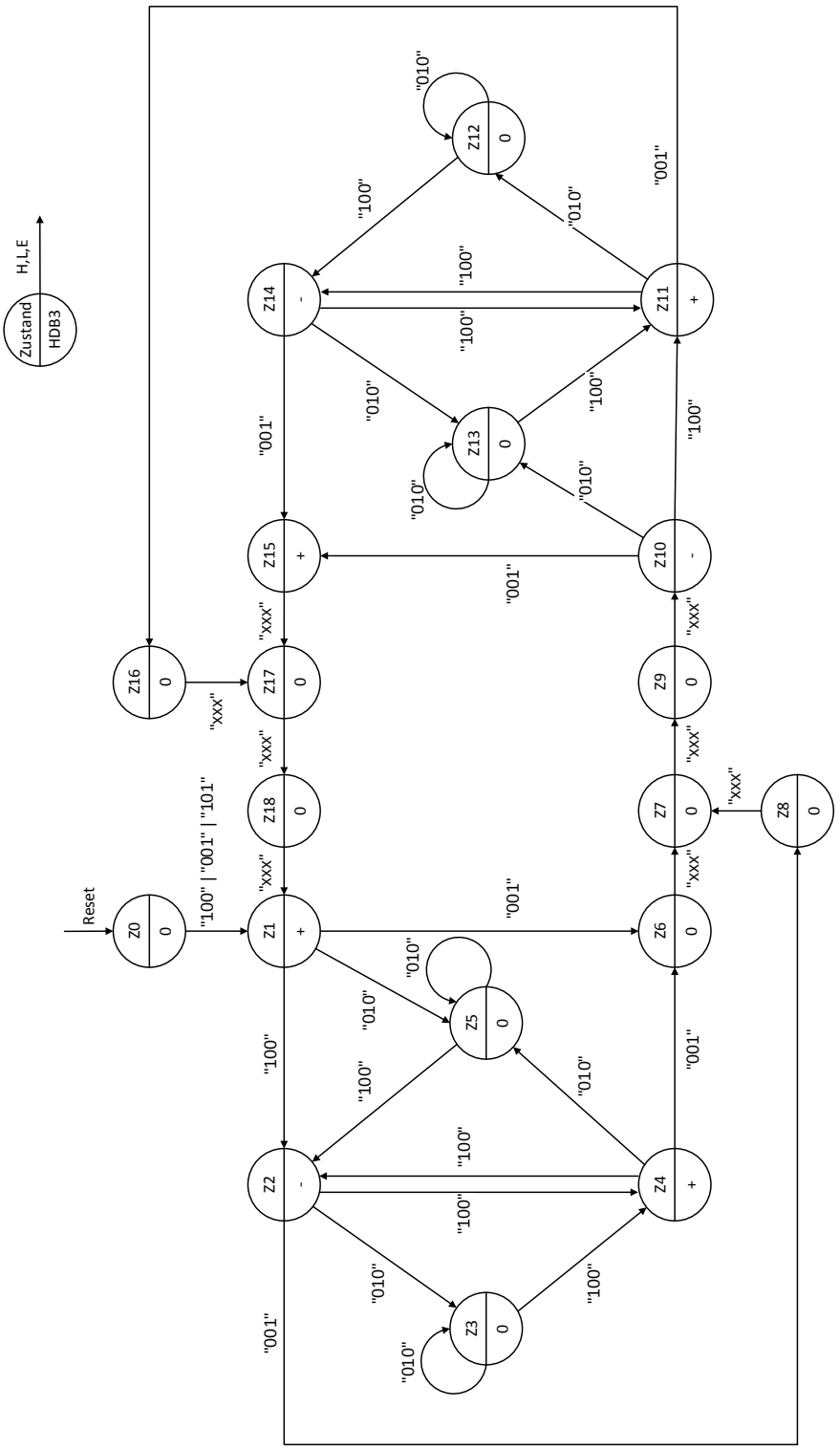


Abbildung 3.26: Finite-state-mashine des HDB3-Encoders

Der HDB3-Decoder muss nun erkennen können, wann eine AMI-Codeverletzung eingefügt wurde, um diese dann entfernen zu können. Ein Blockschaltbild des Decoders ist in Abbildung 3.27 dargestellt.

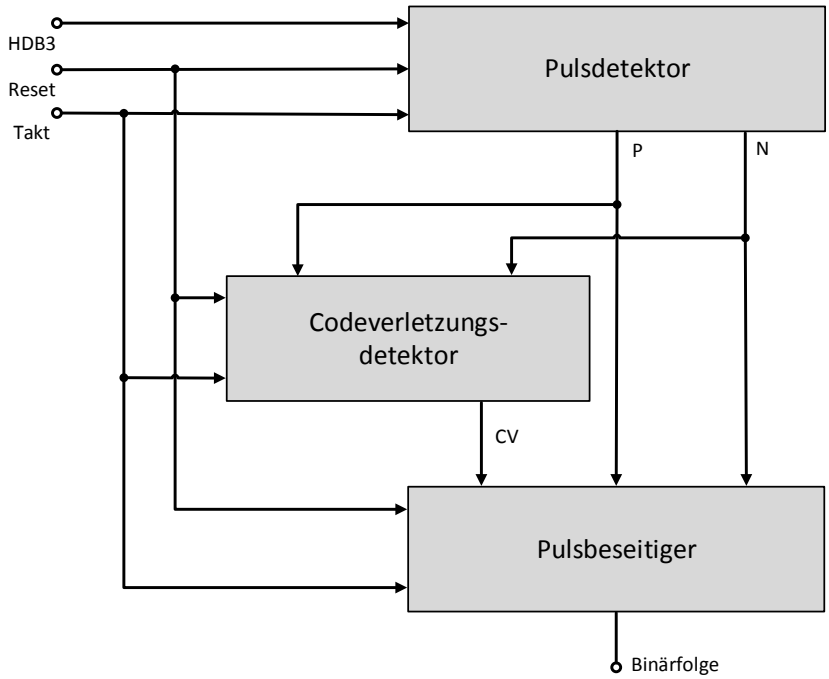


Abbildung 3.27: HDB3-Decoder Blockschaltbild

Zunächst muss detektiert werden, ob ein positiver oder negativer Impuls empfangen wurde. Dies geschieht mit einer einfachen '>' oder '<' Operation. Der Codeverletzungsdetektor speichert die zuletzt empfangene Pulspolarität LP. Damit kann die Codeverletzungsregel mittels boolscher Algebra ermittelt werden zu

$$CV = (P \cdot LP) + (N \cdot \overline{LP}). \tag{3.37}$$

Im Pulsbeseitiger wird die empfangene Symbolfolge AMI decodiert. Dies geschieht einfach über die oder-Bildung des Pulsdetektors. Anschließend müssen nur noch die fehlerhaften Bits entfernt werden. Tritt eine Codeverletzung auf (*CV='1'*), so wird dieses Signal invertiert genutzt, um das jeweilige Bit mittels und-Bildung zu entfernen. Abbildung 3.28 zeigt das Simulink Modell des HDB3-Decoders.

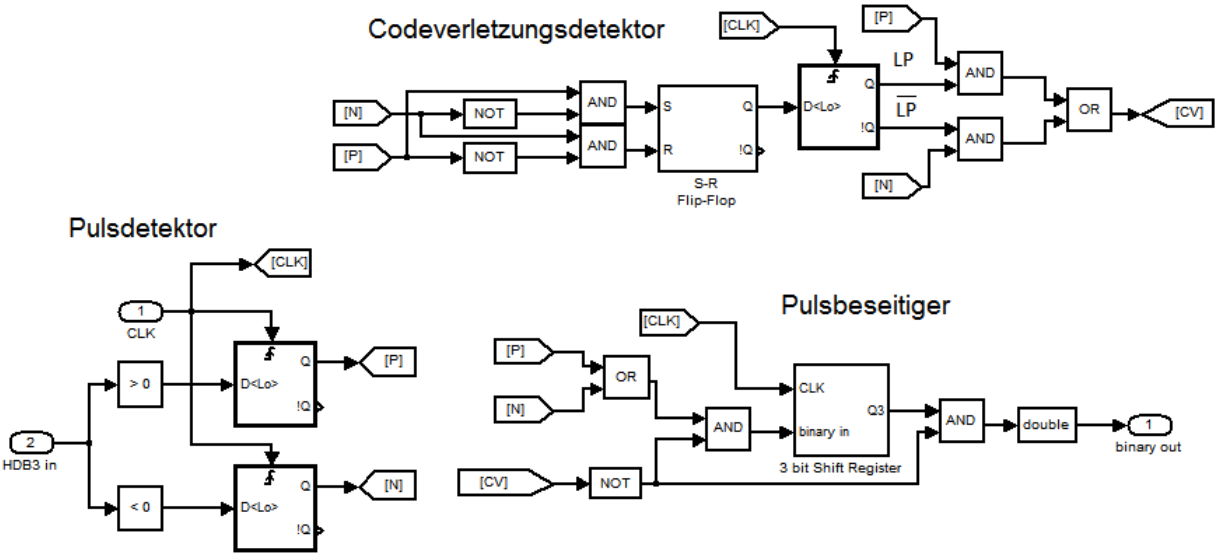


Abbildung 3.28: Simulink Modell des HDB3-Decoders

Abbildung 3.29 veranschaulicht die Codierung. Hier ist zu erkennen, dass bei vier gesendeten Nullen in Folge dem AMI-Code diese auch zu entnehmen sind. Beim HDB3-Signal hingegen ist eine Codeverletzung eingefügt.

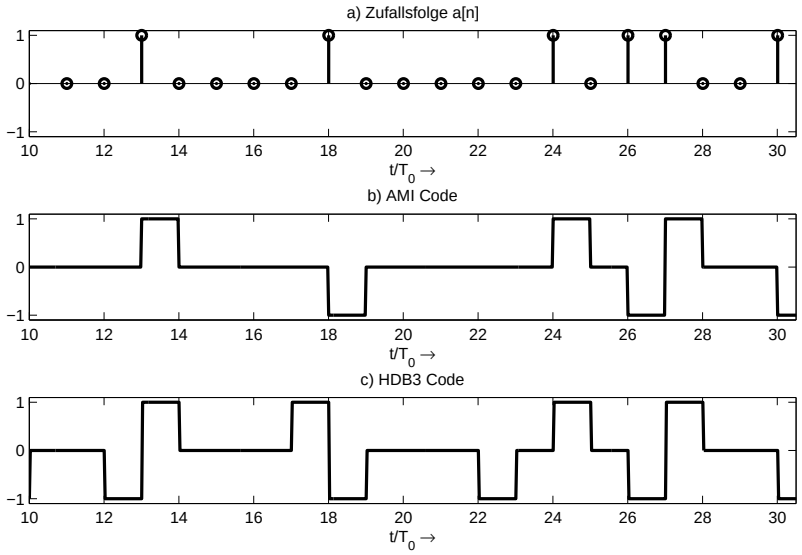


Abbildung 3.29: Veranschaulichung der HDB3-Codierung

Um eine Aussage über den Leistungsanstieg an der Taktfrequenz f_0 im Vergleich zum AMI-Code machen zu können, wird eine Messung des Leistungsdichtespektrums durchgeführt. Die Auftrittswahrscheinlichkeit von *'0','1'* der zu codierenden Eingangsfolge wird hierbei als gleichverteilt eingestellt. Eine Messung erfolgt über 2^{18} Bits beim 5RC-Kanalmodell am Ausgang der nichlinearen Operation zur Taktrückgewinnung. Abbildung 3.30 zeigt das Messergebnis.

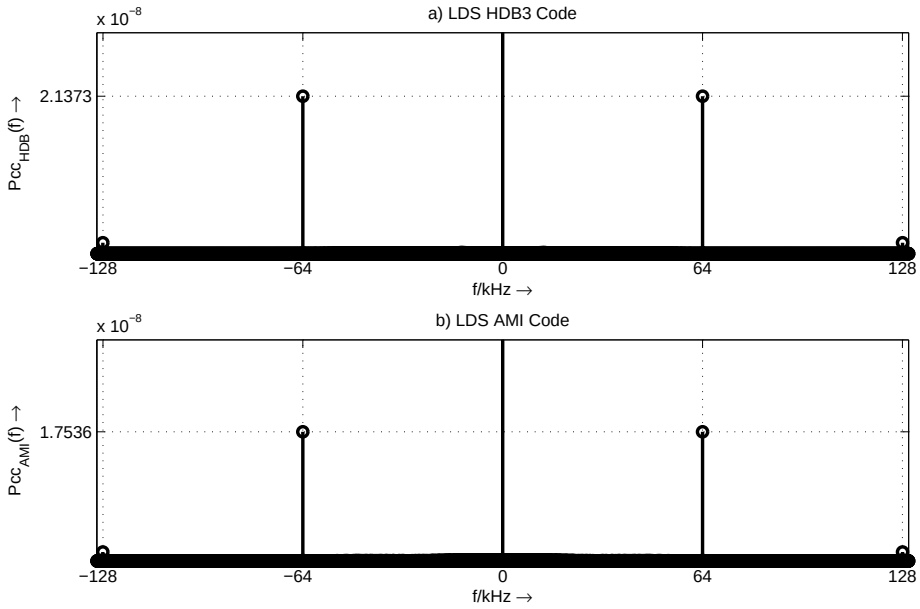


Abbildung 3.30: Leistungsdichtespektrum des AMI- und HDB3-Codes

Beide Messwerte wurden mit einem Hamming-Fenster gewichtet. Aus dem Ergebnis kann dem Leistungsdichtespektrum des HDB3-codierten Signals eine um $(2,1373/1,7536-1)\cdot 100 = 21.88\%$ größere Leistung an der Taktfrequenz entnommen werden.

An dieser Stelle soll noch eine Betrachtung des Loopfilterausgangs nach Beenden der PLL-Trainingssequenz erfolgen. Dabei wird zwischen bipolarem NRZ- und HDB3-Code unterschieden. Die Simulation erfolgt an einem 6RC-Kanalmodell. Nach Beenden der Trainingssequenz ($t/T_0 > 64$) erfolgt durch gesendete Zufallsfolgen ein hin und her zitterndes Loopfilterausgangssignal, vgl. Abbildung 3.31.

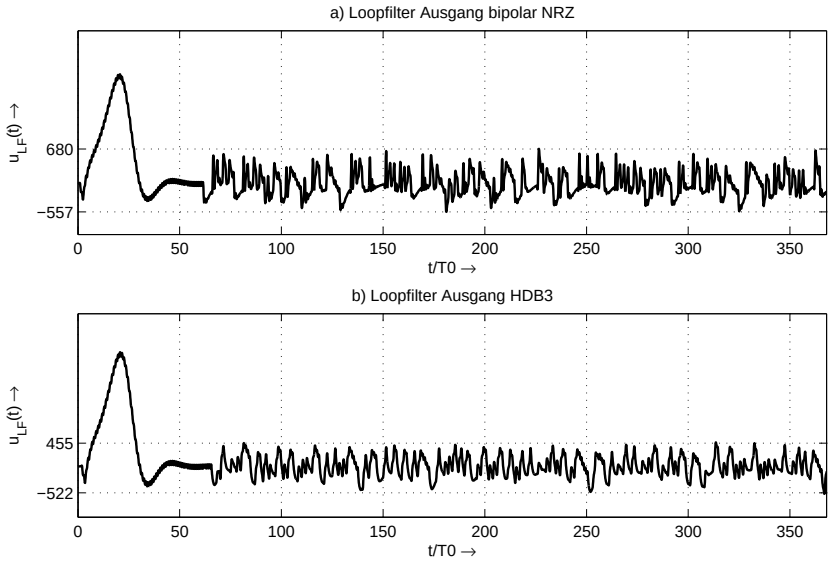


Abbildung 3.31: Loopfilterausgang im Vergleich mit bipolar NRZ- und HDB3-Code

Dies entsteht dadurch, dass das Signal $u_1(t)$ nun keine periodischen Nullstellen im Taktraster T_0 mehr besitzt. Abbildung 3.32 veranschaulicht dieses. Die Multiplikaton mit dem VCO-Ausgangssignal zeigt den zu filternden Phasendetektorausgang. Anhand der Sitze-Spitze Werte in Abbildung 3.31 kann die Schlussfolgerung getroffen werden, dass der in Kauf zu nehmende Jitter bei der HDB3-codierten Sendefolge geringer ist als bei der bipolaren NRZ-codierten Folge.

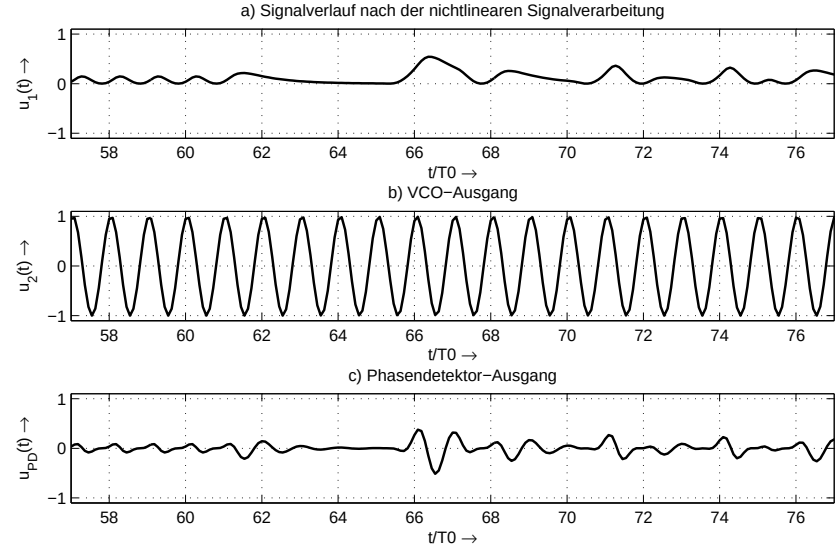


Abbildung 3.32: Veranschaulichung des Phasendetektorausgang

4 Audio-Interface

Ein Audio-Interface besitzt mehrere Aufgaben. Dabei beziehen sich diese auf den analogen sowie digitalen Signalbereich. Die Hauptkomponenten bestehen aus einem ADC und DAC, welche alleine jedoch nicht ausreichen.

4.1 Anti-Aliasing- und Interpolations-Filter

In Kapitel 2.2.1 wurden bereits die Digitalisierung sowie Aliasing bzw. das Anti-Aliasing-Filter beschrieben. Wird das Audiosignal im Empfänger durch den DAC wieder rekonstruiert, so weist dieses Stufen im Abstand T_A auf. Dies soll Abbildung 4.1 b), mit einer Abtastfrequenz von $f_A = 8kHz$, veranschaulichen. Durch ein einfaches Tiefpassfilter kann das Ausgangssignal des DAC interpoliert werden, sodass dieses wieder mit dem gesendeten Signal übereinstimmt, vgl. Abbildung 4.1 a) und c). Aus diesem Grund nennt man dieses Filter auch Interpolations- oder Rekonstruktionsfilter.

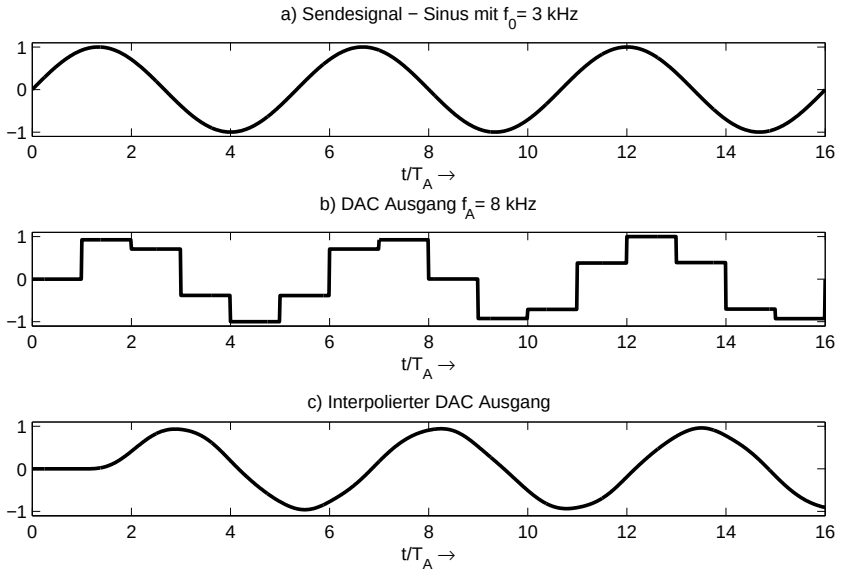


Abbildung 4.1: Veranschaulichung der Interpolation mit einem Butterworth-Tiefpassfilter 4. Ordnung $f_g = 3,4\ kHz$

An dieser Stelle soll die Dimensionierung des AAF und des Interpolationsfilters erfolgen. Durch die Gegebenheiten des Fernspechnetzes kann aus Gleichung 2.25 eine obere Grenzfrequenz von $f_g = 3,4\ kHz$ entnommen werden. Im Folgenden wird von einem 12-Bit ADC ausgegangen. Die zu digitalisierende Eingangsspannung hat einen Spitze-Spitze Wert von $U_{pp} = 2 \cdot 1,024\ V$. Die kleinste Quantisierungsstufe stellt das LSB (engl. für least significant bit) der 12 Bit dar. Die kleinste aufzulösende Eingangsspannung des ADC lässt sich damit errechnen zu

$$U_{LSB} = \frac{2,048\ V}{2^{12} - 1} = 500\ \mu V. \tag{4.1}$$

Nach dem Nyquist-Shannon Abtasttheorem ergibt sich eine erforderliche Mindest-Dämpfung des Anti-Aliasing-Filters bei der halben Abtastfrequenz zu

$$A_{AAF}\left(\frac{f_A}{2}\right) \geq 20 \cdot log_{10}\left(\frac{U_{pp}}{U_{LSB}}\right) = 72,2\ dB. \tag{4.2}$$

Eine Umsetzung des AAF erfolgt zwangsläufig immer im analogen Signalbereich. Die erforderliche Ordnung des Filters kann durch die Steilheit abgeschätzt werden [16].

$$\delta_{Dek} = \frac{-A_{AAF}\left(\frac{f_A}{2}\right)}{log_{10}\left(\frac{f_A}{2}\right) - log_{10}\left(f_g\right)} = -1023\ dB/Dek \tag{4.3}$$

Damit ergibt sich ein Filter der Ordnung 52. Solch ein Filter zu realisieren bedeutet nicht nur großen Aufwand, hinzu kommen noch die Toleranzen der Bauelemente. Diese führen dazu, dass die Filtereigenschaften stark schwanken. Die Filterordnung kann jedoch einfach verringert werden, indem die Steilheit geringer wird. Dies erfordert eine Erhöhung der Abtastfrequenz (auch Oversampling genannt). Wird für diese eine um den Faktor 100 größere Abtastfrequenz $f_A = 800\ kHz$ gewählt, so ergibt sich nach Gleichung 4.3 eine Steilheit von $\delta_{Dek} = -35\ dB/Dek$. Demnach reicht ein Tiefpassfilter 2. Ordnung. Im Bezug auf die Implementierung ist zu erwähnen, dass die Abtastfrequenz f_A sychron mit der Bittaktfrequenz $f_0 = 64\ kHz$ sein muss. Diese werden durch den Systemtakt abgeleitet. Mehr dazu im späteren Kapitel 5.3. Bedingt durch den Systemtakt und der Synchronität mit dem Bittakt, wird die nächst mögliche Abtastfrequnez von $f_A = 704\ kHz$ gewählt. Ein Tiefpass 2. Ordnung ist dabei immer noch ausreichend. Da zu hohen Frequenzbereichen hin keine großen Störeinflüsse zu erwarten sind, wird ein einfaches RC-Tiefpassfilter 1. Ordnung mit den Parametern $C = 10\ nF$, $R = 4,7\ k\Omega$ gewählt. Der Amplitudengang dieses Filters ist in Abbildung 4.2 dargestellt.

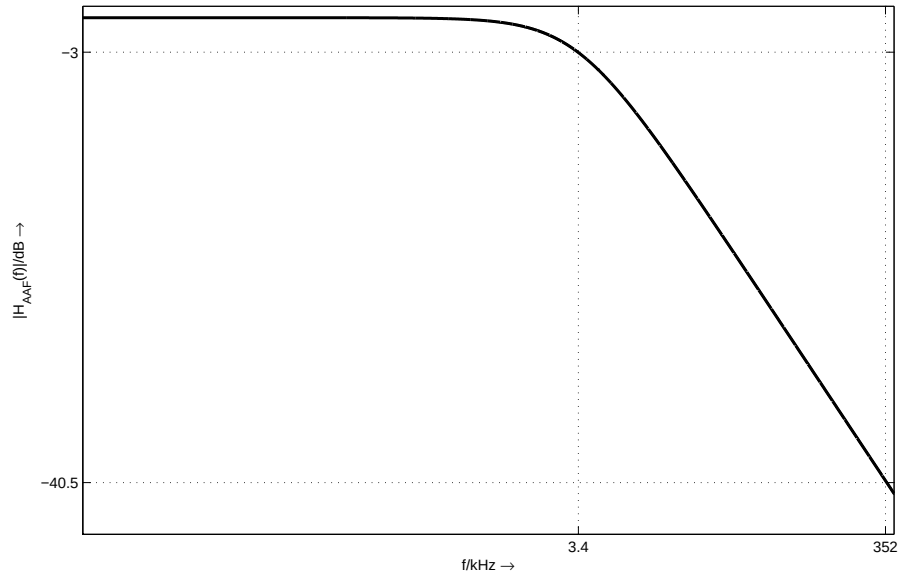


Abbildung 4.2: Amplitudengang des analogen Anti-Aliasing-Filters 1. Ordnung

Mit der hohen Abtastrate wird aber noch kein Aliasing vermieden. Alias-Effekte werden lediglich mit digitalisiert. Demnach muss ein zusätzliches digitales Anti-Aliasing-Filter hinzukommen. Das von Mathworks entwickelte Programm Filter Design and Analysis Tool (FDA-Tool) in Matlab bietet eine genaue Dimensionierung eines Filters [17]. Das FDA-Tool wird für die Auslegung des Anti-Aliasing-Filters und des Interpolationsfilters verwendet. Dies kann ein IIR-Filter (engl. infinite impulse response filter) oder ein FIR-Filter sein. Die Ordnung eines FIR-Filters mit Rechteckfensterung kann nach [18] abgeschätzt werden zu

$$M = \frac{0,9}{\Delta f'}. \tag{4.4}$$

Wobei $\Delta f'$ die auf die Abtastfrequenz normierte Breite des Überganges von Durchlass- zu Sperrbereich ist. Mit einer Übergangsbreite von $\frac{4\ kHz - 3{,}4\ kHz}{704\ kHz}$, ergibt sich eine Ordnung von 1056.

Ein IIR-Filter hingegen bietet die Möglichkeit ein analoges Filter nachzubilden. Ein Cauer-Filter oder auch elliptisches Filter genannt, bietet einen sehr steilen Übergang des Frequenzgangs vom Durchlass- in den Sperrbereich. Dieses weist dabei minimale Ordnungen auf [19]. Mit Angabe von Abtastfrequenz f_A, Grenzfrequenz f_g, Sperrfrequenz f_c, Sperrdämpfung a_c und Welligkeit im Durchlassbereich a_p, wird das Filter mittels FDA-Tool in Matlab dimensioniert. Tabelle 4.1 listet die eingestellten, sowie vom FDA-Tool erzeugten Filterparameter auf.

$f_A = 704\ kHz$
$f_g = 3,4\ kHz$
$f_c = 4,0\ kHz$
$a_p = 1\ dB$
$a_c = 80\ dB$
Ordnung = 10
Stabilität gewährleistet

Tabelle 4.1: Filterparameter des FDA-Tool

Mit einer Ordnung von 10 lässt sich sogar noch eine theoretische Sperrdämpfung von 80 dB erzielen. Die wahre Sperrdämpfung bestimmt jedoch der Dynamikbereich des ADC, vgl. Gleichung 4.2. Das FDA-Tool erzeugt sogenannte SOS-Kaskaden (section-order section-cascade). Dies sind hintereinander geschaltete IIR-Filter 2. Ordnung (sogenannte Biquads [18]). Diese bieten den Vorteil, dass durch Pol-/Nullstellenpaarung der SOS-Kaskaden, die Koeffizienten mit höheren Indizes nicht so klein werden, dass diese ungenau quantisiert werden [20]. Des Weiteren lässt sich mit dem FDA-Tool bestimmen, welche Realisierungsform die einzelnen Biquad-Sektionen haben sollen. Die Standardeinstellung erzeugt die Direktform II. Abbildung 4.3 zeigt das erzeugte Filter in Simulink. Darstellungshalber wurden zwei Biquad-Sektionen weggelassen.

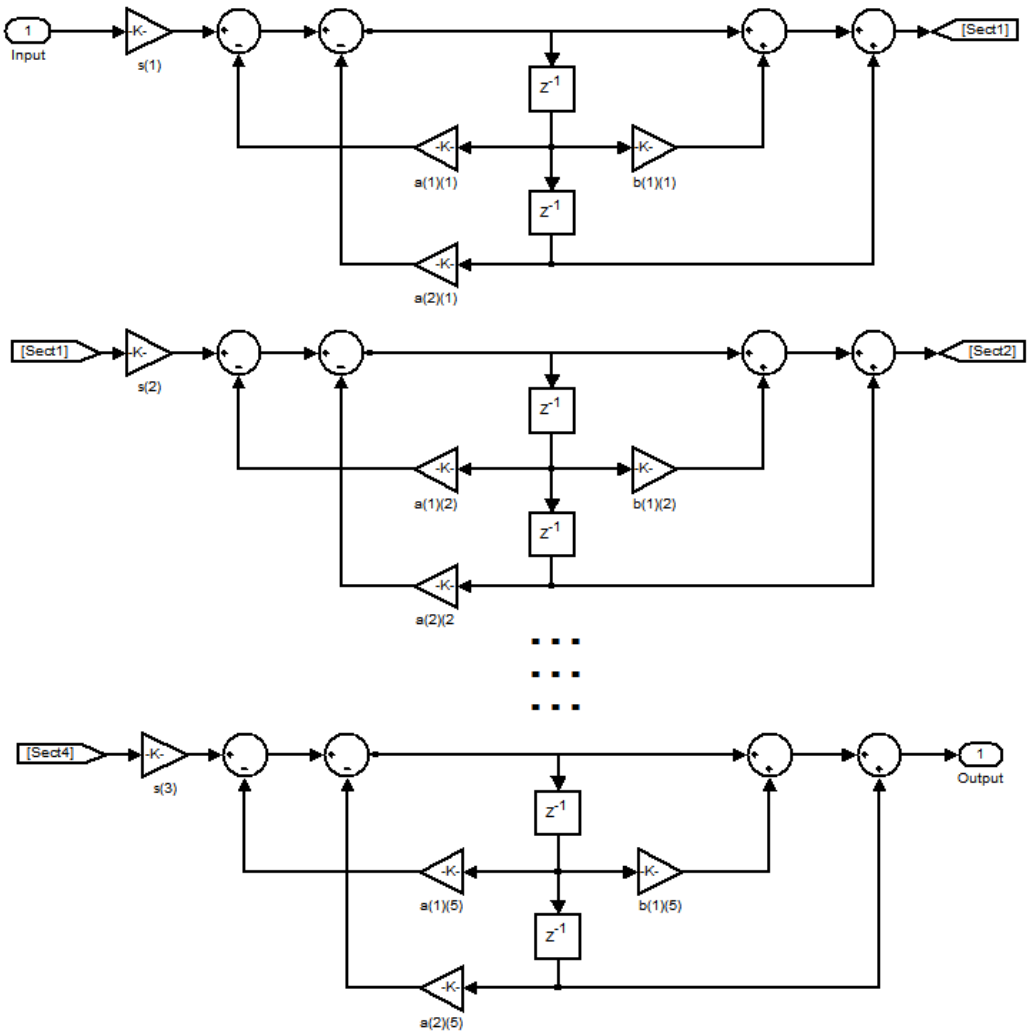


Abbildung 4.3: Elliptisches IIR-Tiefpassfilter 8. Ordnung in Biquad-Kaskadierung mit der Direktform II

Im Nachhinein wurde jedoch festgestellt, dass die transponierte Direktform II eine bessere Wahl gewesen wäre. Hier können zusätzlich zwei Multiplizierer eingespart werden. Der längste Signalpfad weist zudem einen Addierer weniger auf [20]. Zur Interpolation wird ein identisches Filter verwendet. Der Amplitudengang ist in Abbildung 4.4 dargestellt. Diesem sind die gewünschten Parameter aus Tabelle 4.1 zu entnehmen.

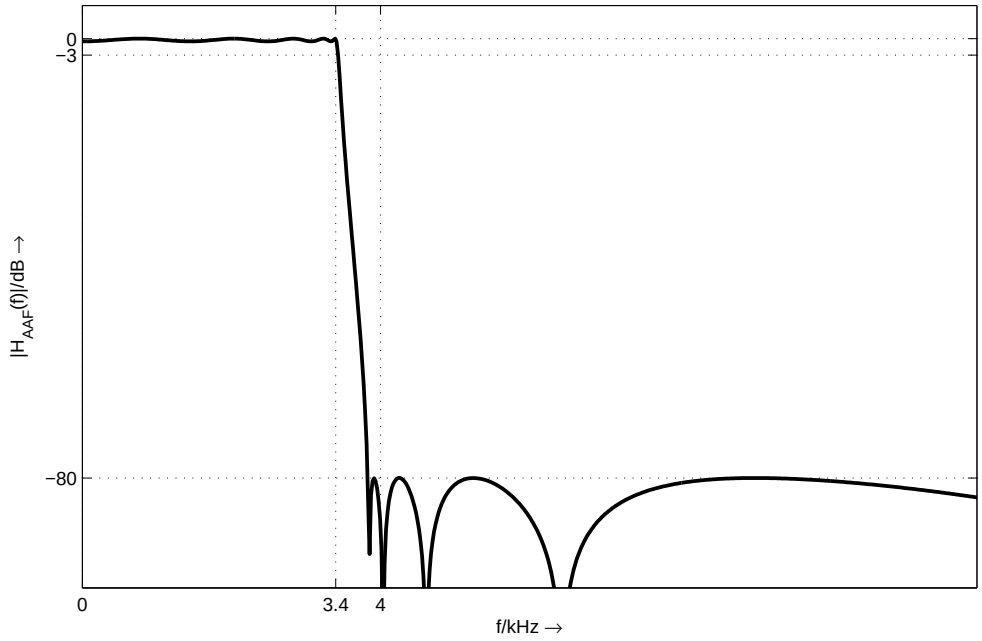


Abbildung 4.4: Amplitudengang des elliptischen IIR-Tiefpassfilters

4.2 P/S- und S/P-Umsetzung

Weitere Komponenten eines Audio-Interfaces gehen aus der Datenübertragung der PCM-Technik hervor. Hier werden die Daten seriell über den Kanal übertragen. Ein ADC liefert jedoch zu jedem Abtastzeitpunkt ein paralleles Wort mit einer definierten Bitbreite. Dies führt zwangsweise dazu, dass die parallelen Datenbits seriell umgesetzt werden müssen. Dies kann mit einem Parallel-Seriell-Wandler (P/S) realisiert werden. Zunächst wird das Wort über einen Demultiplexer in einzelne parallele Bits überführt. Dem P/S-Umsetzer werden zwei unterschiedliche Taktfrequenzen zugeführt. Der Takt für die serielle Ausgabe ist dem der parallelen Eingabe um den Faktor der Bitbreite des parallelen Wortes größer. Bei der ISDN-Technik wird ein analoges Signal mit der Abtastfrequenz von $f_A = 8\ kHz$ abgetastet und auf dem Kanal mit einer Bitrate von $v = 64\ kbps$ übertragen [6]. Daraus folgt eine Wortbreite von 8 Bit. Abbildung 4.5 zeigt den 8 Bit Parallel-Seriell-Wandler in Simulink.

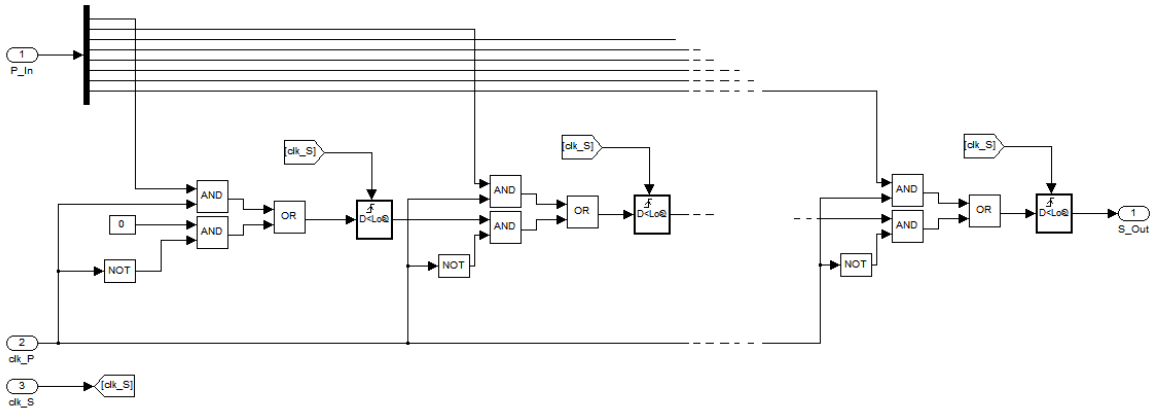


Abbildung 4.5: Simulink Modell des 8 Bit Parallel-Seriell-Wandlers

Analog dazu muss im Empfänger das ursprüngliche Wort zurückgewonnen werden. Hierfür wird ein Seriell-Parallel-Wandler eingesetzt. Dieser speichert die seriellen Bits in Flip Flops. Dessen Ausgänge werden dann mit dem um Faktor 8 heruntergeteilten Takt auf einen Multiplexer gegeben, vgl. Abbildung 4.6.

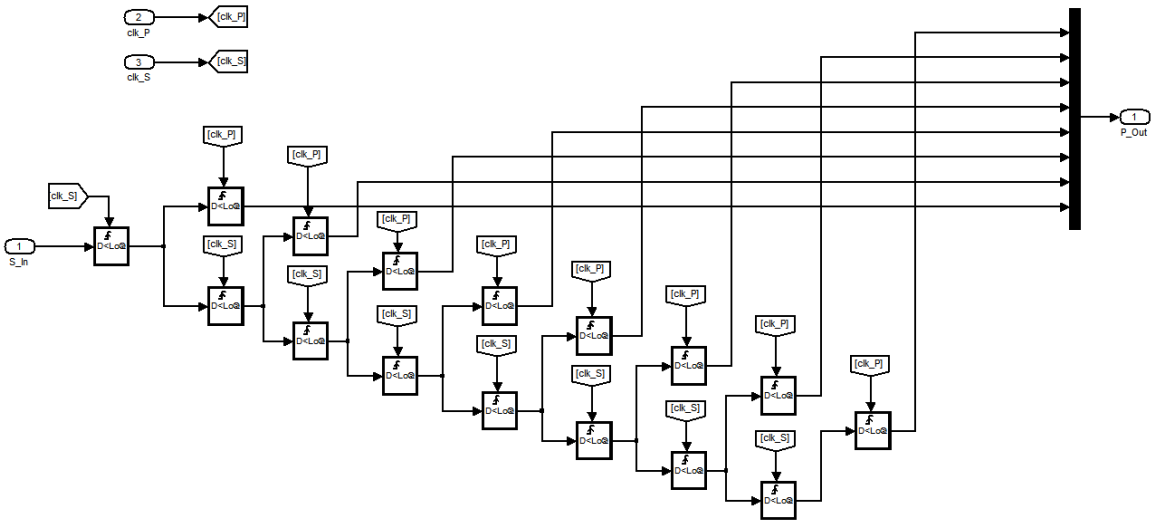


Abbildung 4.6: Simulink Modell des 8-Bit Seriell-Parallel-Wandlers

4.3 Kompandierung

Wie bereits erwähnt liefert der ADC eine Auflösung von 12 Bit. Mit der Kompandertechnik, welche bei der ISDN-Technik üblich ist, findet eine Komprimierung auf 8 Bit statt. Das Blockschaltbild in Abbildung 4.7 zeigt das Prinzip.

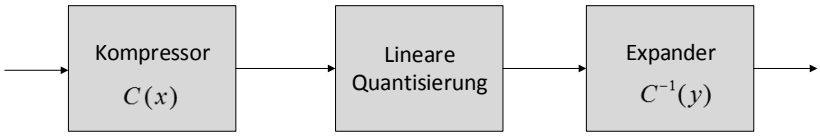


Abbildung 4.7: Blockschaltbild der nichtlinearen Quantisierung mittels Kompandertechnik

In Abschnitt 2.1 wurde auf den geforderten Mindest-Störabstand von 30 dB hingewiesen. Dieser wird mit linearer Quantisierung von 8 Bit für kleine Eingangspegel nicht erreicht. Gefordert wird ein pegelunabhängiger SNR. Dies wird teilweise durch eine logarithmische Kennlinie erreicht. Dabei wird zwischen zwei unterschiedlichen Kennlinien unterschieden. Der μ-LAW Kennlinie, welche Anwendung in Nordamerika und Japan findet, und der A-LAW Kennlinie, welche in Europa angewandt wird. In dieser Thesis wird die europäische Variante genutzt. Die A-LAW Kennlinie ist als stetige Funktion definiert.

$$C(x) = \begin{cases} x \cdot \dfrac{A}{1 + ln(A)}, & 0 \leq x \leq 1/A \\ \dfrac{ln(A) \cdot x}{1 + ln(A)} + \dfrac{1}{1 + ln(A)}, & 1/A < x \leq 1 \end{cases} \tag{4.5}$$

Bei ISDN wird eine Konstante von $A = 87,6$ verwendet [21]. Der Gewinn des Kompanders beträgt demnach

$$G_K = 20 \cdot log10 \left(\frac{A}{1 + ln(A)} \right) = 24\ dB, \tag{4.6}$$

welcher sich bei betrachten des SNR bemerkbar macht, vgl. Abbildung 4.8, -14 dB bis -38 dB. Für größere Eingangspegel wird hingegen ein verminderter SNR in Kauf genommen.

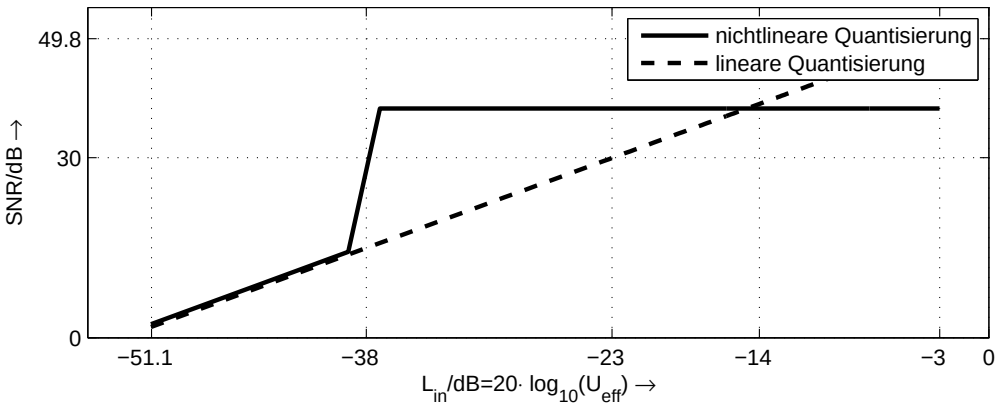


Abbildung 4.8: SNR als Funktion des Eingangspegels - Vergleich lineare- und nichlineare Quantisierung

Für die Implementierung der Kennlinie nach Gleichung 4.5 erfährt diese eine stückweise, durch 13 Segmente lineare Approximation. Diese ist in Form einer Lookup-Tabelle[1] in der ITU-T zu finden [22]. Die Lookup-Tabellen des Kompressors und des Expanders sind in Matlab Skript Files implementiert, welche dem Anhang B und C zu entnehmen sind. Die Kennlinie des Kompressors für 12- auf 8-Bit, zeigt Abbildung 4.9. Deutlich zu erkennen ist die Verstärkung der kleiner quantisierten Eingangswerte mit logarithmischen Verlauf.

[1] In Lookup-Tabellen (LUT) werden Werte zur Vermeidung aufwändiger Berechnungen statisch definiert

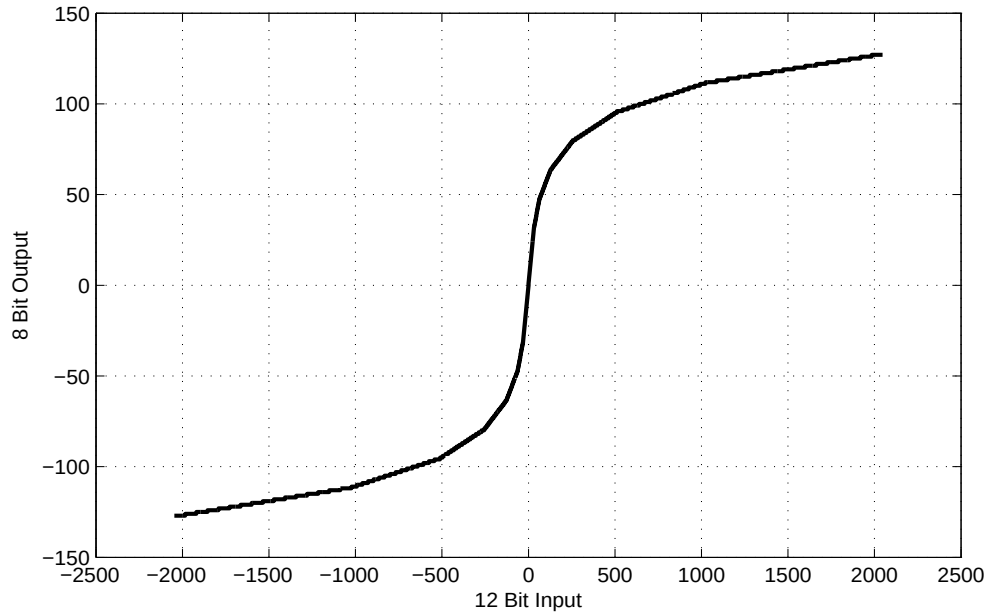


Abbildung 4.9: Kompressor A-LAW Kennlinie

Empfängerseitig muss schließlich mit einer inversen Kennlinie wieder zurück auf 12 Bit expandiert werden.

$$C^{-1}(y) = \begin{cases} sgn(y) \cdot \dfrac{|y|(1+ln(A)}{A}, & |y| < \dfrac{1}{1+ln(A)} \\ \dfrac{exp(|y|(1+ln(A))-1)}{A}, & \dfrac{1}{1+ln(A)} \leq |y| < 1 \end{cases} \tag{4.7}$$

Abbildung 4.10 zeigt die Expander Kennlinie nach Umsetzung der Lookup-Tabelle aus der ITU-T [22]. Hier werden die kleiner quantisierten Signale wieder herabgesenkt. Zu erkennen ist eine durch den Ursprung verschobene Exponentialfunktion.

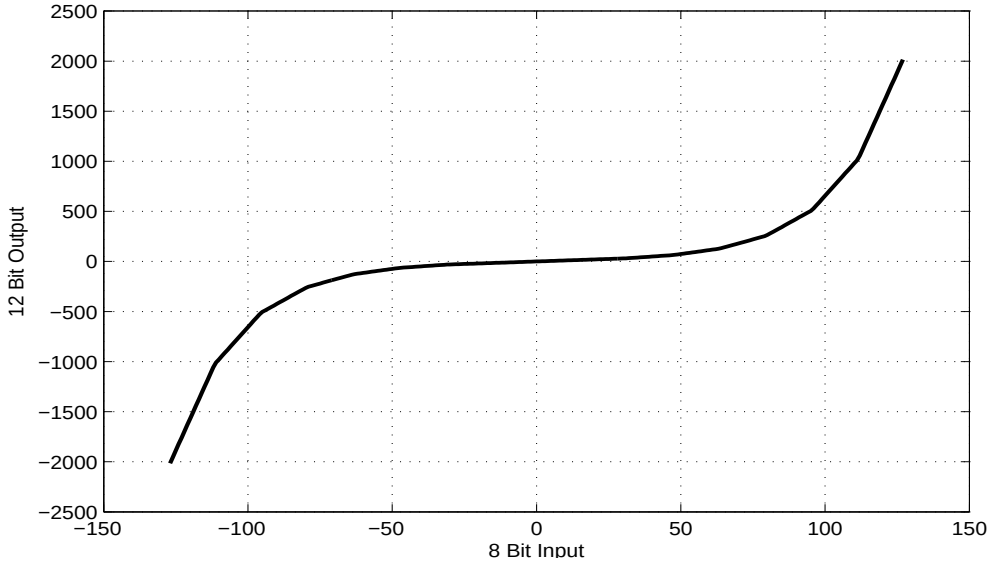


Abbildung 4.10: Expander A-LAW Kennlinie

Abbildung 4.11 zeigt die Einflüsse der Kompandierung über die Zeit. Die vertikale Richtung zeigt die Quantisierungsstufen. Die Kennlinien wurden mit einem Sinus voll ausgesteuert. Die Differenz des Kompandereingangs zum Expanderausgang, zeigt den Verlust, den die Kompandierung verursacht. Dieser ist jedoch im akustischen Bereich nicht wahrnehmbar.

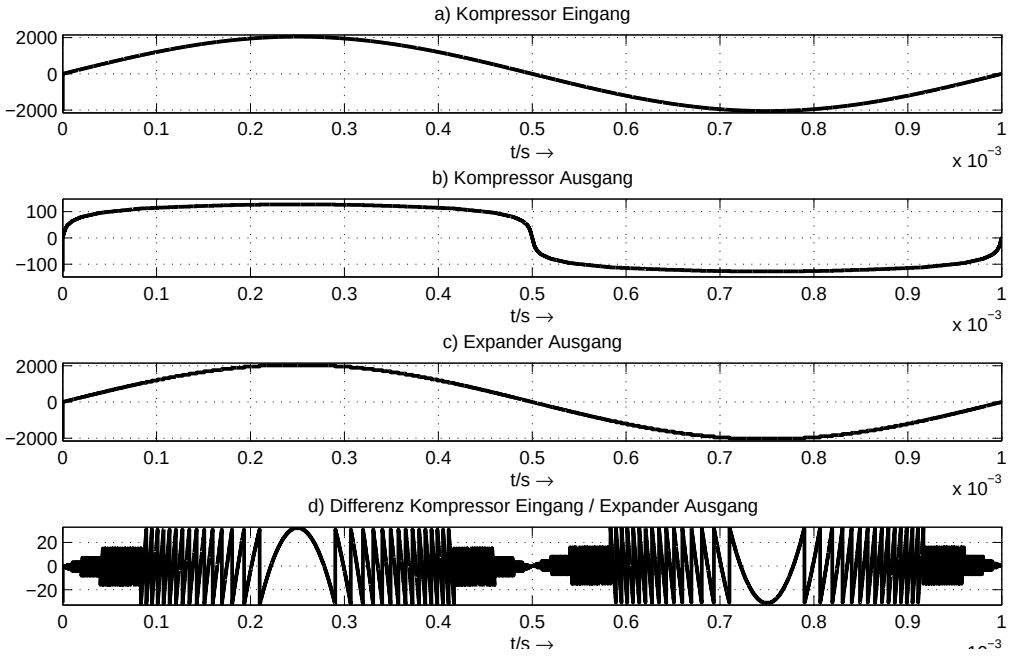


Abbildung 4.11: Veranschaulichung der Kompandierung im Zeitbereich

5 Konzept und Realisierung

Mit den Erkenntnissen der bisherigen Kapitel kann nun ein Gesamtsystem zusammgefasst, und auf dessen Hardware-Realisierung eingegangen werden. Zuvor werden die verwendete Hardware und die bereits bestehenden Komponenten erläutert. Die Implementierung erfolgt auf einem FPGA (Field Programmable Gate Array), dessen Aufbau und Funktion kurz erläutert wird. Der Virtex 5 ist ein FPGA der Firma Xilinx, welcher aus einer Vielzahl an konfigurierbaren Logikblöcken (CLB) besteht (vgl. Abbildung 5.1). Diese CLBs sind in sogenannte Slices unterteilt, wovon jeder vier Flip-Flops, Multiplexer und Speicherblöcke besitzt. Die Speicherblöcke können dabei als Lookup-Tabellen oder RAM-Bausteine dienen.

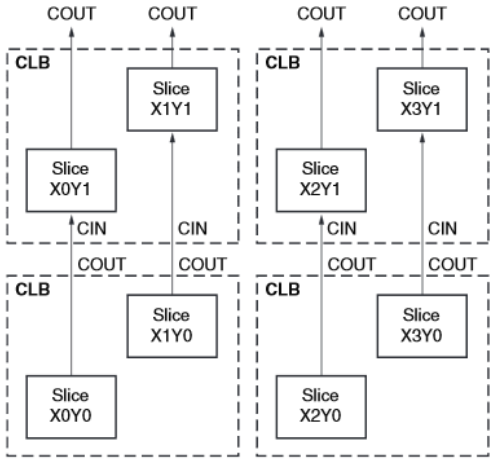


Abbildung 5.1: Konfigurierbare Logikblöcke (CLBs) Quelle [23]

Die konfigurierbaren Logikblöcke können über Verbindungsmatrizen flexibel verschaltet werden oder direkt als Ein- oder Ausgänge genutzt werden. Weitere Komponenten sind Hardware-Multipliziereinheiten, Taktgeneratoren und Block RAM. Ein solches FPGA ist auf dem ML507-Board vorhanden, auf welchem die Implementierung erfolgt.

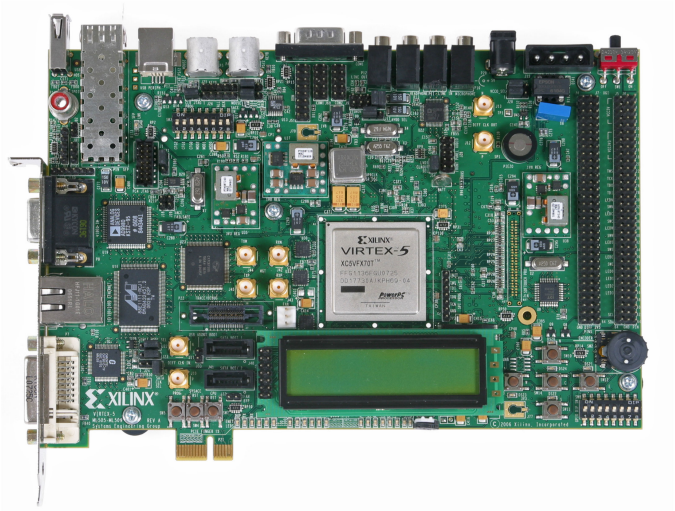
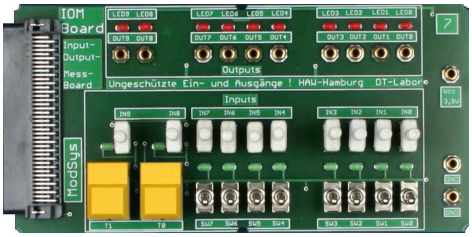
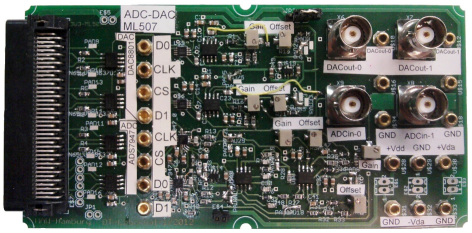


Abbildung 5.2: Xilinx Entwicklungsboard ML507 Quelle [24]

Des Weiteren befindet sich auf dem Board vielseitig einsetzbare Peripherie. Ein ADC bzw. DAC ist jedoch nicht vorhanden. Für diesen Fall sind bestimmte Connectoren vorhanden, über die spezielle Peripherie nachgerüstet werden kann. Das Labor für Digitale Systeme an der HAW Hamburg besitzt für Lehrzwecke ein ADC/DAC-Board, welches für diese Thesis verwendet wird. Neben diesem wird noch ein Input-Output-Messboard (IOM-Board) verwendet.



(a) IOM-Board



(b) ADC/DAC Board

Abbildung 5.3: ModSys-Peripherieboards Quelle [24]

Für komplexere Aufgaben der Signalverarbeitung bietet Xilinx sogenannte IP-Core[1]. Diese werden durch einen 32-Bit-RISC[2]-Mikrocontoller auf dem ML507-Board realisiert. Auf diesem Prozessor lassen sich mehrere Soft-Cores (MicroBlaze) instanziieren. Ein solcher MicroBlaze-Core stellt dabei die zentrale Einheit eines Embedded Systems dar.

[1] Als IP-Core (engl. Intellectual Property Core) wird eine mehrfach verwendbare Funktionseinheit eines Chipdesigns bezeichnet

[2] RISC (Reduced Instruction Set Computer) kennzeichnet Rechner mit reduziertem Befehlssatz

5.1 Embedded System

Systeme, die für spezielle Anwendungen entwickelt wurden, werden als Embedded Systems (eingebettete Systeme) bezeichnet. Dabei wird üblicherweise eine Kombination aus Hard- und Software eingesetzt. Ein solches System ist in dieser Thesis entwickelt worden und läuft auf Basis des MicroBlaze-Core. Für die Entwicklung stellt Xilinix ein Embedded Development Kit (EDK) zur Verfügung. Dieses bietet eine vollständige Entwicklungsumgebung für das Design eines Embedded Systems. Mit dem Xilinx Platform Studio (XPS) lässt sich unterschiedliche Peripherie bzw. IP-Cores, über den Processor Local Bus (PLB) mit dem MicroBlaze verbinden. Abbildung 5.4 zeigt die Hardwarearchitektur des Embedded Systems.

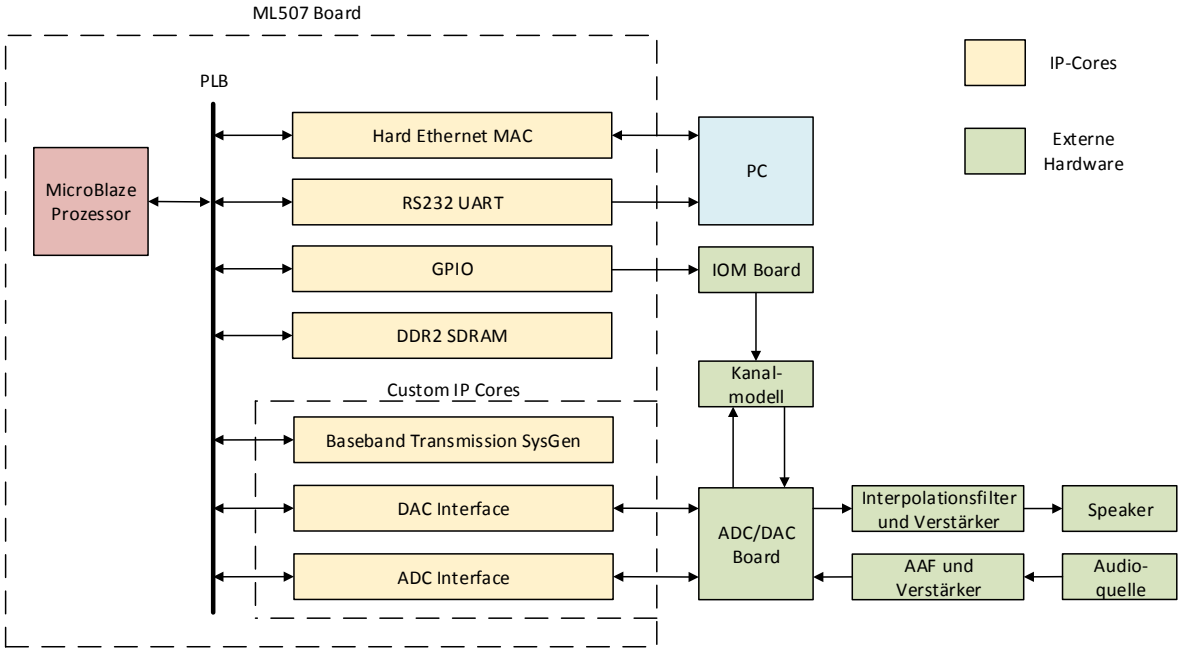


Abbildung 5.4: Hardwarearchitektur des Embedded Systems mit externer Hardware

- **Hard Ethernet MAC**
  Eine Ethernet-Verbindung zwischen PC und ML507-Board dient einer Steuerung sowie grafischer Signalauswertung über Matlab
- **RS232 UART**
  Um Statusflags und gesetzte Registerparameter des Systems über ein Terminalpro-

gramm auslesen zu können, wird eine RS232 Verbindung zwischen PC und ML507-Board aufgebaut

- **GPIO**
  Die Einstellung des Kanalmodells erfolgt über General-Purpose Input-/Output- (GPIO-) Ports, welche mit dem IOM-Board verbunden sind
- **DDR2 RAM**
  Speicher, welcher für die Ethernet-Übertragung der grafisch darzustellenden Signale in Matlab bereitgestellt wird
- **Baseband Transmission SysGen**
  Der zu implementierende Hardwareentwurf des Systems wird mittels System Generator entworfen, mehr dazu im Abschnitt 5.3
- **ADC/DAC Interface**
  Die Umsetzer auf dem ADC/DAC-Board, kommunizieren mittels Serial Peripheral Interface (kurz SPI). Dies ist ein serielles Bussystem. Das ADC Interface wandelt den seriellen Datenstrom vom Board in ein paralleles Wort, welches im System Generator Core verarbeitet werden kann. Das DAC Interface wandelt hingegen das parallele Wort vom System Generator Core in einen seriellen Datenstrom für den DAC-Umsetzer.

5.2 Bestehendes System

In der Bachelor-Thesis eines früheren Studenten der HAW wurde eine digitale Übertragungsstrecke im Basisband auf dem ML507-Board implementiert. Dabei wurden einige Komponenten realisiert, welche für diese Thesis übernommen werden können. Bei diesen handelt es sich unter anderem um den Entzerrer, das Kanalmodell und das ADC- bzw. DAC-Interface. Es soll eine kurze Beschreibung dieser Komponenten folgen. Für eine detaillierte Ausführung wird jedoch auf die Bachelor-Thesis [2] verwiesen.

Der Entzerrer wurde sequentiell erstellt, damit verschiedene Ordnungen (1...16) untersuchen werden können. Dies hat zum Nachteil, dass der Entzerrer nach einer Trainingssequenz in der sich dieser adaptiv einstellt, mit festen Filterkoeffizienten weiterläuft. Eine in der Praxis häufig angewandte Alternative bietet jedoch der Übergang in einen sogenannten Decision-Directed-Mode. In diesem Modus wird nach der Trainingssequenz das Entzererausgangssignal mit dem Entscheiderausgang verglichen. So passen sich die Koeffizienten auch nach der Trainingssequenz weiter an. Voraussetzung dabei ist, dass nach der Trainingssequenz eine ausreichend gute Adaption stattgefunden hat. In der Thesis stellte sich heraus, dass für die gegebenen Kanalverzerrungen bereits ein Entzerrer ab 2. Ordnung ausreichend ist. Auf Seite 48 der Thesis [2] ist ein Simulink-Modell des Entzerrers 3. Ordnung abgebildet, welcher als Vorlage für die Implementierung dient.

Das Kanalmodell wurde mit passiven Bauelementen auf einer Punktraster-Europlatine realisiert. Das Hinzu- oder Abschalten der Tiefpässe bzw. des Hochpasses geschieht über Relais, welche durch das IOM-Board geschaltet werden. Den schematischen Aufbau zeigt Abbildung 5.5.

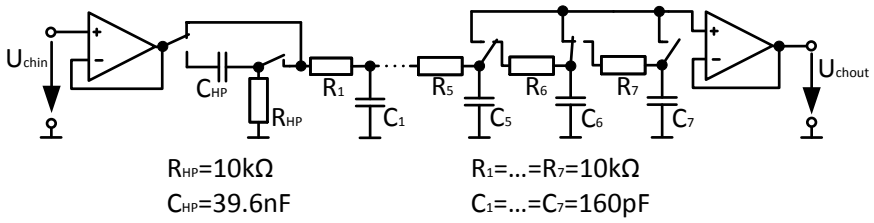


Abbildung 5.5: Schematischer Aufbau des Kanalmodells Quelle [2]

Das ADC- und DAC-Interface muss lediglich doppelt instanziiert werden, da neben dem Übertragungskanal noch das Audio-Interface jeweils einen ADC bzw. DAC benötigt. Die Übertragungskennlinien der Umsetzer sind in Abbildung 5.6 abgebildet.

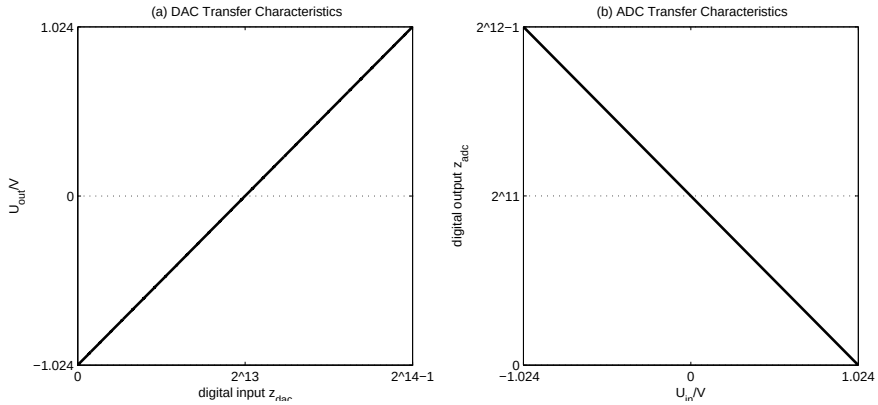


Abbildung 5.6: Übertragungskennlinien der Umsetzer Quelle [2]

5.3 System Generator Core

Mit den bereits bestehenden Systemkomponenten kann das Gesamtsystem zusammengefasst werden (vgl. Abbildung 5.7 auf der folgenden Seite). Die blau umrandeten Blöcke werden zusammen als custom IP-Core *Baseband Transmisson SysGen* implementiert (vgl. Abbildung 5.4). Die Umsetzung erfolgt dabei durch die Xilinx System Generator Toolbox in Simulink, welche in diesem Abschnitt genauer beschrieben wird. Die oberste Ebene des Modells zeigt Abbildung 5.8. Die Blöcke *Transmitter*, *Receiver* und *Clock domain* sind mit sogenannten Gateways verbunden. Diese ermöglichen eine Verbindung zu anderen IP-Cores.

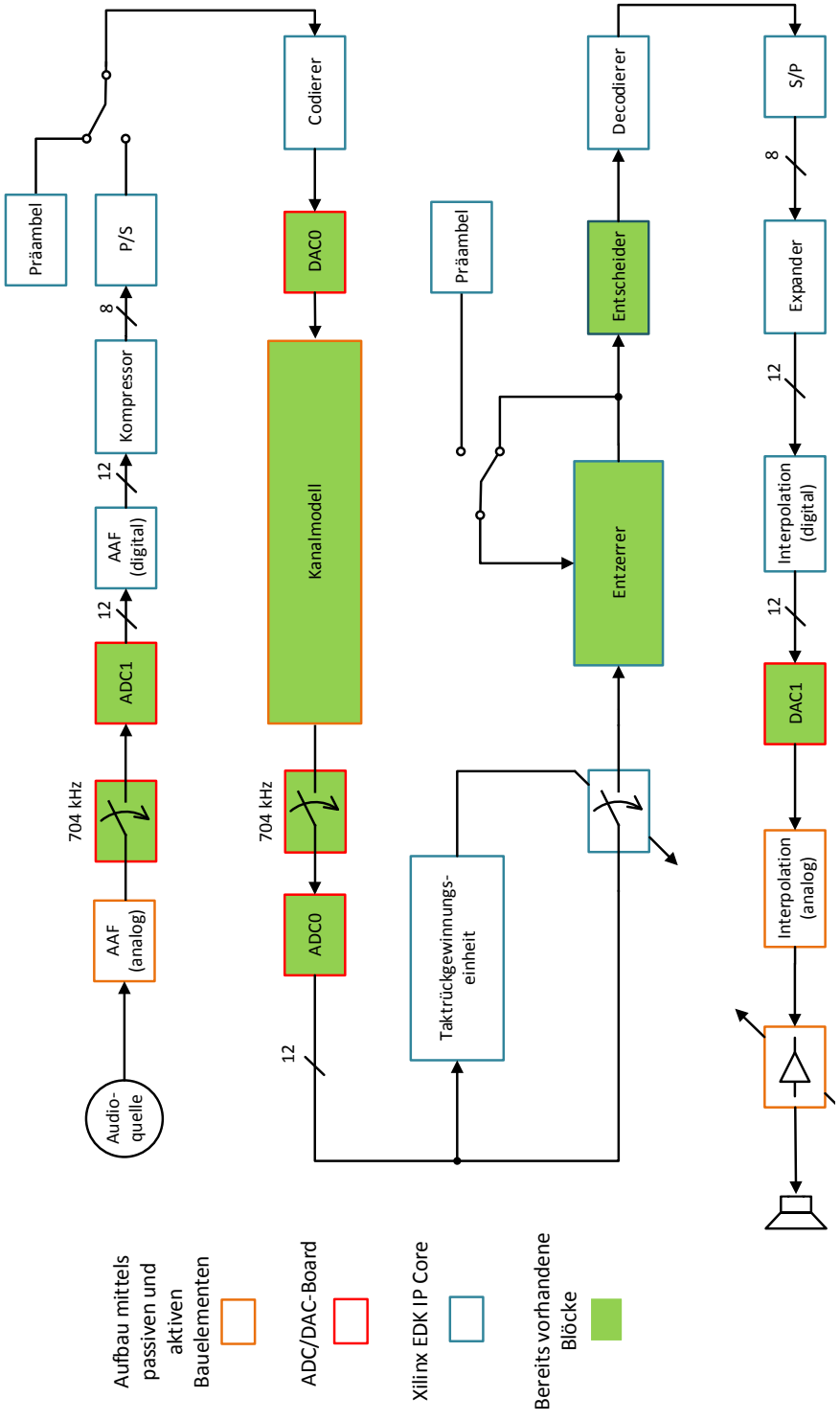


Abbildung 5.7: Blockschaltbild des Gesamtsystems

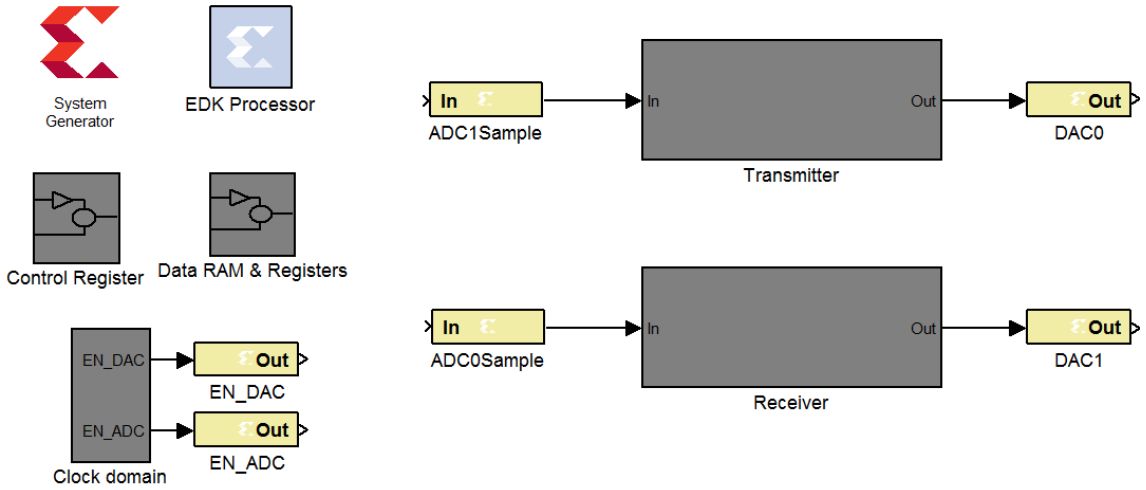
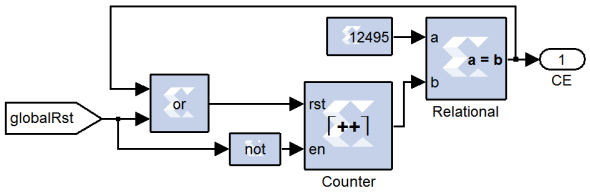


Abbildung 5.8: System Generator Gesamtmodell

Der Block *Clock domain* erzeugt verschiedene Enable Signale, welche in Tabelle 5.1 aufgelistet sind. Um diese untereinander mit steigender Systemtaktflanke synchron zu erzeugen, muss die Abtastfrequenz von $f_A = 704\ kHz$ gewählt werden. Um diese zu erzeugen wird ein Counter mit dem Systemtakt von $f_{sys} = 100\ MHz$ hochgezählt. Das Ausgangssignal wird mittels Vergleicher erzeugt (vgl. Abbildung 5.9). Dieses wird für eine Systemtaktperiode auf High-Pegel gehalten.



Freq.	Systemticks	Genutzt im
8 kHz	12495	Sender
64 kHz	1561	Sender
704 kHz	141	Sender, Empfänger

Tabelle 5.1: Enable Signale

5.3.1 Sender

Der Senderblock bekommt vom ADC1 Samples im Abtasttakt $f_A = 704\ kHz$. Diese werden für die weitere Verarbeitung in ein 1Q11-Format umgewandelt. Das Q-Format stellt im Gegensatz zur Gleitkommadarstellung eine Festkommadarstellung dar, welche zunehmend für digitale Hardware verwendet wird [25]. In diesem Fall wird ein ganzzahliger Wert mit 1 Bit dargestellt und die Nachkommastellen mit 11 Bit. Dieser Block konnte ebenso der Bachelor-Thesis [2] entnommen werden.

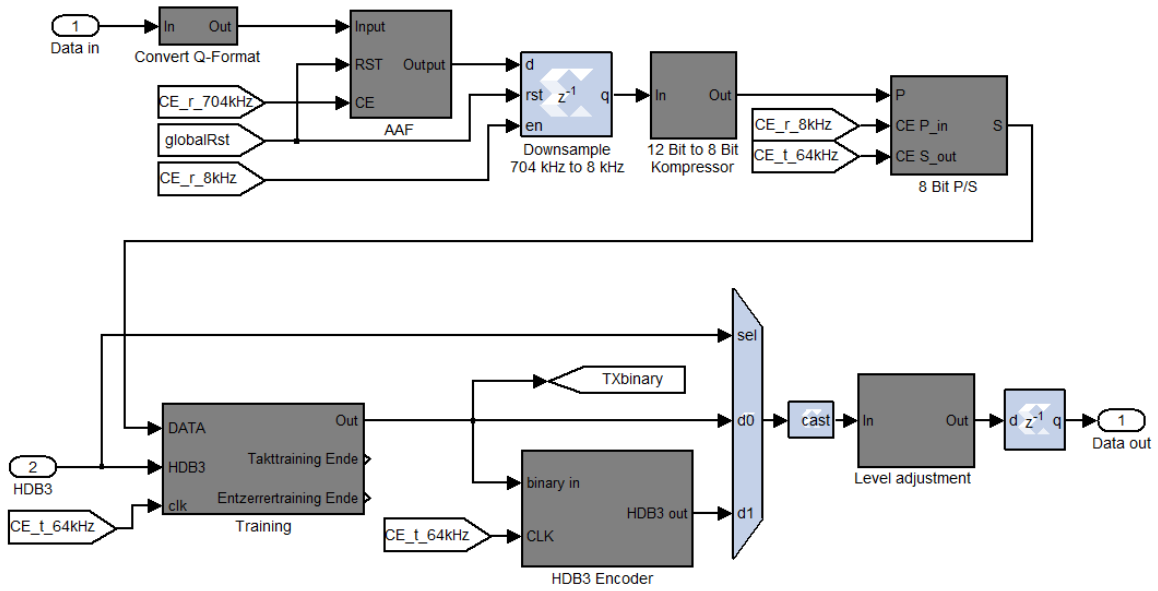


Abbildung 5.10: Transmitter

Darauf folgt das Anti-Aliasing-Filter, welches die gleiche Struktur wie das Simulink Simulationsmodell aufweist (vgl. Abbildung 4.3 auf Seite 68). Eine Abbildung des System Generator Modells ist im Anhang D zufinden. Das gefilterte Signal wird nun mit dem nachgeschalteten Register auf 8 kHz downgesampelt. Der Kompressor ist mittels Matlab-Script-File realisiert, welche dem Anhang B zu entnehmen ist. Der Parallel-/Seriell-Umsetzer wurde bereits im Kapitel 4 erläutert und ist als System Generator Modell ebenso dem Anhang E beigefügt. Dem seriellen Datenstrom wird nun eine Präambel vorgeschaltet. Die Erzeugung dieser zeigt Abbildung 5.11.

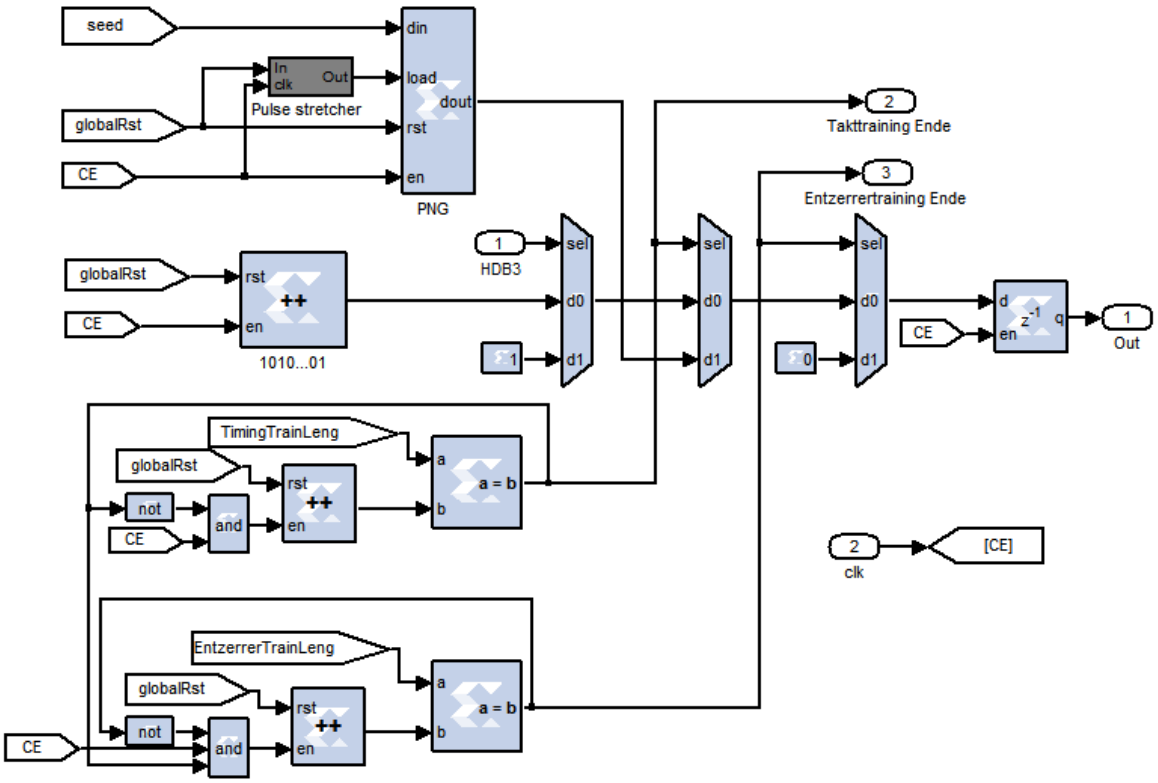


Abbildung 5.11: Präambel

Bei Programmstart ist das Signal *globalRst* auf aktiv high-Pegel und setzt alle Counter auf Null. Mit fallender Flanke des *globalRst* Signals startet die Übertragung. Der Counter *1010...01* beginnt mit dem Takt *CE* von $f_0 = 64\ kHz$ hochzuzählen. Dieser hat eine Bitbreite von 1 Bit. Dabei werden mit jedem zweiten Takt Überläufe erzeugt, sodass eine $1010...01$-Folge als Trainingsphase für die PLL entsteht. Damit bei einer HDB3-Codierung die Trainingsequenz identisch ist, wird eine konstante eins-Folge gesendet. Ein weiterer Counter zählt parallel den *CE* Takt mit. Bei erreichen der Größe *TimingTrainLeng* schaltet ein Multiplexer auf das Signal des Pseudo-Zufallsgenerator *PNG*. Dieser dient dem Training des Entzerrers. Mit erreichen der Trainingslänge *EntzerrerTrainLeng* schaltet schließlich ein Vergleicher einen weiteren Multiplexer auf den seriellen Datenstrom um. Der weitere Signalverlauf des *Transmitters* beinhaltet den HDB3-Encoder, welcher in Kapitel 3.3 bereits erläutert wurde und auch identisch implementiert ist. Der Block *Level adjustment* erzeugt ein paralleles 16-Bit Wort um den *DAC0* anzusteuern.

5.3.2 Receiver

Der Empfänger erzeugt aus dem *ADC0* Sample ein Q-Format. Diesem wird mit dem Xilinx Toolblock *White Gaussian Noise Generator* digital erzeugtes Rauschen überlagert (vgl. Abbildung 5.13). Mit dem Steuersignal *noise* kann die Rauschleistung eingestellt werden.

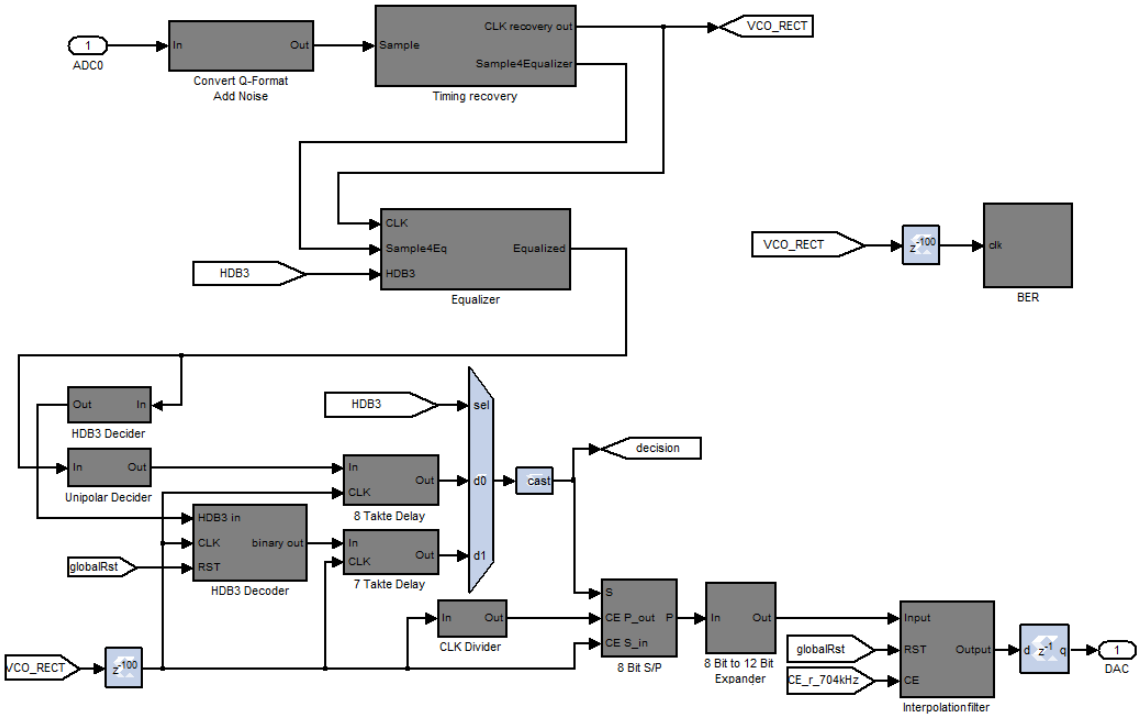


Abbildung 5.12: Receiver

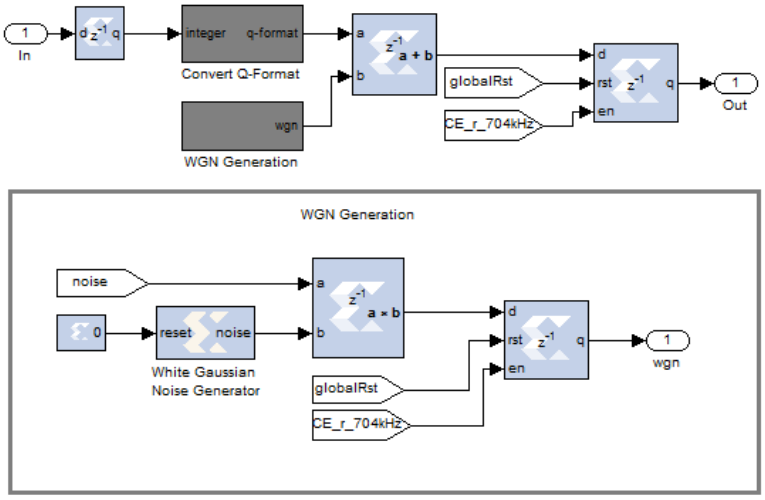


Abbildung 5.13: Empfangssignal mit Überlagerung von bandbegrenztem weißen Rauschen

Der Virtex-5 besitzt ein PLL-Modul, welches für eine Taktsynchronisierung genutzt werden kann [26]. Die PLL kann jedoch nicht mit den Parametern dimensioniert werden, mit welchen der Entwurf in Simulink aus Kapitel 3.2 erstellt wurde. Daher wird äquivalent zum Simulink-Modell der Taktrückgewinnungseinheit das System Generator Modell aufgebaut (vgl. Abbildung 3.16 auf Seite 49 und Abbildung 5.15). Mit dem Signal *chooseLF* kann ein Loopfilter und damit die Ordnung der PLL bestimmt werden. Abbildung 5.14 zeigt das Loopfilter 2. Ordnung.

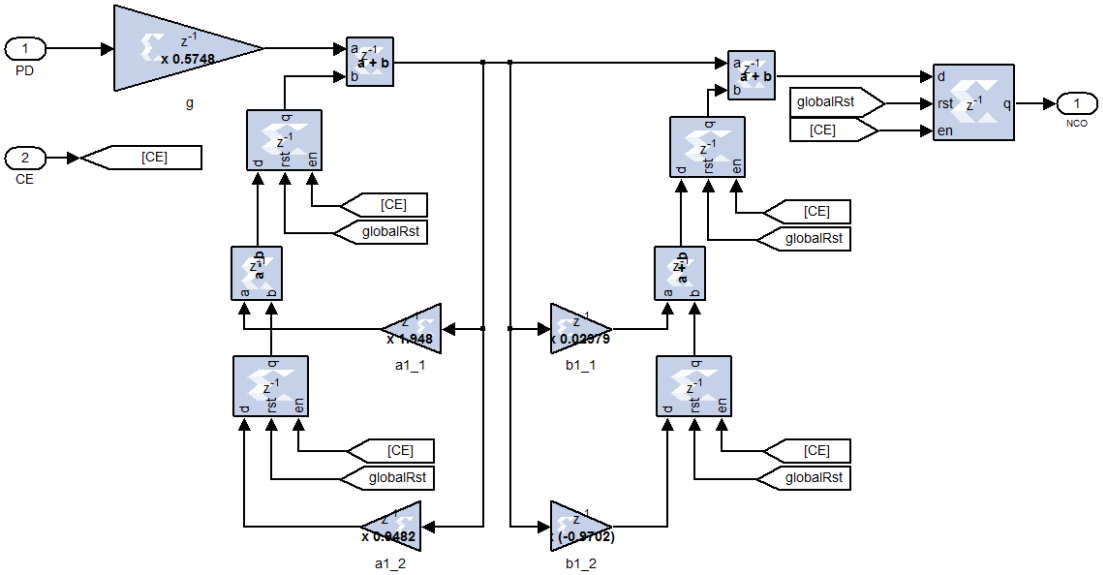


Abbildung 5.14: System Generator Modell des Loopfilters 2. Ordnung

Das Signal *sampleDelay* stellt die Taktquelle des Empfängers ein. Dabei kann neben dem zurückgewonnenen Takt auch der kanalspezifische, ideale Takt ausgewählt werden. Da die Hardware numerisch arbeitet, wird aus dem VCO ein NCO (Numerically Controlled Oscillator), welcher in Abbildung 5.16 abgebildet ist.

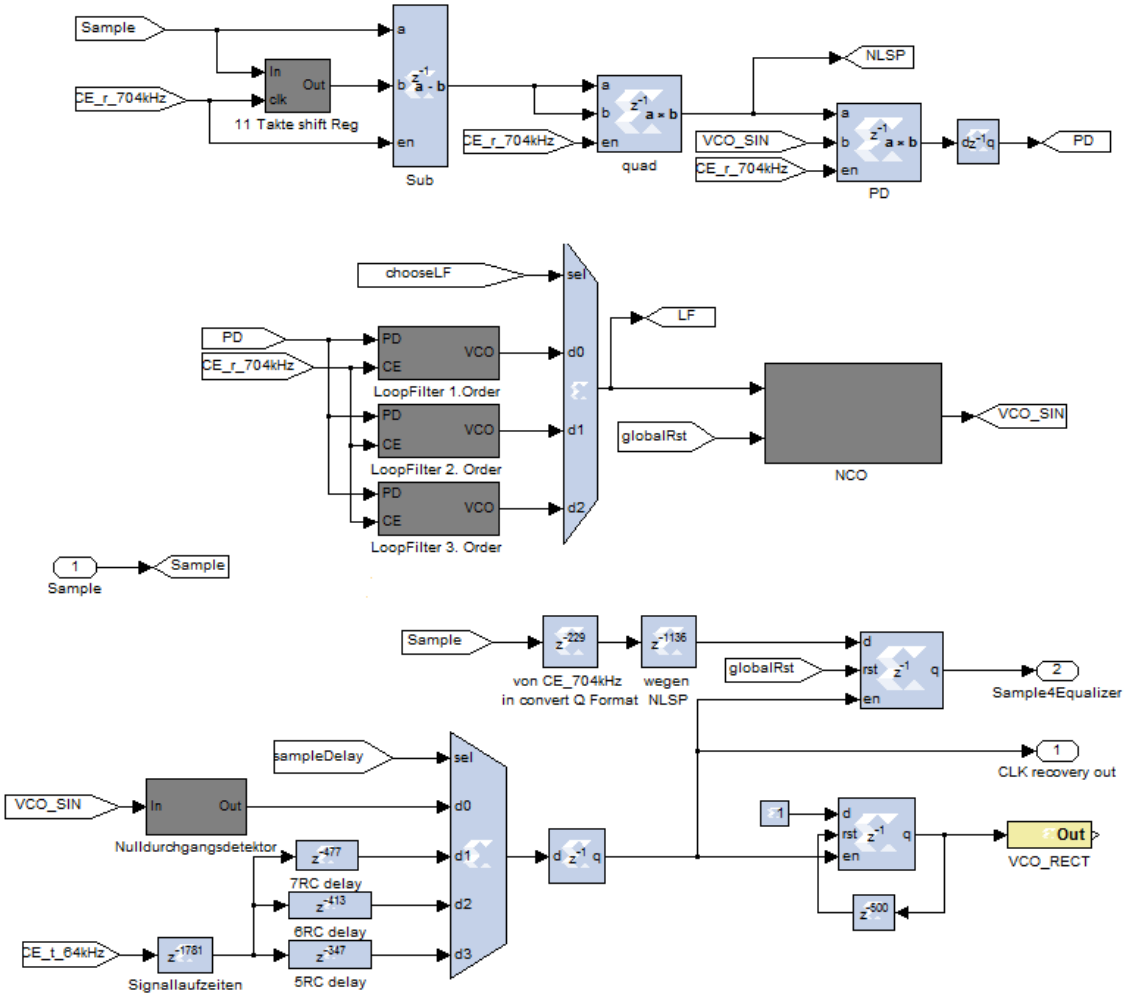


Abbildung 5.15: Taktrückgewinnungseinheit

Mit einem Nulldurchgangsdetektor wird aus dem Ausgangssignal des NCO ein Taktsignal erzeugt. Das Sample vom ADC0 wird mit diesem Takt über einen Register erneut abgetastet und dem Entzerrer zugeführt. Das Gateway *VCO_RECT* dient dabei für externe Messungen des zurückgewonnenen Takts.

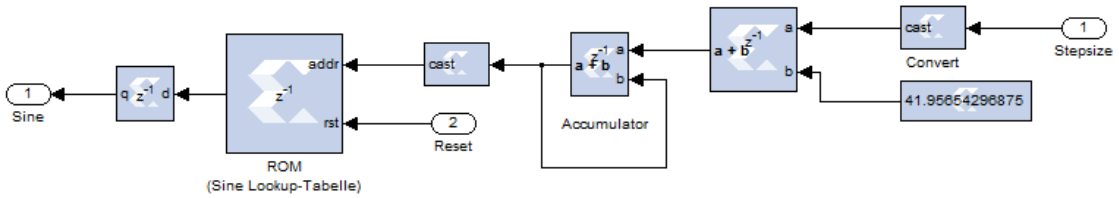


Abbildung 5.16: System Generator Modell des NCO

Der Sinus wird durch eine Lookup-Tabelle mit der Auflösung von $N = 16\ Bit$ erzeugt. Die Mittefrequenz $f_0 = 64\ kHz$ kann mit dem numerischen Wert

$$N_{f0} = \frac{f_0 \cdot 2^N}{f_{sys}} = 41,956... \tag{5.1}$$

angesteuert werden. Der Loopfilterausgang wirkt additiv auf den Wert N_{f0}, womit die Ausgangsfrequenz des NCO variiert. Der Accumulator besitzt eine Nachkommaauflösung von $L = 11\ Bit$. Daraus folgt eine Frequenzauflösung von

$$f_{min} = \frac{f_{sys}}{2^N \cdot 2^L} = 0.745\ Hz. \tag{5.2}$$

Dem Entzerrer wird zu Beginn der Übertragung eine Präambel (zum Sender identisch) angefügt, wobei die Trainingssequenzen durch den Takt der Taktrückgewinnungseinheit gespeist sind. Mit Beenden der Entzerrer Trainingssequenz schaltet das Signal *TrainingEntzerrerEnd* ein Multiplexer um, sodass der Entzerrer von nun an im Decision-Directed-Mode arbeitet. Dabei ist der Entzerrerausgang mit dem Entscheider (Leitungscode spezifisch) gekoppelt.

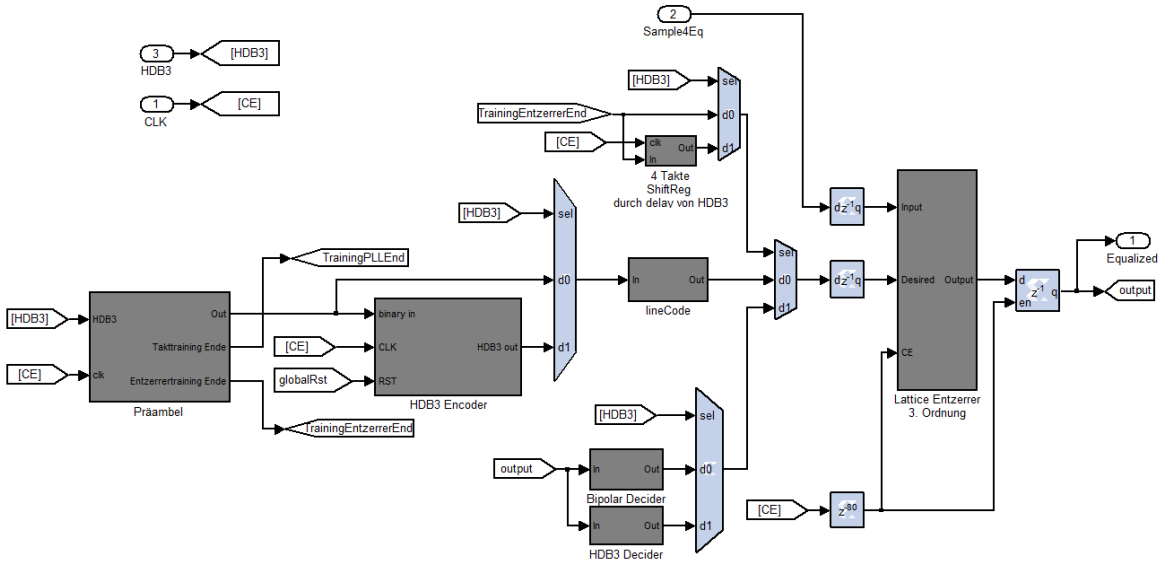


Abbildung 5.17: System Generator Modell des Entzerrers

Nachdem im Empfänger decodiert wurde, muss ein spezifischer Delay erfolgen. Dies ist notwendig, damit aus dem Seriell-/Parallel-Umsetzer wieder das korrekte 8-Bit Wort erscheint. Die Ausgangsfrequenz des Seriell-/Parallel-Umsetzers wird über den *Clock Divider* erzeugt. Dabei wird der Eingangstakt um den Faktor 8 herabgesetzt. Dies geschieht mit einem Counter, welcher bei stiegender Flanke hochzählt und bei jeder 8. Taktflanke ein Ausgangssignal erzeugt. Der Seriell-/Parallel-Umsetzer sowie der darauf folgende Expander und das Interpolationsfilter ist dem Anhang D, C und F zu entnehmen.

Für eine BER-Messung wird das Sendersignal *TXbinary* verzögert, sodass dies bei fehlerfreier Übertragung taktsynchron mit dem Decoderausgangssignal *decoded* verglichen werden kann. Dies geschieht üblicherweise mit einer xor-Logikverknüpfung. Ein Accumulator zählt die gemessenen Bits. Der Bitfehler-Counter beginnt erst zu zählen, wenn die Präambel bzw. das Entzerrertraining beendet worden ist (vgl. Abbildung 5.18).

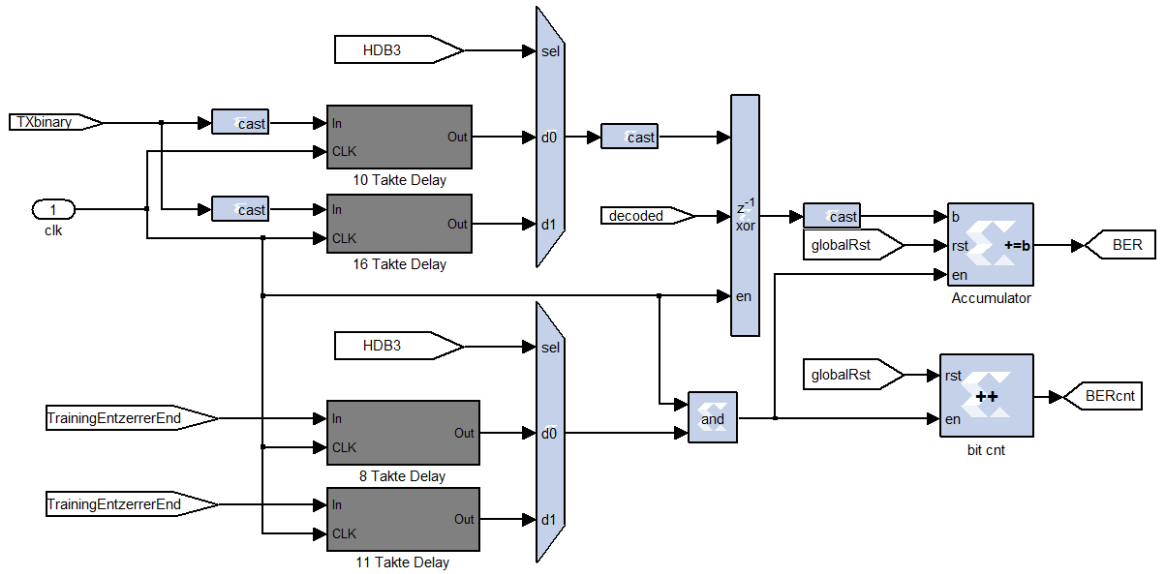


Abbildung 5.18: Messung der Bitfehlerrate

Die Steuerung des Systems geschiet über Registerblöcke welche in Abbildung 5.19 zusammengefasst sind. Ein globaler Reset kann sowohl durch ein softwareseitig gesetztes Register, als auch über einen externen Taster am ML507-Board, ausgelöst werden. Bei Übertragung mit HDB3-Codierung muss der Trainingssequenzlänge jeweils vier Takte abgezogen werden, da der nachgeschaltete Encoder eine Laufzeit von vier Takten besitzt.

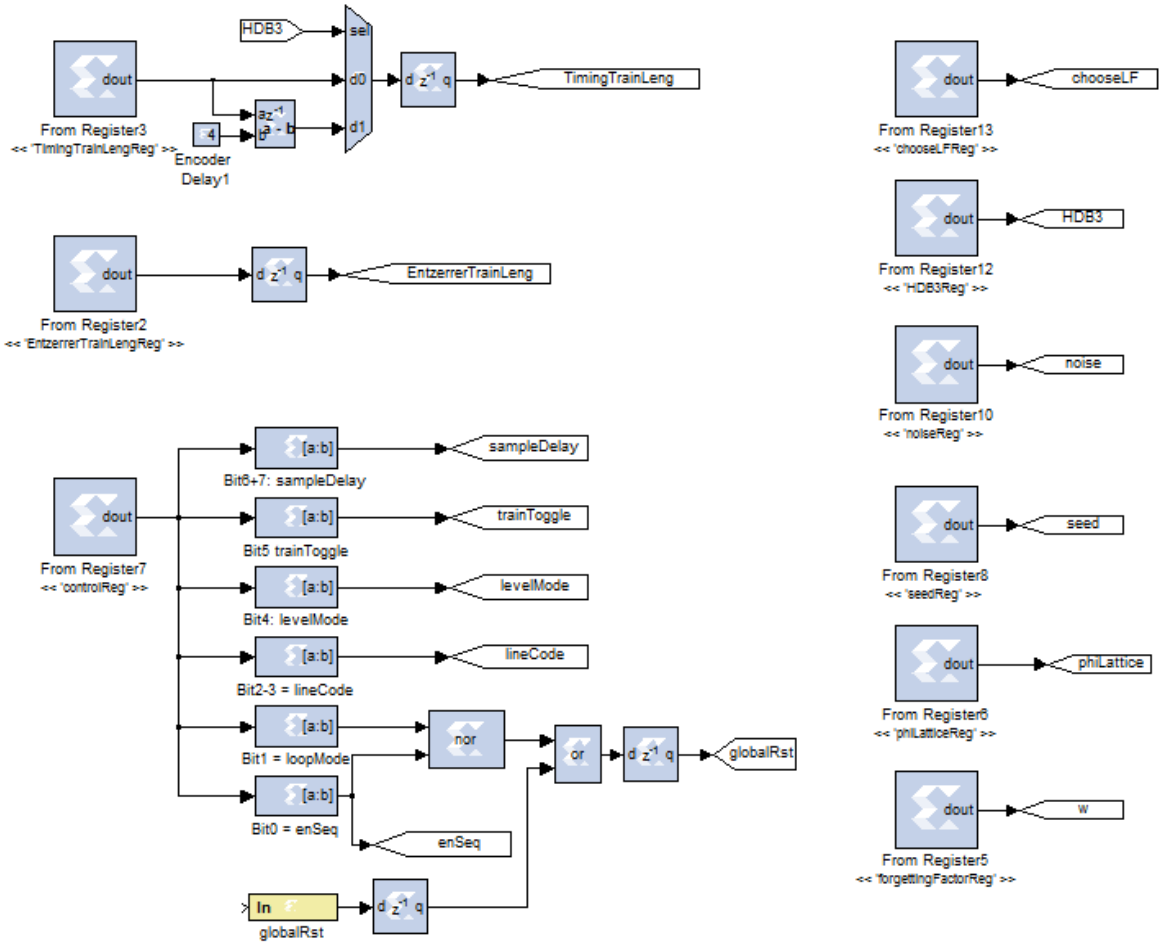


Abbildung 5.19: Control Register

Xilinx bietet sogenannte Shared Memorys, womit Signale zwischengespeichert werden können um diese dann an Matlab zu übermitteln zu können. Die in Abbildung 5.20 dargestellten Signale werden mit 2048 Abtastwerten der Frequenz $f_A = 704\ kHz$ und einer Auflösung von 16 Bit gespeichert. Die Adressansteuerung mit dem Signal *addr_shmem* wurde aus der Bachelor-Thesis [2] übernommen. Für die Bitfehlerrate sowie den Counter der gesendeten Bits werden jeweils zwei 16-Bit Register verwendet.

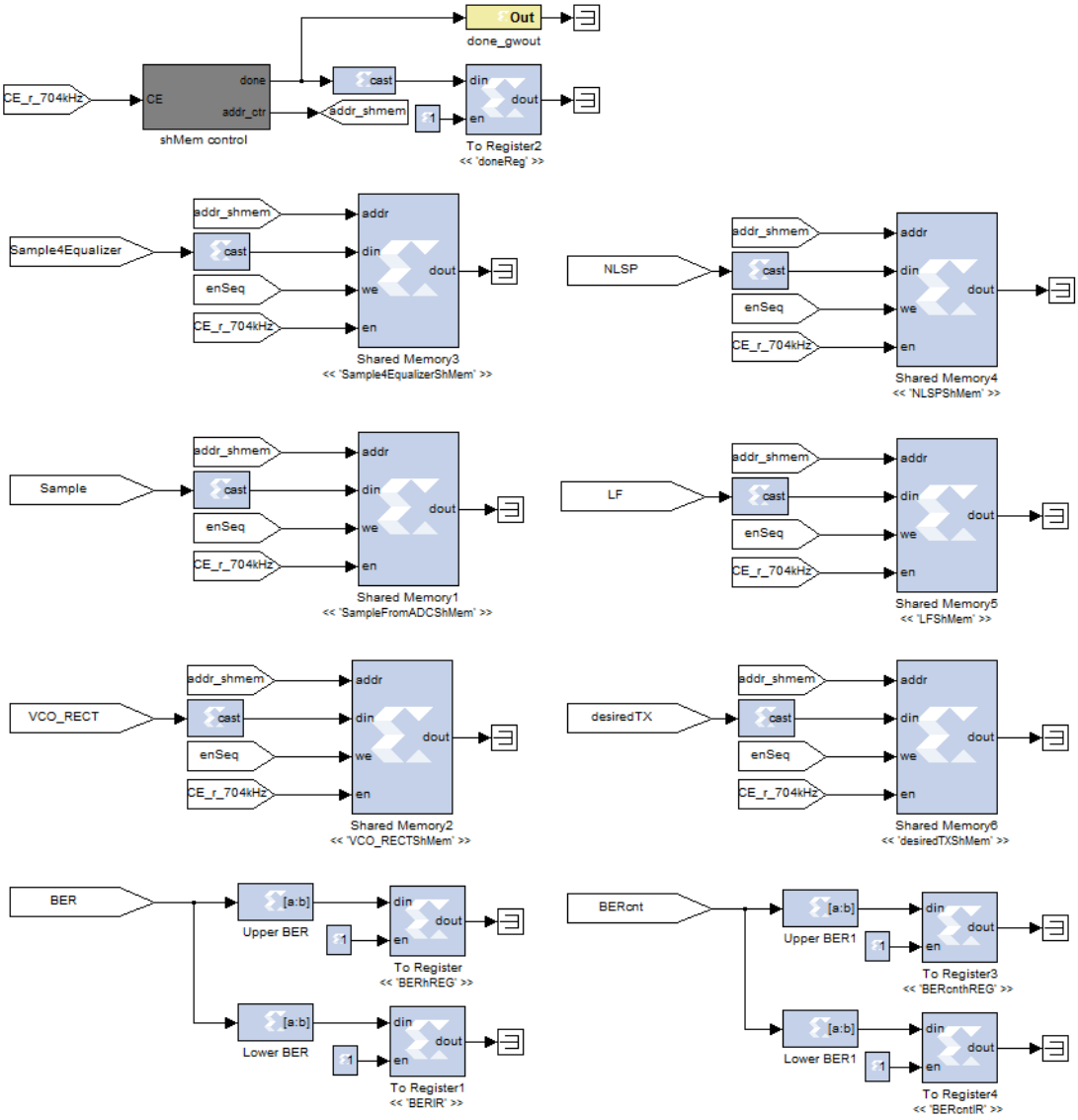


Abbildung 5.20: Shared Memory

5.4 Firmware

Die Firmware bildet die zentrale Steuereinheit des Embedded Systems und wird mit dem Xilinx Software Development Kit (SDK) in der Programmiersprache C erstellt und auf dem MicroBlaze programmiert. Die Firmware in der Bachelor-Thesis [2] diente für die Implementierung dieses Systems als Vorlage. Da sich im Wesentlichen lediglich die Shared Memorys und Control Register unterscheiden, wird auf das Flussdiagramm referenziert.

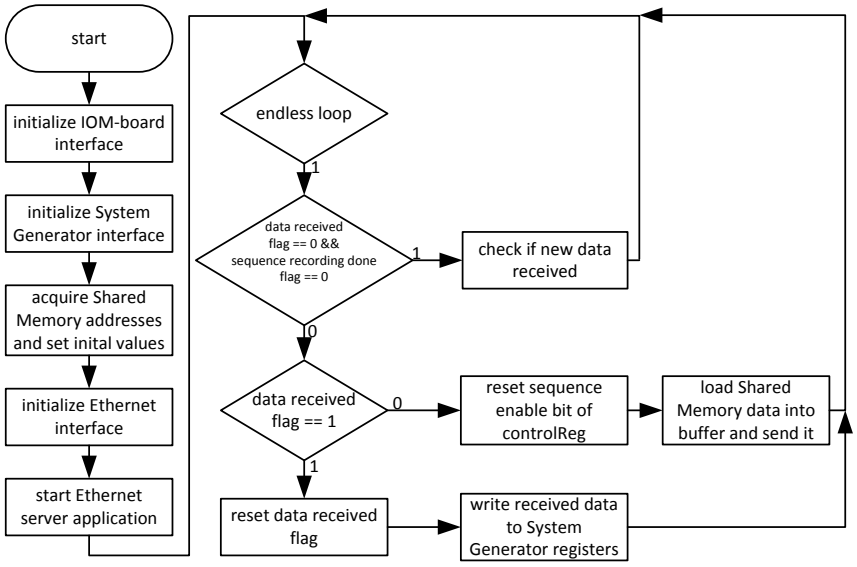


Abbildung 5.21: Flussdiagramm der Firmware Quelle [2]

Bei Programmstart werden zunächst die veschiedenen IP-Cores initialisiert. Hierfür bietet das SDK vorgefertigte Treiberbibliotheken. Die Endlosschleife prüft, ob vom Ethernet-Interface Steuerungsparameter empfangen werden. Ist dies der Fall, so werden die Parameter dem System Generator Modell übertragen und das System gestartet. Ist der Shared Memory Speicher voll, werden die Daten über Ethernet an die Matlab GUI übertragen.

5.5 Graphical User Interface (GUI)

Über die in Matlab programmierte GUI lässt sich das System Generator Modell ansteuern. Dieses bietet unter anderem Auswahlmöglichkeiten für Loopfilter Ordnung, Trainingssequenzlängen, Leitungscode und Kanalmodell. Die Shared Memory Signale können über zwei Plotfenster angezeigt werden. Mit einem Pop-up kann zwischen einer Single-Übertragung (Dauer knapp 3 ms) zur Auswertung der Signale und einer dauerhaften Audioübertragung gewählt werden. Um die Übertragungsqualität des Systems bestimmen zu können, kann eine Messung der Bitfehlerhäufigkeit (BER-Bit Error Rate) durchgeführt werden.

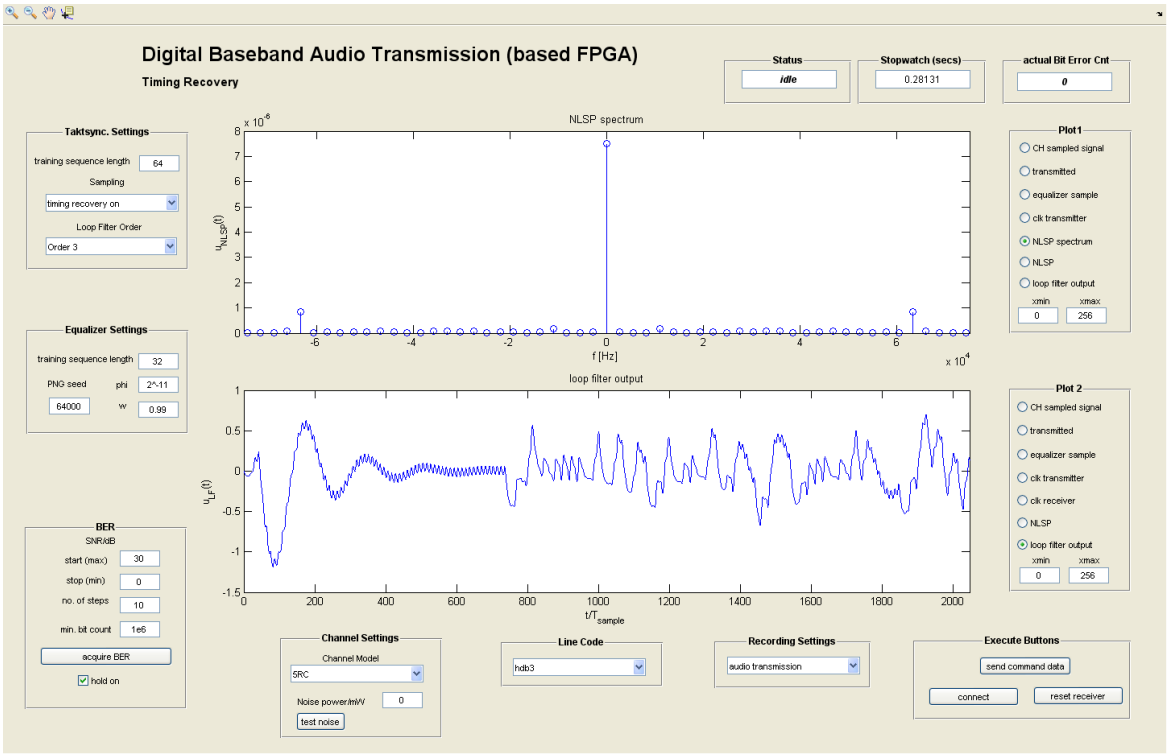


Abbildung 5.22: Grafische Benutzeroberfläche

Für die BER gilt allgemein

$$BER = \frac{\textit{Fehlerhafte Bits}}{\textit{Gesendete Bits}}. \tag{5.3}$$

Da dem gesendeten Signal während der Übertragung Rauschen überlagert ist, wird die BER überlicherweise in Abhängigkeit zum Signal-zu-Rausch-Verhältnis dargestellt. Für dieses gilt

$$SNR = 10 \cdot log_{10}\left(\frac{P_S}{P_N}\right), \tag{5.4}$$

wobei P_S die Leistung des gesendeten Signals und P_N die Leistung des Rauschsignals darstellt. Vor jeder BER-Messung wird eine Übertragung ohne Rauschen durchgeführt, um die Signalleistung P_S ermitteln zu können. Für diese gilt theoretisch

$$P_S = E[u_S^2(kT_A)], \tag{5.5}$$

wobei $u_S(kT_A)$ das Sender-Ausgangssignal darstellt. Da der Erwartungswert für eine unendliche Folge definiert ist, lässt sich die Leistung nur über eine endliche Anzahl von M-gesendeter

Bits abschätzen [8].

$$\hat{P}_S = \frac{1}{M-1} \sum_{k=0}^{M-1} u_S^2(kT_A) \tag{5.6}$$

Mit diesem Schätzwert lässt sich mit Gleichung 5.4 die Rauschleistung für ein vorgegebenes SNR berechnen.

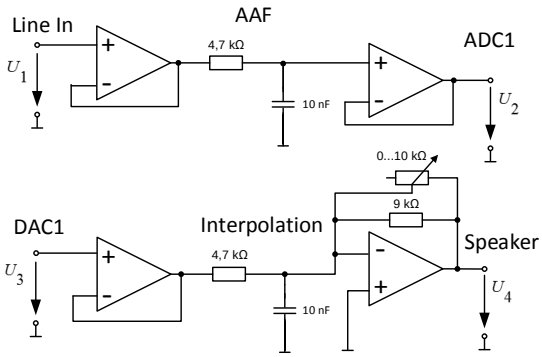
$$P_N = \frac{\hat{P}_S}{10^{SNR/10}} \tag{5.7}$$

5.6 Anti-Aliasing- und Interpolations-Filter

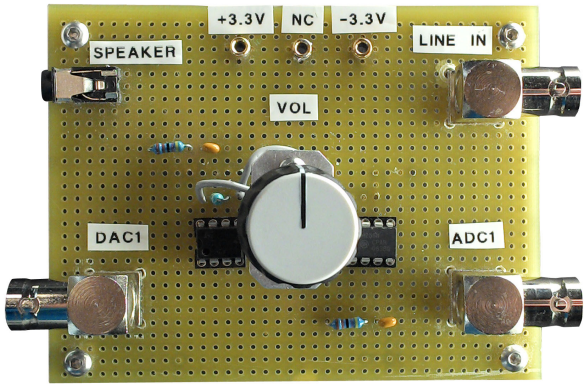
Die analogen Filter für Anti-Aliasing und Interpolation sind zusammen auf einer Lochrasterplatine realisiert. Zudem ist dem Interpolationsfilter ein invertierender Verstärker mit dem Verstärkungsfaktor

$$v = -\frac{\frac{R_v \cdot 9\ k\Omega}{R_v + 9\ k\Omega}}{4,7\ k\Omega} = 0... -1, \quad mit\ R_v = 0...10\ k\Omega \tag{5.8}$$

nachgeschaltet. Mit diesem kann die Lautstärke reguliert werden. Der Schaltplan sowie die Realisierung der Platine ist in Abbildung 5.23 abgebildet.



(a) Schematisches Blockschaltbild der Platine



(b) Realisierung auf einer Lochrasterplatine

Abbildung 5.23: Platine mit AAF, Interpolationsfilter und einstellbarer Verstärkung

Eine Verstärkung der Audioquelle ist nicht nötig da ein Line-Out eines mp3-Players durchaus Spitzenwerte von $U_{pp} = 2\ V$ erreichen kann [27]. Dies reicht für eine Aussteuerung des ADC aus. Eine Messung der Ausgangsspannung eines mp3-Players bei voller Lautstärke bestätigt dieses (vgl. Abbildung 5.24).

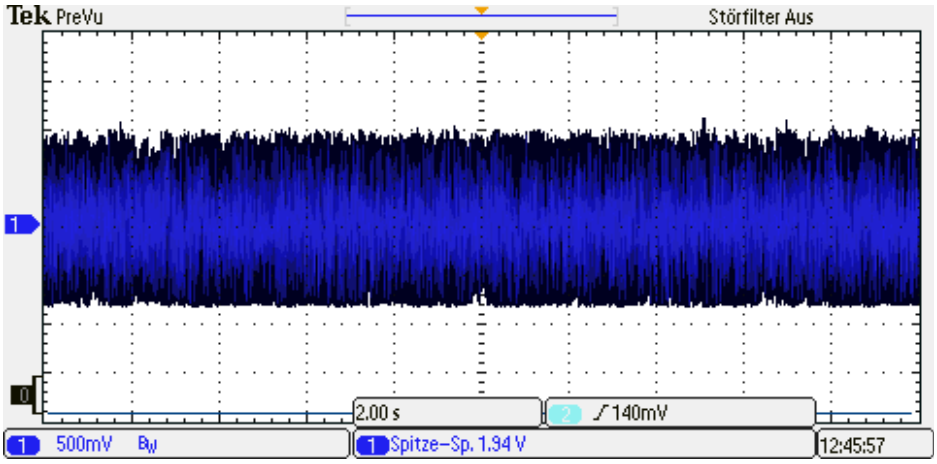


Abbildung 5.24: Ausgangssignal eines iPod nano 5G

6 Inbetriebnahme

In diesem Kapitel soll die Implementierung verifiziert werden. Mit dem Xilinx Platform Studio (XPS) wird aus dem erstellten Gesamtsystem ein Bitstream generiert, welcher auf dem FPGA implementiert wird. Während der Generierung wird geprüft, ob beim Routen der Signale Timing Probleme entstanden sind. Diese resultieren z.B. durch kritische Pfade aus kombinatorischer Logik. Die Timing Analyse zeigt, dass kein Signalpfad besteht, welcher länger als die minimale Systemtaktperiode von 9.966 ns andauert (vgl. Abbildung 6.1).

```
Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 1012969034 paths, 16 nets, and 176796 connections

Design statistics:
   Minimum period:   9.966ns   (Maximum frequency: 100.341MHz)
   Maximum path delay from/to any node:   7.446ns
   Maximum net delay:   0.835ns
   Minimum input required time before clock:   1.514ns
```

Abbildung 6.1: Post-PAR (Place and Route) Static Timing Report

Nach der Programmierung des FPGAs mit dem Xilinx SDK konnten via serieller Schnittstelle erfolgreich initialisierte Cores sowie Shared Memorys ausgelesen werden. Eine TCP-Ethernetverbindung konnte über die GUI erfolgreich hergestellt werden. Das System wird mit einer HDB3-Codierung und dem 7RC+HP Kanalmodell gestartet. Abbildung 6.2 zeigt das Sendesignal mit der Präambel. Die drei Stufen des HDB3-Codes sind deutlich zu erkennen.

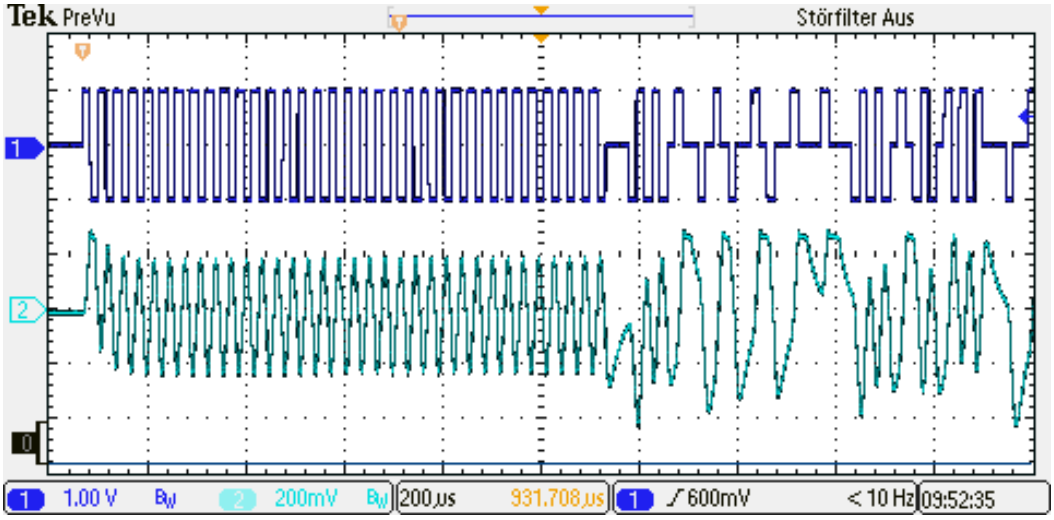


Abbildung 6.2: Ch.1: HDB3-codiertes Sendesignal Ch.2: 7RC+HP Kanalausgang

Der Plot in der Matlab GUI zeigt den identischen Signalverlauf wie die Messung am Oszilloskop zeigt (vgl. Abbildung 6.3).

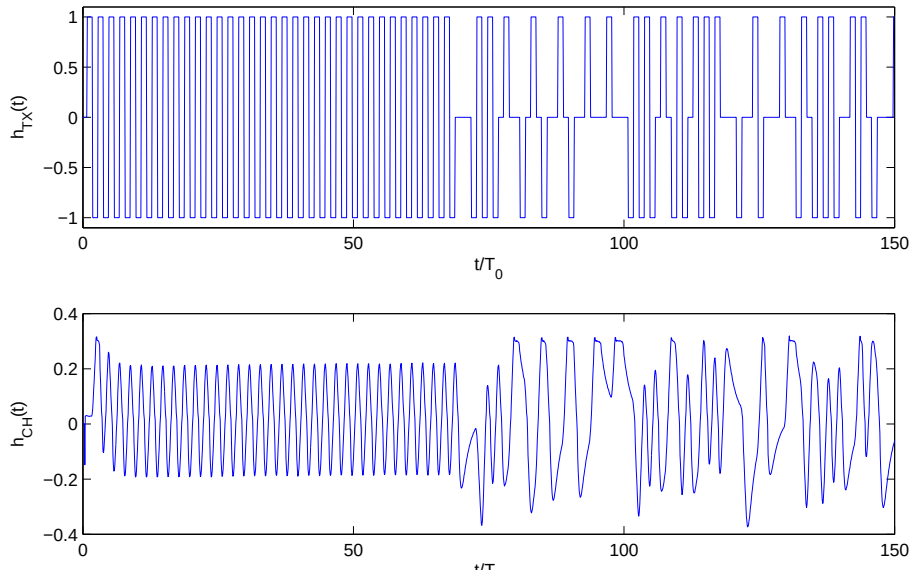


Abbildung 6.3: HDB3-codiertes Sendesignal und die Kanalantwort des 7RC+HP

Des Weiteren zeigt die GUI das Ausgangssignal $h_{NLSP}(t)$ der nichtlinearen Operation zur Taktrückgewinnung. Das Spektrum zeigt einen diskreten Anteil an der Taktfrequenz $f_0 = 64\ kHz$.

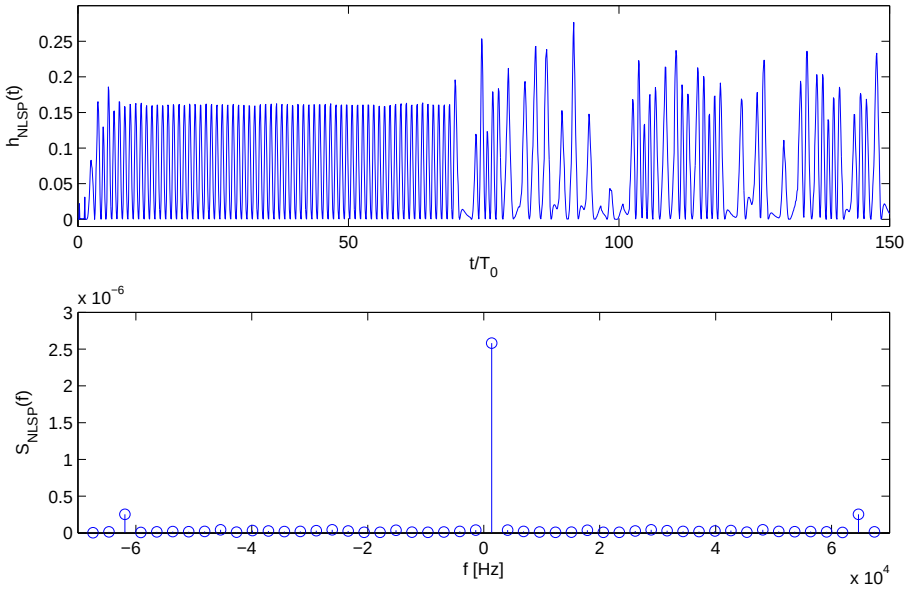


Abbildung 6.4: Ausgangssignal der nichtlinearen Signalverarbeitung und dessen LDS

Der Loopfilterausgang zeigt das Regelverhalten der PLL.

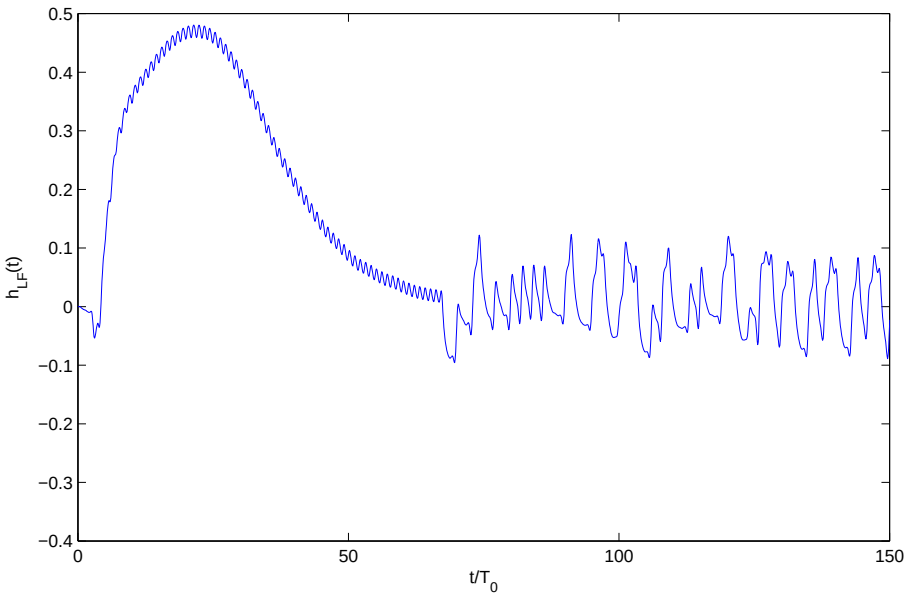


Abbildung 6.5: Loopfilterausgang

Nun soll geprüft werden, ob bei einer Audioübertragung das gesendete Signal rekonstruiert werden kann. Ch. 1 der Oszilloskopaufnahme in Abbildung 6.6 zeigt einen gesendeten Sinus. Ch. 3 zeigt den Interpolationsfilterausgang. Es ist zu erkennen, dass der Sinus nach der Präambel wieder korrekt empfangen wird.

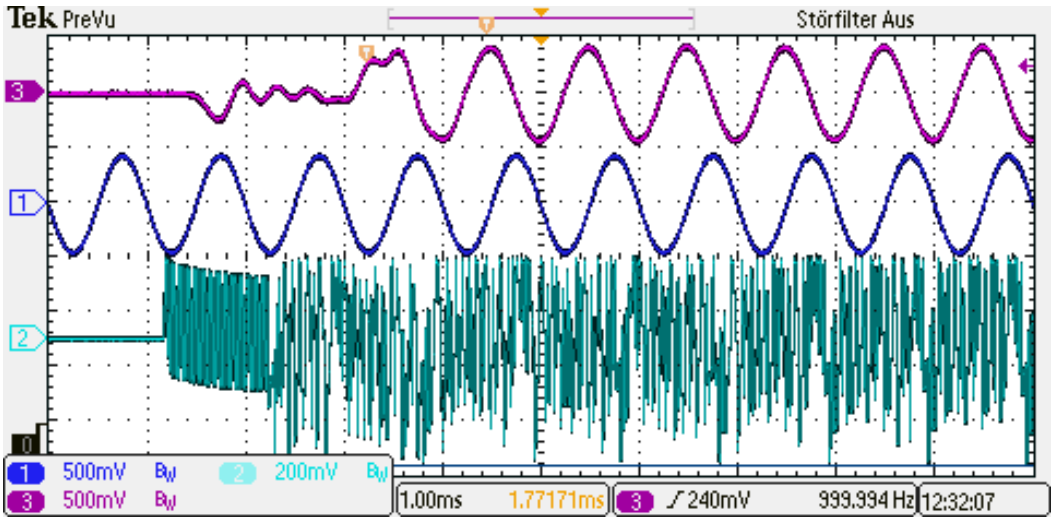


Abbildung 6.6: Ch. 1: Sender: Sinus mit $f_0 = 1kHz$, Ch. 2: 7RC+HP Kanalausgang, Ch. 3: Interpoliertes Empfangssignal

Anhand des Ausgangssignals der Taktrückgewinnung, welches über das IOM-Board nach außen geführt ist, kann der Taktjitter ermittelt werden. Hierfür wird die Nachleuchtzeit erhöht. Das 5RC Kanalmodell zeigt einen Jitter von $T_J = 4,35\ \mu s$, welcher bereits knapp 28 % der Taktperiodendauer ist. Gesendet wurde mit einem HDB3-Code.

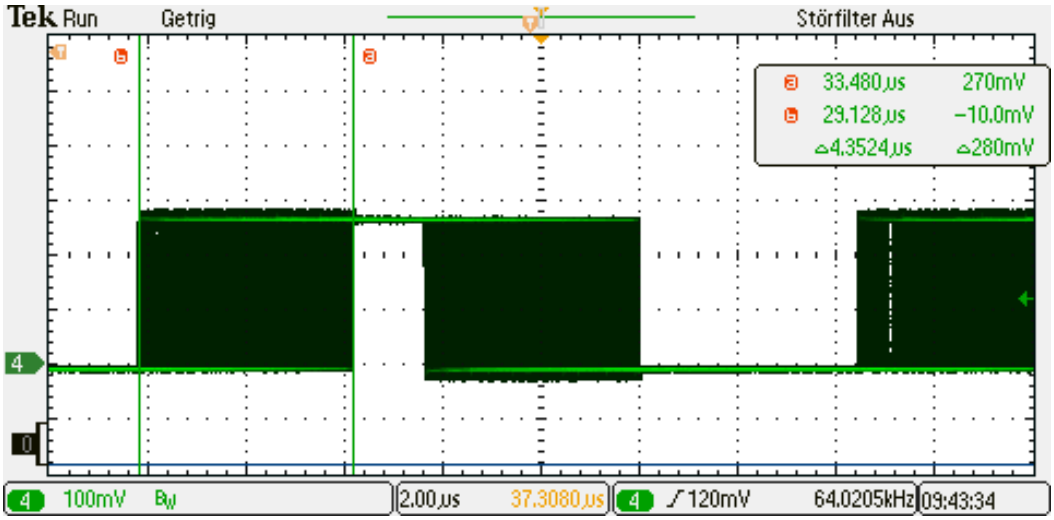


Abbildung 6.7: Ausgangssignal der Taktrückgewinnungseinheit beim 5RC Kanalmodell

Eine Messung am 7RC Kanalmodell zeigt hingegen einen Jitter von $T_J = 472\ ns$, welcher 3 % der Taktperiode ausmacht.

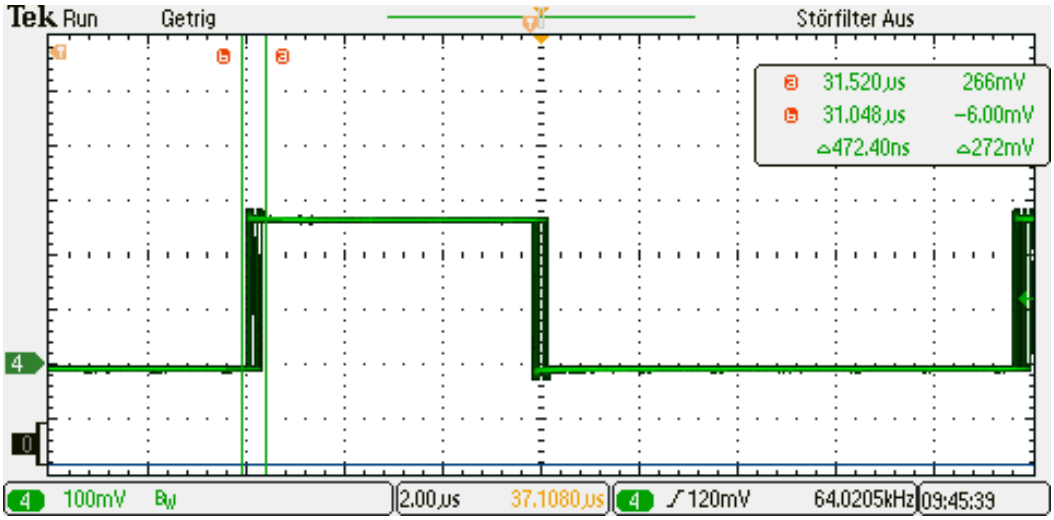
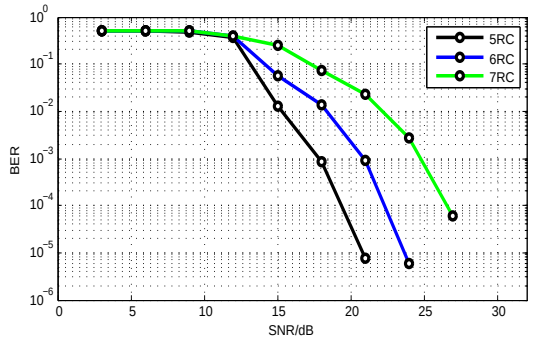


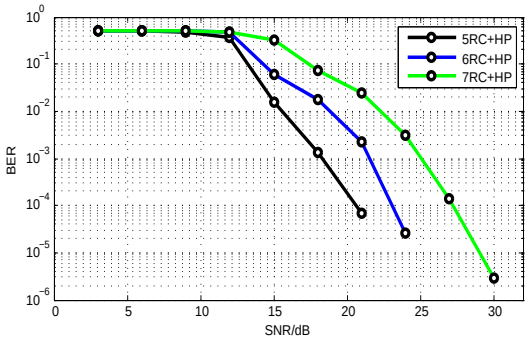
Abbildung 6.8: Ausgangssignal der Taktrückgewinnungseinheit beim 7RC Kanalmodell

Eine Hörprobe ist vergleichbar mit Telefonqualität. Höhen und Tiefen werden durch den begrenzten Frequenzbereich gedämpft wahrgenommen.

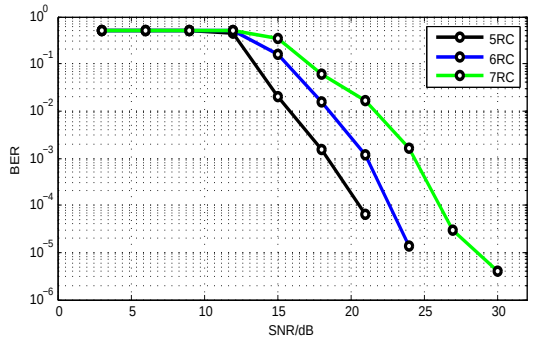
Schlussendlich sollen BER-Messungen die Übertragungsqualität unter Rauscheinfluss darstellen. Eine Messung wird jeweils über $M = 10^6$ Bits, ab einen SNR von 30 dB, in 3 dB Schritten durchgeführt. Zu Vergleichszwecken erfolgt zusätzlich eine BER-Messung mit optimalem Abtastzeitpunkt.



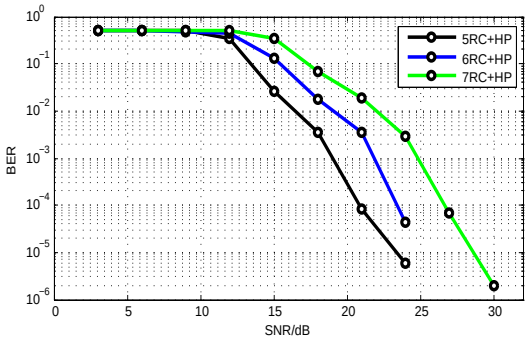
(a) ohne Hochpass, optimale Abtastung



(b) mit Hochpass, optimale Abtastung



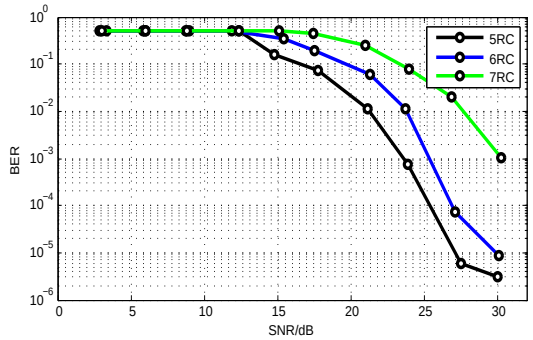
(c) ohne Hochpass, mit Taktrückgewinnung



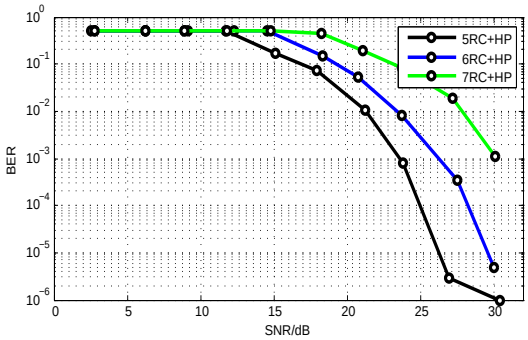
(d) mit Hochpass, mit Taktrückgewinnung

Abbildung 6.9: BER Messungen bei bipolarer NRZ-Codierung, unterschiedlichen Kanalmodellen sowie mit und ohne Taktrückgewinnung

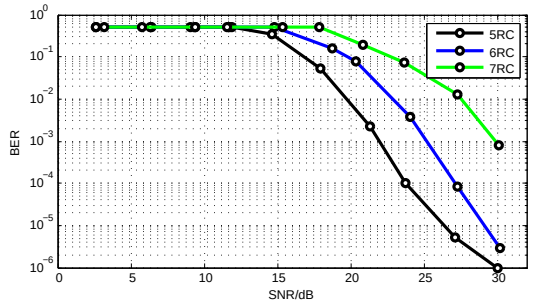
Erwartungsgemäß ist erkennbar, dass Kanalmodelle höherer Ordnung eine erhöhte Bitfehlerrate aufweisen. Hat ein Kanalmodell zusätzlich Hochpasscharakteristik, hat dieses ebenso einen Anstieg der BER zur Folge. Die Kanalmodelle 5. und 6. Ordnung verursachen jedoch bei einem SNR von größer gleich 27 dB gar keine Bitfehler. Der Vergleich zwischen optimaler Abtastung und der Abtastung mittels Taktrückgewinnung zeigt, dass letztere nur eine minimale Verschlechterung der Bitfehlerrate mit sich zieht.



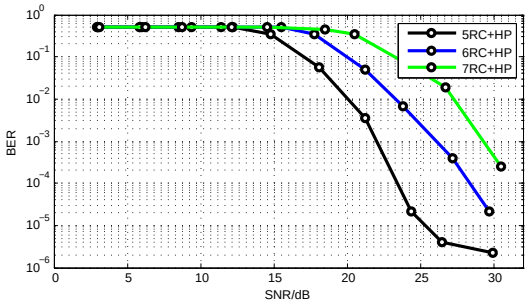
(a) ohne Hochpass, optimale Abtastung



(b) mit Hochpass, optimale Abtastung



(c) ohne Hochpass, mit Taktrückgewinnung



(d) mit Hochpass, mit Taktrückgewinnung

Abbildung 6.10: BER Messungen bei HDB3-Codierung, unterschiedlichen Kanalmodellen sowie mit und ohne Taktrückgewinnung

Die Bitfehlerhäufigkeit mit HDB3-Codierung ist bei jeder Messung merklich höher als beim bipolaren NRZ-Code. Dieses spricht gegen die Erwartungen, da die HDB3- sowie auch die AMI-Codierung mit dem Hintergrund der Kanalanpassung durchgeführt wurden. Die Kanalmodelle scheinen zu starke Verzerrung zu haben, als dass die Vorteile der Codierung zum Tragen kommen kann. Eine Verbesserung könnte sich durch Erhöhung der Sendeleistung ergeben. Die HDB3-Codierung scheint jedoch im Vergleich zwischen optimaler Abtastung zur Abtastung mit der Taktrückgewinnung einen positiven Einfluss auf die Bitfehlerhäufigkeit zu haben, als es beim bipolaren NRZ-Code der Fall ist. Dies liegt daran, dass der HDB3-Code mehr Synchronisationsinformation besitzt. Zudem zeigt die Hochpassanbindung hier keine Verschlechterung.

7 Schlussbetrachtung

7.1 Zusammenfassung

Mit dem realisierten System wurde eine Audioübertagungsstrecke entwickelt, welche zunächst anhand theoretischer Modelle in Simulink überführt und simuliert wurde. Darauf folgte eine Implementierung in Hardware, welche durch die Inberiebnahme verifiziert wurde. In diesem Prozess wurden unterschiedliche Stufen behandelt, die analoge und digitale Signalverarbeitung erforderten. Eine Gegenüberstellung mit den in der Spezifikation gestellten Anforderungen zeigt, dass keine Unstimmigkeiten festgestellt werden können. Somit gilt diese als erfüllt.

7.2 Ausblick

Das entwickelte System kann für zukünftige Bachelor-Thesen weiterentwickelt werden. Daher wird abschließend ein Ausblick der möglichen Weiterentwicklungen aufgestellt.

- **Automatic Gain Control (AGC)**
  Die Taktrückgewinnung ist auf Kanalmodelle mit starker Dämpfung ausgelegt. Erfährt das übertragene Signal keine oder nur sehr geringe Dämpfung, so wirkt sich dieses direkt auf den Regelkreis der PLL aus. Diese regelt zu stark nach und kann folglich dem Eingangssignal nicht mehr folgen. Eine automatische Verstärkungsregelung (AGC) kann hier Abhilfe schaffen. Diese faktorisiert das Eingangssignal mit einem automatisch ermittelten Wert, sodass die Amplitude einen nahezu konstanten Maximalpegel nicht überschreitet.
- **Vollduplex-Übertragung**
  Das Gesamtsystem bietet lediglich eine Simplex-Übertragung. Um ein bidirektionales Kommunikationssystem im Sinne der Telefonie zu erstellen, wird jedoch eine Vollduplex-Verbindung benötigt. Diese wird über eine Zweidrahtleitung realisiert. Demnach bedarf es eines zweiten Kanalmodells. Zudem muss nun unter Teilnehmern und nicht mehr zwischen Sender und Empfänger unterschieden werden, da ein Teilnehmer beides beinhaltet. Die Implementierung könnte auf zwei FPGAs umgesetzt werden.

- **ISDN Protokoll**
  Das implementierte System ist angelehnt an ISDN. Ein wesentlicher Unterschied besteht darin, dass keine Rahmenstruktur bzw. kein Protokoll besteht. ISDN besitzt mit der $S_0 - Schnittstelle$ eine Multiplexübertragung, welche zwei B-Kanäle als Nutzkanal und einen D-Kanal zur Signalisierung besitzt. Die $S_0 - Schnittstelle$ stellt damit einen komplexen Rahmen dar, an welchen viele verschiedene Anforderungen gestellt sind. Die hinzugefügte Redundanz muss im Empfänger korrekt erkannt und klar von den Nutzdaten getrennt werden. Zudem erhöht sich die Sendedatenrate.

Literaturverzeichnis

[1] WIKIPEDIA: *Integrated Services Digital Network*. http://de.wikipedia.org/wiki/Integrated_Services_Digital_Network#Geschichtliche_Entwicklung. Version: 2013. – [Online; Aufruf am 20.09.2013]

[2] GÄTH, Michael: *Entwurf, Simulation und Implementierung auf einem FPGA von einem digitalen Basisband-Übertragungssystem*. 2013

[3] MAGAZIN, Projekt: *Projekt Spezifikation und Pflichtenheft*. https://www.projektmagazin.de. Version: 2013. – [Online; Aufruf am 18.09.2013]

[4] XILINX: *ML505/ML506/ML507 Evaluation Platform*. http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf. Version: 2011. – [Online; Aufruf am 22.09.2013]

[5] KAMMEYER, Karl-Dirk: *Nachrichtenübertragungstechnik*. Vieweg+Teubner, 2011. – ISBN 978–3–8348–0896–7

[6] LOCHMANN, Dietmar: *Digitale Nachrichtentechnik*. Verlag Technik Berlin, 2002. – ISBN 3–341–01321–0

[7] ITU, Telecommunication Standardization S.: *G.712 - Digital terminal equipments - Coding of analogue signals by pulse code modulation*. http://www.itu.int/rec/T-REC-G.711. Version: 1993. – [Online; Aufruf am 02.09.2013]

[8] KAMMEYER, Karl-Dirk ; KROSCHEL, Kristian: *Digitale Signalverarbeitung*. Vieweg+Teubner, 2009. – ISBN 978–3–8348–0610–9

[9] MICHEEL, Hans J.: *Vorlesungsskript Modulation*. 2011

[10] MICHEEL, Hans J.: *Vorlesungsskript Grundlagen Nachrichtentechnik*. 2009

[11] KOVINTAVEWAT, Piya ; BARRY, John: *Iterative Timing Recovery: A Per-Survivor Approach*. VDM Verlag, 2009. – ISBN 978–3639205633

[12] BARRY, John R. ; LEE, Edward A. ; MESSERSCHMITT, David G.: *Digital Communication*. Springer, 2004. – ISBN 978–1–4613–4975–4

[13] BEST, Roland E.: *Phase Locked Loops*. McGraw Hill, 2007. – ISBN 978–0–07149926–2

[14] WIKIPEDIA: *Ethernet*. http://de.wikipedia.org/wiki/Ethernet#Die_Pr.C3.A4ambel_und_SFD. Version: 2013. – [Online; Aufruf am 13.09.2013]

[15] WÖHLKE, Wilfried: *Vorlesungsskript Grundlagen der Regelungstechnik*. 2011

[16] TECHNIK BERLIN, Beuth H.: *Anti-Aliasing-Filter Vorlesungsskript*. http://.beuth-hochschule.de/uploads/media/Filter.pdf. Version: 2013. – [Online; Aufruf am 12.09.2013]

[17] MATHWORKS: *Filter Design and Analysis Tool (FDATool)*. http://www.mathworks.de/de/help/signal/ug/opening-fdatool.html. Version: 2013. – [Online; Aufruf am 12.09.2013]

[18] REICHARDT, Jürgen: *Vorlesungsskript digitale Signalverarbeitung*. 2010

[19] TIETZE, Ulrich ; SCHENK, Christoph: *Halbleiter-Schaltungstechnik*. Springer, 2001. – ISBN 978–3540641926

[20] REICHARDT, Jürgen ; SCHWARZ, Bernd: *VHDL Synthese*. Oldenbourg, 2009. – ISBN 978–3–486–58987–0

[21] PÖLLHUBER, M.: *Moderne Telekommunikation*. http://tkhf.adaxas.net. Version: 2006/2007. – [Online; Aufruf am 05.09.2013]

[22] ITU, Telecommunication Standardization S.: *G.711 - Pulse code modulation (PCM) of voice frequencies*. http://www.itu.int/rec/T-REC-G.711. Version: 1993. – [Online; Aufruf am 02.09.2013]

[23] XILINX: *Virtex-5 FPGA User Guide*. http://www.xilinx.com/support/documentation/user_guides/ug190.pdf. Version: März 2012. – [Online; Aufruf am 17.09.2013]

[24] HAMBURG, HAW: *Labor für Digitaltechnik und Digitale Systeme*. http://www.haw-hamburg.de/en/ti-ie/labore/digitaltechnik-digitale-systeme/download.html. Version: 2013. – [Online; Aufruf am 15.09.2013]

[25] REICHARDT, Jürgen: *Lehrbuch Digitaltechnik*. Oldenbourg, 2009. – ISBN 978–3–486–58908–5

[26] XILINX: *Virtex-5 FPGA User Guide*. http://www.xilinx.com/support/documentation/user_guides/ug190.pdf. Version: 2012. – [Online; Aufruf am 22.09.2013]

[27] WIKIPEDIA: *Line Level*. http://en.wikipedia.org/wiki/Line_level#cite_note-3. Version: 2013. – [Online; Aufruf am 16.09.2013]

[28] NOCKER, Rudolf ; HEDKE, Laurent: *HDB3 Leitungscodierer mit programmierbarem Logikbaustein*. http://f1.hs-hannover.de/fileadmin/media/doc/f1/Aktivitaeten/publikationen/91-elektronik-h12-s74-80-hdb3-encoder.pdf. Version: 1991. – [Online; Aufruf am 12.09.2013]

[29] XILINX: *System Generator for DSP User Guide UG640*. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/sysgen_user.pdf. Version: Oktober 2012. – [Online; Aufruf am 23.09.2013]

Anhang

Anhang A

Matlab Skript File der HDB3 Moore FSM

```
function [STATE,HDB3]= hdb_encoder(H, L, E, POR, CLK_EN)

pos=xfix( {xlSigned, 16, 0}, 1);        % signed fixed point hdb3 output '+'
neg=xfix( {xlSigned, 16, 0}, -1);       % signed fixed point hdb3 output '-'
zero=xfix( {xlSigned, 16, 0}, 0);       % signed fixed point hdb3 output '0'

persistent NEXT_S, NEXT_S = xl_state(0, {xlUnsigned, 5, 0});    % state variable

if POR  % power on reset
        S=0;            % set default state 0
else
        S=NEXT_S;       % get next state
end
% state machine %
    switch S
        case 0
            HDB3=zero;
            if xl_and(CLK_EN, xl_not(POR))
                if xl_or(H, E)
                    NEXT_S=1;
                else
                    NEXT_S=0;
                end
            else
                NEXT_S=0;
            end
        case 1
            HDB3=pos;
            if CLK_EN
                if H
                    NEXT_S=2;
                elseif L
                    NEXT_S=5;
                elseif E
                    NEXT_S=6;
                end
            else
                NEXT_S=1;
            end
        case 2
            HDB3=negs;
            if CLK_EN
                if H
                    NEXT_S=5;
```

```
45                  elseif L
46                      NEXT_S=3;
47                  elseif E
48                      NEXT_S=8;
49                  end
50              else
51                  NEXT_S=2;
52              end
53          case 3
54              HDB3=zero;
55              if CLK_EN
56                  if H
57                      NEXT_S=4;
58                  else
59                      NEXT_S=3;
60                  end
61              else
62                  NEXT_S=3;
63              end
64          case 4
65              HDB3=pos;
66              if CLK_EN
67                  if H
68                      NEXT_S=3;
69                  elseif L
70                      NEXT_S=5;
71                  elseif E
72                     NEXT_S=6;
73                  end
74              else
75                  NEXT_S=4;
76              end
77          case 5
78              HDB3=zero;
79              if CLK_EN
80                  if H
81                      NEXT_S=2;
82                  elseif L
83                      NEXT_S=5;
84                  end
85              else
86                  NEXT_S=5;
87              end
88           case 6
89              HDB3=zero;
90              if CLK_EN
91                  NEXT_S=7;
92              else
93                  NEXT_S=6;
94              end
95           case 7
96              HDB3=zero;
97              if CLK_EN
98                  NEXT_S=8;
99              else
100                NEXT_S=7;
101             end
102         case 8
103            HDB3=zero;
104            if CLK_EN
105                 NEXT_S=9;
106            else
107                NEXT_S=7;
```

```
108             end
109          case 9
110             HDB3=zero;
111             if CLK_EN
112                  NEXT_S=10;
113             else
114                 NEXT_S=9;
115             end
116         case 10
117              HDB3=neg;
118              if CLK_EN
119                   if H
120                        NEXT_S=11;
121                   elseif L
122                        NEXT_S=13;
123                   elseif E
124                      NEXT_S=15;
125                   end
126              else
127                   NEXT_S=10;
128              end
129          case 11
130              HDB3=pos;
131              if CLK_EN
132                   if H
133                        NEXT_S=14;
134                   elseif L
135                        NEXT_S=12;
136                   elseif E
137                      NEXT_S=16;
138                   end
139              else
140                   NEXT_S=11;
141              end
142          case 12
143              HDB3=zero;
144              if CLK_EN
145                   if H
146                        NEXT_S=14;
147                   elseif L
148                        NEXT_S=12;
149                   end
150              else
151                   NEXT_S=12;
152              end
153          case 13
154              HDB3=zero;
155              if CLK_EN
156                   if H
157                        NEXT_S=11;
158                   elseif L
159                        NEXT_S=13;
160                   end
161              else
162                   NEXT_S=13;
163              end
164          case 14
165              HDB3=neg;
166              if CLK_EN
167                   if H
168                        NEXT_S=11;
169                   elseif L
170                        NEXT_S=13;
```

```
171                     elseif E
172                         NEXT_S=15;
173                     end
174                 else
175                     NEXT_S=14;
176                 end
177             case 15
178                 HDB3=pos;
179                 if CLK_EN
180                     NEXT_S=17;
181                 else
182                     NEXT_S=15;
183                 end
184             case 16
185                 HDB3=zero;
186                 if CLK_EN
187                     NEXT_S=17;
188                 else
189                     NEXT_S=16;
190                 end
191             case 16
192                 HDB3=zero;
193                 if CLK_EN
194                     NEXT_S=17;
195                 else
196                     NEXT_S=16;
197                 end
198             case 17
199                 HDB3=zero;
200                 if CLK_EN
201                     NEXT_S=18;
202                 else
203                     NEXT_S=17;
204                 end
205             otherwise
206                     NEXT_S=0;
207                     HDB3=zero;
208         end
209             HDB3 = xfix({xlSigned, 16, 0},HDB3);     % set outup hdb3
210             STATE = NEXT_S;                          % get next state
211     end
```

Anhang B

Matlab Skript File des Kompressors

```
function [OUTPUT_VALUE] = Kompressor12Bit(INPUT_VALUE)

outp = xfix( {xlUnsigned, 8, 0}, 0 );       % Init

if INPUT_VALUE < 0                          % negativ?
        inpAbs=xl_not(INPUT_VALUE)+1;           % calc. absolut value
else                                        % positiv
        inpAbs=INPUT_VALUE;                     % take value
end
if INPUT_VALUE == -2048                         % if 2048 -> carry
    inpAbs=2047;
end
sgn=xfix( {xlUnsigned, 1, 0}, 0 );      % set sgn. as default '+'

% Lookup Table
if inpAbs < 32/2
    prebits=xfix( {xlUnsigned, 3, 0}, 0 );
    value=xl_slice(inpAbs, 3, 0);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
elseif inpAbs < 64/2 && inpAbs > 15
    prebits=xfix( {xlUnsigned, 3, 0}, 1 );
    value=xl_slice(inpAbs, 3, 0);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
elseif inpAbs < 128/2 && inpAbs > 31
    prebits=xfix( {xlUnsigned, 3, 0}, 2 );
    value=xl_slice(inpAbs, 4, 1);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
elseif inpAbs < 256/2 && inpAbs > 63
    prebits=xfix( {xlUnsigned, 3, 0}, 3 );
    value=xl_slice(inpAbs, 5, 2);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
elseif inpAbs < 512/2 && inpAbs > 127
    prebits=xfix( {xlUnsigned, 3, 0}, 4 );
    value=xl_slice(inpAbs, 6, 3);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
elseif inpAbs < 1024/2 && inpAbs > 255
    prebits=xfix( {xlUnsigned, 3, 0}, 5 );
    value=xl_slice(inpAbs, 7, 4);
    outp=xl_concat(sgn, prebits, value);
    if INPUT_VALUE < 0                          % calc. sgn
        outp=xl_not(outp)+1;
    end
```

```
58    elseif inpAbs < 2048/2 && inpAbs > 511
59        prebits=xfix( {xlUnsigned, 3, 0}, 6 );
60        value=xl_slice(inpAbs, 8, 5);
61        outp=xl_concat(sgn, prebits, value);
62        if INPUT_VALUE < 0                          % calc. sgn
63            outp=xl_not(outp)+1;
64        end
65    elseif inpAbs > 1023
66        prebits=xfix( {xlUnsigned, 3, 0}, 7 );
67        value=xl_slice(inpAbs, 9, 6);
68        outp=xl_concat(sgn, prebits, value);
69        if INPUT_VALUE < 0                          % calc. sgn
70            outp=xl_not(outp)+1;
71        end
72    elseif inpAbs >= 2047
73        outp=xfix( {xlUnsigned, 8, 0},  127);
74        if INPUT_VALUE < 0                          % calc. sgn
75            outp=xl_not(outp)+1;
76        end
77    end
78    OUTPUT_VALUE=outp;                    % set output
```

Anhang C

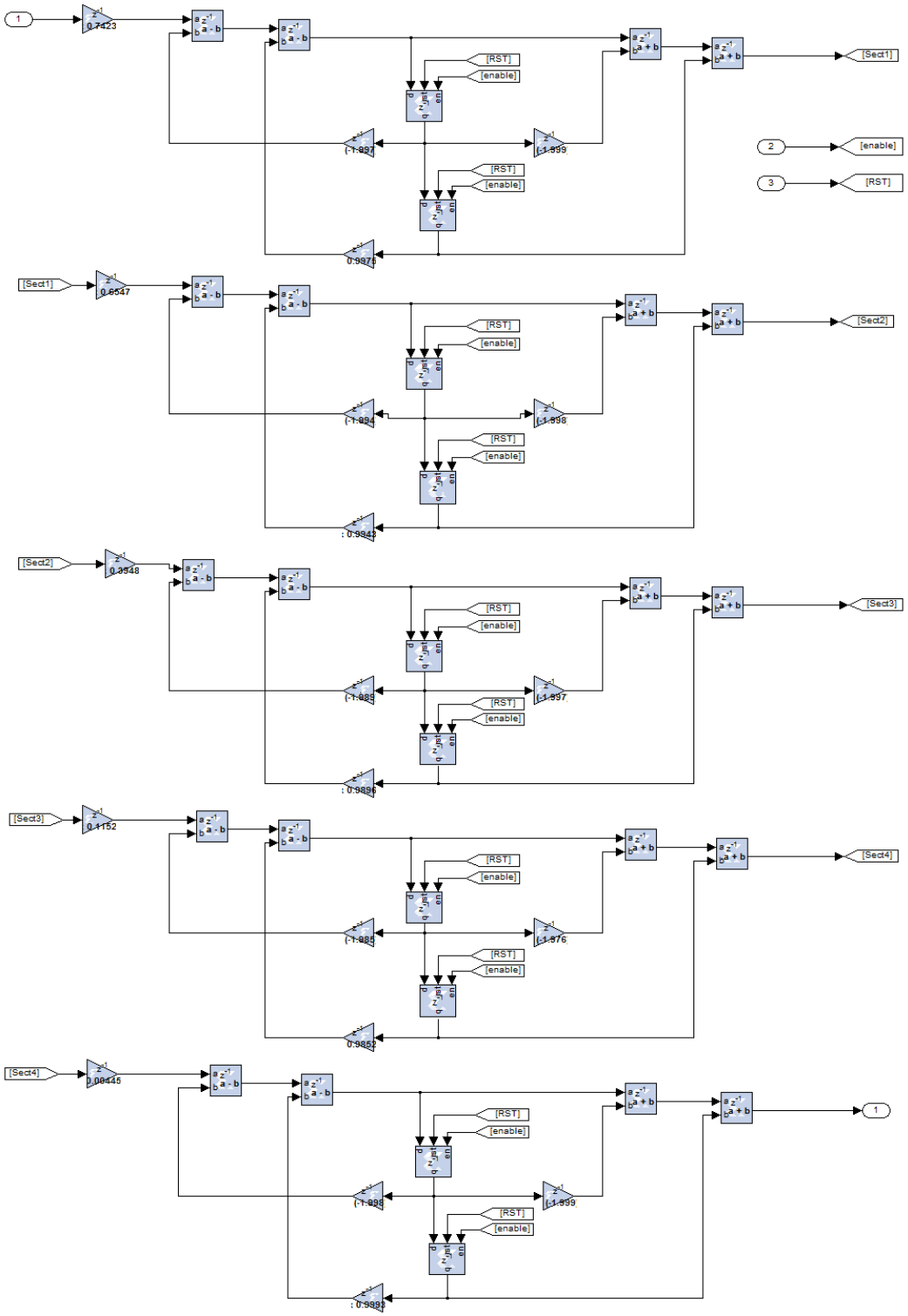
Matlab Skript File des Expanders

```
1  function [OUTPUT_VALUE] = Expander12Bit(INPUT_VALUE)
2
3  outp = xfix( {xlUnsigned, 12, 0}, 0 );       % Init
4
5  if INPUT_VALUE < 0                           % negative?
6      wert=xl_slice(INPUT_VALUE, 6, 0);        % calc. absolut value
7      inpAbs=xl_not(wert)+1;
8  else                                         % positiv
9      inpAbs=xl_slice(INPUT_VALUE, 6, 0);      % take value
10 end
11 sgn=xfix( {xlUnsigned, 1, 0}, 0 );              % set sgn as default '+'
12
13 % Lookup Table
14 if inpAbs < 16
15     prebits=xfix( {xlUnsigned, 7, 0}, 0 );
16     value=xl_slice(inpAbs, 3, 0);
17     outp=xl_concat(sgn, prebits, value);
18     if INPUT_VALUE < 0                       % calc. sgn
19         outp=xl_not(outp)+1;
20     end
21 elseif inpAbs < 32 && inpAbs > 15
22     prebits=xfix( {xlUnsigned, 7, 0}, 1 );
23     value=xl_slice(inpAbs, 3, 0);
24     outp=xl_concat(sgn, prebits, value);
25     if INPUT_VALUE < 0                       % calc. sgn
26         outp=xl_not(outp)+1;
27     end
28 elseif inpAbs < 48 && inpAbs > 31
29     prebits=xfix( {xlUnsigned, 6, 0}, 1 );
30     postbits=xfix( {xlUnsigned, 1, 0}, 1 );
31     value=xl_slice(inpAbs, 3, 0);
32     outp=xl_concat(sgn, prebits, value, postbits);
33     if INPUT_VALUE < 0                       % calc. sgn
34         outp=xl_not(outp)+1;
35     end
36 elseif inpAbs < 64 && inpAbs > 47
37     prebits=xfix( {xlUnsigned, 5, 0}, 1 );
38     postbits=xfix( {xlUnsigned, 2, 0}, 2 );
39     value=xl_slice(inpAbs, 3, 0);
40     outp=xl_concat(sgn, prebits, value, postbits);
41     if INPUT_VALUE < 0                       % calc. sgn
42         outp=xl_not(outp)+1;
43     end
44 elseif inpAbs < 80 && inpAbs > 63
45     prebits=xfix( {xlUnsigned, 4, 0}, 1 );
46     postbits=xfix( {xlUnsigned, 3, 0}, 4 );
47     value=xl_slice(inpAbs, 3, 0);
48     outp=xl_concat(sgn, prebits, value, postbits);
49     if INPUT_VALUE < 0                       % calc. sgn
50         outp=xl_not(outp)+1;
51     end
52 elseif inpAbs < 96 && inpAbs > 79
53     prebits=xfix( {xlUnsigned, 3, 0}, 1 );
54     postbits=xfix( {xlUnsigned, 4, 0}, 8 );
55     value=xl_slice(inpAbs, 3, 0);
56     outp=xl_concat(sgn, prebits, value, postbits);
57     if INPUT_VALUE < 0                       % calc. sgn
```

```
58          outp=xl_not(outp)+1;
59      end
60  elseif inpAbs < 112 && inpAbs > 95
61      prebits=xfix( {xlUnsigned, 2, 0}, 1 );
62      postbits=xfix( {xlUnsigned, 5, 0}, 16 );
63      value=xl_slice(inpAbs, 3, 0);
64      outp=xl_concat(sgn, prebits, value, postbits);
65      if INPUT_VALUE < 0                  % calc. sgn
66          outp=xl_not(outp)+1;
67      end
68  elseif inpAbs > 111
69      prebits=xfix( {xlUnsigned, 1, 0}, 1 );
70      postbits=xfix( {xlUnsigned, 6, 0}, 32 );
71      value=xl_slice(inpAbs, 3, 0);
72      outp=xl_concat(sgn, prebits, value, postbits);
73      if INPUT_VALUE < 0                  % calc. sgn
74          outp=xl_not(outp)+1;
75      end
76  end
77  OUTPUT_VALUE=outp;              % set output
```

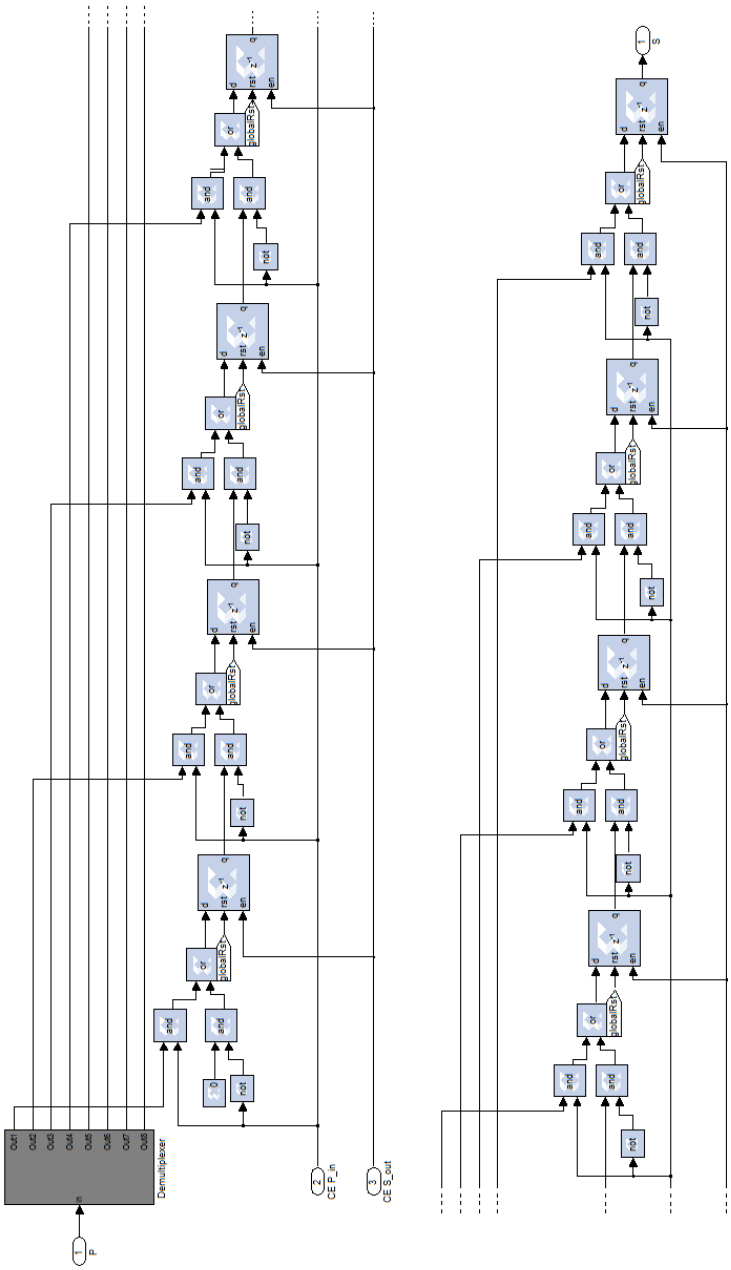
Anhang D

Anti-Aliasing- und Interpolations-Filter im Simulink System Generator



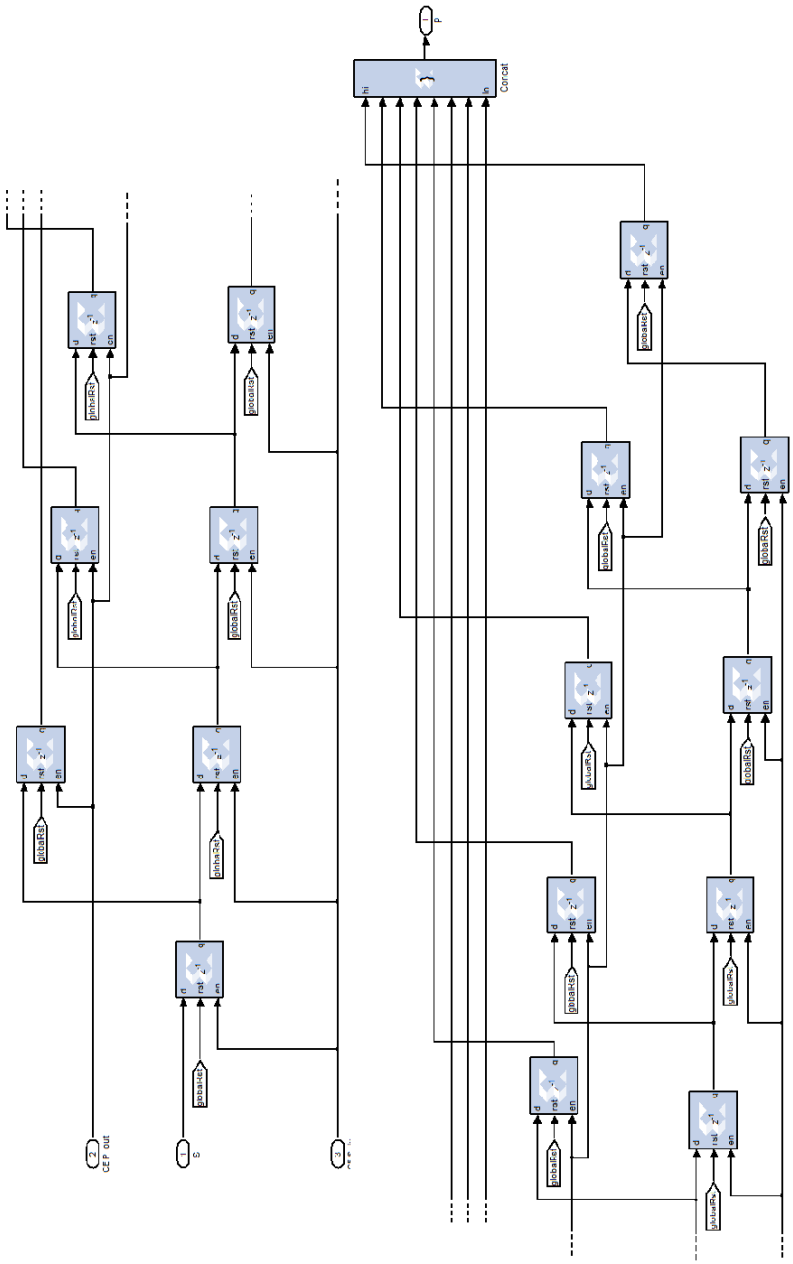
Anhang E

Parallel-/Seriell-Umsetzer im Simulink System Generator



Anhang F

Seriell-/Parallel-Umsetzer im Simulink System Generator



Anhang G

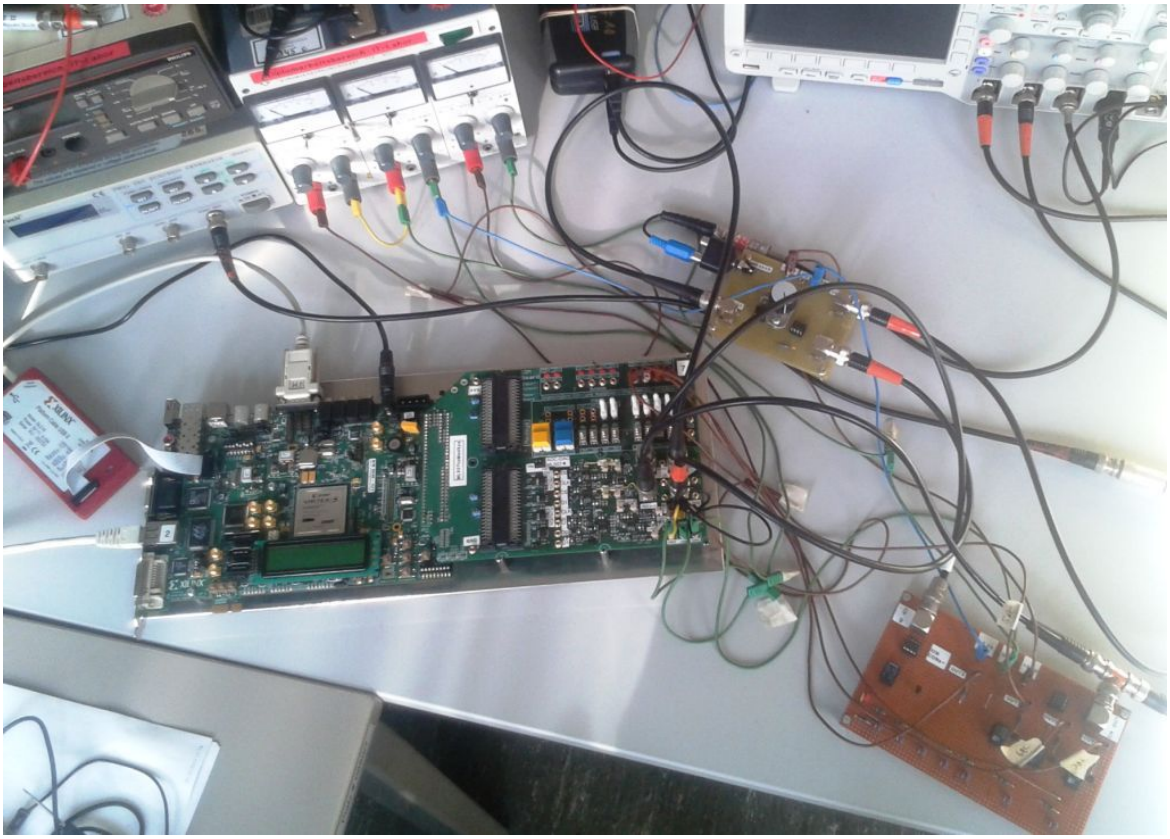
Betriebsanleitung

Um das System nutzen zu können, wird die CD mit den Projektdateien benötigt. Diese ist beim Erstprüfer Prof. Dr.-Ing. Hans Jürgen Micheel hinterlegt. Zudem werden die unten aufgelisteten Hard- und Software-Komponenten benötigt.

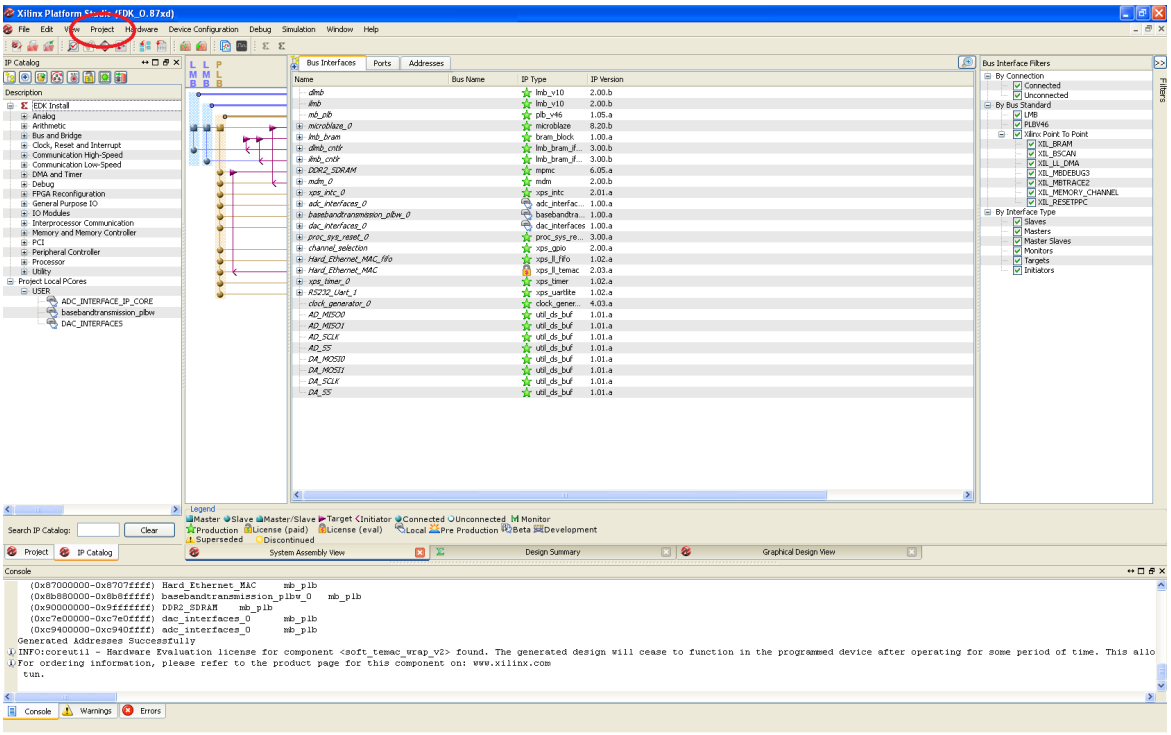
- **Xilinx ML507-Board**
- **Xilinx Platform Cable USB**
- **IOM-Board**
- **ADC-/DAC-Board**
- **mp3-Player oder Funktionsgenerator**
- **Kanalmodell**
- **Filterplatine (AAF, Interpolationsfilter)**
- **Stromversorgungsgerät**
- **Ethernet- und RS232-Kabel**
- **Diverse BNC-Kabel**
- **Xilinx EDK (bzw. ISE Design Suite 13.4)**
- **Matlab (R2011b)**
- **Hyper Terminal**

Zunächst sind das IOM- und das ADC-/DAC-Board an das ML507-Board anzuschließen. Damit der FPGA programmiert werden kann, muss das ML507-Board mit dem Platform-Kabel über USB an den Rechner angeschlossen werden. Für die Steuerung und die Signalauswertung wird das Ethernetkabel an den Rechner und dem ML507-Board angebunden. Für das Auslesen von Systemparameter wird das RS232-Kabel benötigt. Das Stromversorgungsgerät muss $+3,3\ V$ und $-3,3\ V$ liefern. Dies dient der Versorgung des ADC-/DAC-Board, dem Kanalmodell und der Filterplatine. Im Weiteren werden die Komponenten miteinander verbunden. Wenn als Quelle ein Funktionsgenerator verwendet wird, kann dieser mit einem BNC-Kabel und der Filterplatine (Line In) verbunden werden. Falls ein mp3-Player genutzt wird, muss ein BNC→Klinke-Adapter eingesetzt werden.

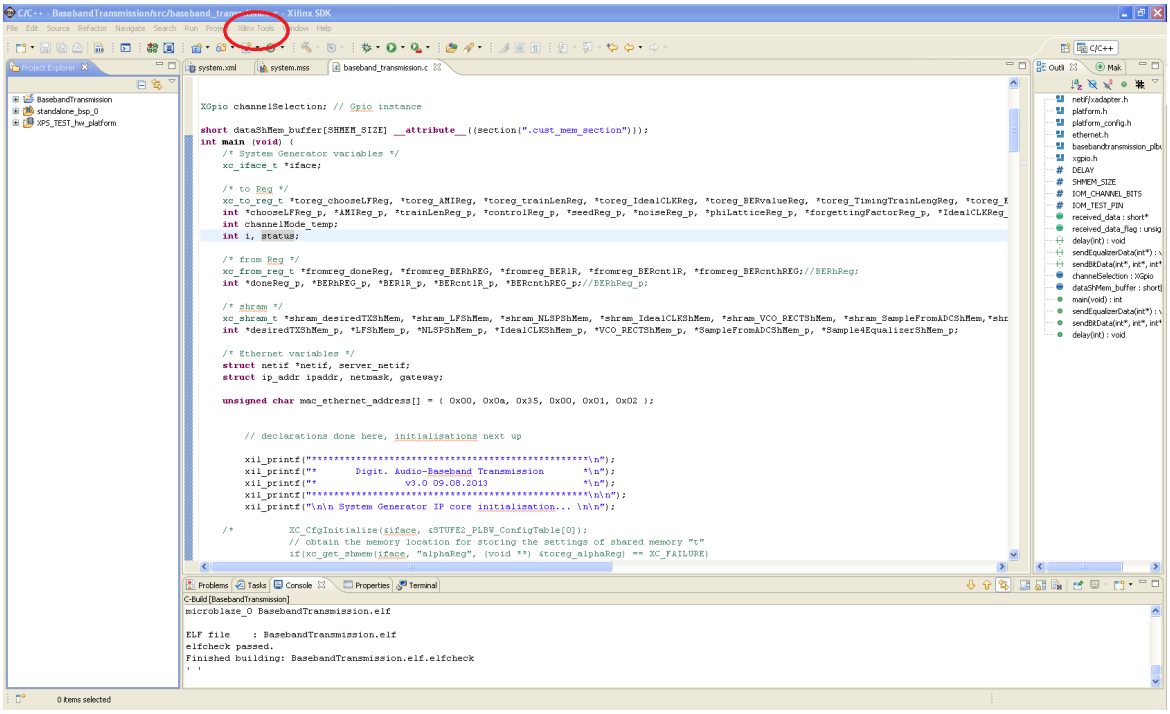
Der Ausgang(ADC1) der Filterplatine muss mit dem ADC1 des ADC-/DAC-Boards verbunden werden. DAC0 ist an den Eingang des Kanalmodells anzuschließen und der Ausgang des Kanalmodells mit dem ADC0. Der DAC1 muss schließlich mit der Filterplatine (DAC1) verbunden werden. Der Ausgang (Speaker) dient der Anbindung an einen Lautsprecher. Unten stehende Abbildung zeigt den Aufbau.



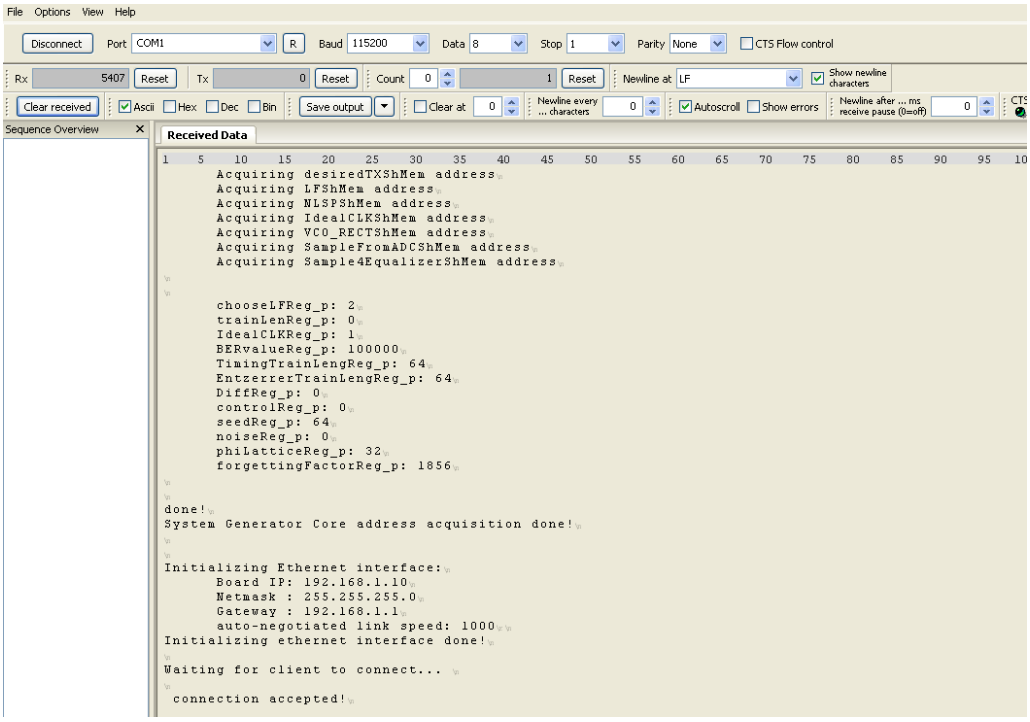
Für die Programmierung muss das Xilinx EDK installiert sein. Das Xilinx Platform Studio (XPS) ist zu starten. Es ist die XPS-File *system.xmp* im Ordner *EDK_Projekt*, aus der Projekt-CD zu öffnen, vgl. nächste Abbildung.



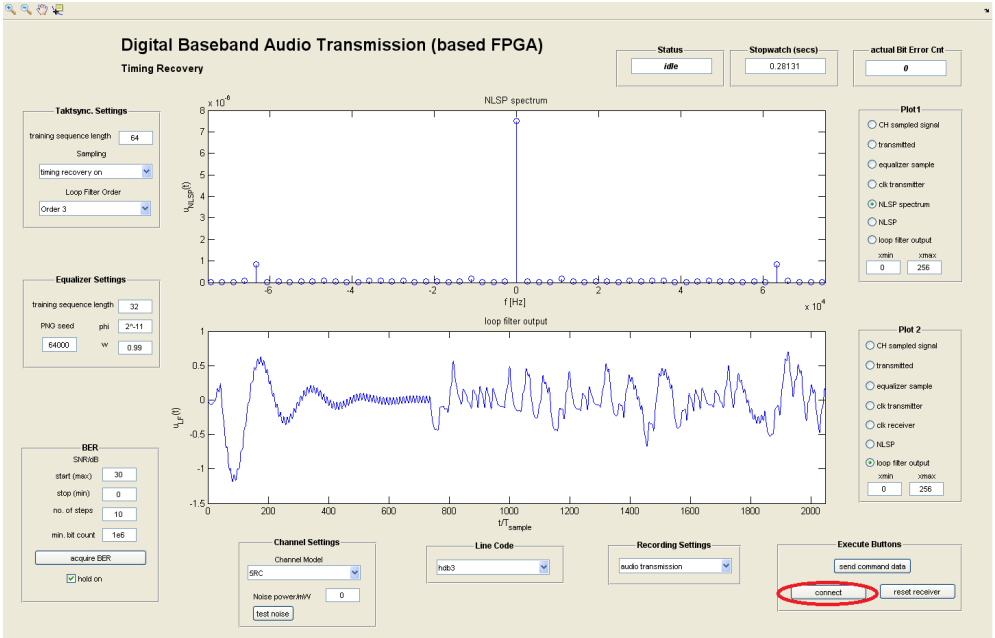
Unter der Registerkarte *Project* muss der Punkt *Export Hardware Design to SDK...* ausgewählt werden. Anschließend öffnet sich das SDK und man wird aufgefordert den Workspace-Ordner auszuwählen. Hier ist der Pfad *EDK_Projekt→SDK→SDK_Export* auf der Projekt-CD auszuwählen. Nächste Abbildung zeigt das geöffnete SDK-Projekt mit der Firmware in C.



Nächster Schritt ist es ein Hyper Terminal Programm zu öffnen. Hier für kann das Open-Source-Tool HTerm genutzt werden. Das ML507-Board ist mit dem COM-Port des PCs zu verbinden. Nun wird im SDK unter *Xilinx Tools* und *Programm FPGA* die Firmware auf den FPGA überspielt. Wenn das Programm erfolgreich übertragen wurde, listet HTerm die Systemparameter auf, vgl. nächste Abbildung.



Nun kann die m-File *AudiotransmissionGUI.m* aus dem Ordner *GUI* der Projekt-CD ausgeführt werden. Die GUI wird gestartet. Zuerst muss das ML507-Board über den Button *connect* verbunden werden. Bei einer erfolgreichen Verbindung ist HTerm der Text *connection accepted!* zu entnehmen. Nun kann das System getestet werden.



Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 30. September 2013	
Ort, Datum	Unterschrift