OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

ELEKTRO- UND
INFORMATIONSTECHNIK

PROJEKTARBEIT

Hannes Bachl

FPGA basierter BPSK/QPSK Empfänger

22.02.2024

Fakultät: Elektro- und Informationstechnik
Studiengang: Bachelor Elektro- und Informationstechnik
Betreuung: Prof. Dr.-Ing. Florian Aschauer

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Projektarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Projektarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum und Unterschrift

Vorgelegt durch:	Hannes Bachl
Matrikelnummer:	3321274
Studiengang:	Bachelor Elektro- und Informationstechnik
Betreuung:	Prof. Dr.-Ing. Florian Aschauer

Vorwort

Inhalt dieser Projektarbeit ist die Erstellung eines einfachen Software defined radio zum Empfang eines QPSK oder BPSK modulierten Signales. Als Basis kommt hier ein Field-programmable gate array mit angeschlossenem externen Analog-to-digital converter zum Einsatz.

Hauptaugenmerk soll dabei auf der eigenen Umsetzung der für ein SDR notwendigen Signalverarbeitungsalgorithmen sowie den Kommunikationsstrukturen, welche benötigt werden um einer Central Processing Unit die Steuerung der umgesetzten Schaltung zu ermöglichen, liegen.

Das Projekt soll als grundlegende Basis für, ein eventuell später noch umzusetzendes, komplexeres SDR dienen und ist deswegen, wo möglich, flexibel gestaltet.

Inhaltsverzeichnis

1 Einleitung

Inhalt dieses Projektes ist die Erstellung eines SDR, wobei ein FPGA als Plattform dienen soll. Der FPGA-Logikteil soll weiterhin von einem Rechenkern, für die weitere Datenverarbeitung, unterstützt werden.

Da es sich um ein Lernprojekt handelt soll der Fokus, bei der Erstellung der FPGA-Schaltung, auf der selbständigen Erstellung aller Signal verarbeitenden Blöcke liegen. Komplexe Blöcke welche für die Kommunikation mit dem Rechensystem benötigt wurden *(z.B. AXI-Infrastruktur)* wurden aus Zeitgründen nicht selbst erstellt, sondern das vorhandene IP des FPGA-Herstellers verwendet.

1.1 Anforderungen

Zu beginn des Projektes wurde mit der Festlegung der Anforderungen an des Gesamtsystem begonnen, um so die notwendigen Hardware- und Logikkomponenten richtig auslegen zu können.

Dabei wurden die folgende Anforderungen an das System definiert:

1. **Unterstützung der Verwendung eines extern ADC**
   - 1.1. Abtastfrequenz von mindestens $100\,\text{MHz}$ [$f_{ADC} > 100\,\text{MHz}$].
   - 1.2. Datenbreite soll mindestens $12\,\text{bit}$ betragen [$w_{adc} > 12$]
   - 1.3. Externer anti-Alias Filter notwendig und Abtastung in der ersten Nyquist-Zone.
2. **Digitale Abwärtsmischung in das Basisband**
   - 2.1. Eingangssignal im Frequenzbereich $1\,\text{MHz}$ bis $20\,\text{MHz}$. [$f_s \in [1\,\text{MHz}; 20\,\text{MHz}]$]
   - 2.2. Interner Numerically controlled oscillator wobei die Frequenz im Frequenzbereich variable verstellbar sein soll
   - 2.3. Interner komplexer Mischer mit $16\,\text{bit}$-Breitem I/Q Signal am Ausgang
   - 2.4. Dezimierungsfilter mit, zum Synthesezeitpunkt einstellbaren, variablen Dezimierungsverhältnis
3. **Carrier-Tracking für BPSK/QPSK Signale**
   - 3.1. Interner Phasen-Komparator mit folgendem Regler zum konditionieren des lokalen Oszillators.
   - 3.2. Modulation (BPSK/QPSK) soll zur Laufzeit wechselbar sein.

3.3. Als Regler soll ein PID-Regler mit, zur Laufzeit einstellbaren Koeffizienten, sein.

3.4. Das Carrier-Tracking soll Abschaltbar sein und nur aktiv sein wenn das Eingangssignal eine gewisse Amplitude aufweist.

4. **Kommunikation mit dem Rechenkern**

4.1. Ein Rechenkern soll die FPGA-Schaltung verwalten und die weitere Verarbeitung der Nutzdaten übernehmen.

4.2. Die Steuerung der einzelnen Module soll der Rechenkern über eine AXI-Lite Registerbank vornehmen.

4.3. Die gemischten und dezimierten Nutzdaten sollen via Direct Memory Access-Controller dem Rechenkern zur Verfügung gestellt werden.

1.2 Auswahl der Hardware-Komponenten

Nach Festlegung der Anforderungen wurde die für die Umsetzung notwendige Hardware ausgewählt.

Aufgrund der Anforderung 4.1 sowie dem Fokus der Vorlesung, FPGA-Logik in CPU-Rechensysteme zu integrieren, ist es notwendig einen FPGA mit eingeschlossenen CPU-Kernen auszuwählen.

Die Wahl fiel hier aus Kostengrund und da Know-How sowie die notwendige Tool-Umgebung bereits vorhanden war auf die Zynq-7000 APSoC Serie des Herstellers Xilinx. Es handelt sich dabei um eine FPGA-Fabric der Xilinx 7-Series *(PL,Programmable-Logic)* mit angeschlossenem Dual-Core ARM Cortex-A9 *(PS,Processing-System)*.

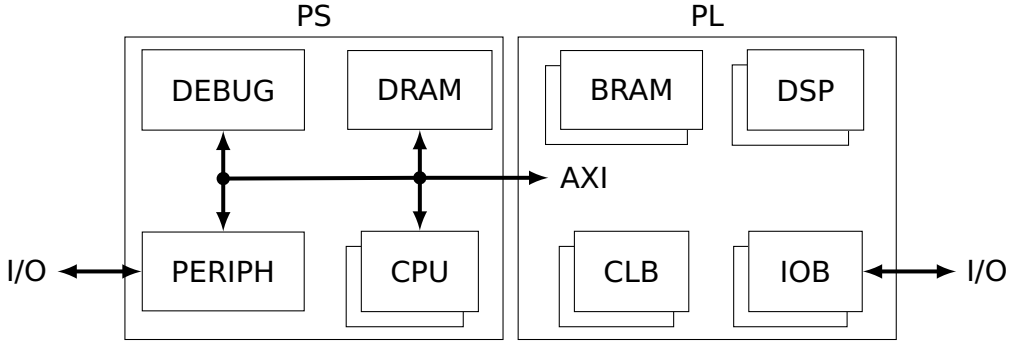

Abbildung 1.1: Struktur des verwendeten Zynq-7000 FPGA

Die in der FPGA-Fabric (PL-Teil) vorhandenen Logic-Resourcen *(BRAM,CLB,IOB, DSP-Slices)* werden gemäß des noch zu erstellenden FPGA-Designs verdrahtet und am Ende mit dem AXI-Bus des Rechensystemes (PS-Teil) verbunden.

Da der FPGA mit ADC Modul selbst beschafft und bezahlt wird war hier vor allem der Preis ausschlaggebend. Die Wahl fiel hier auf ein Eclypse Z7 Board der Firma Digilent.

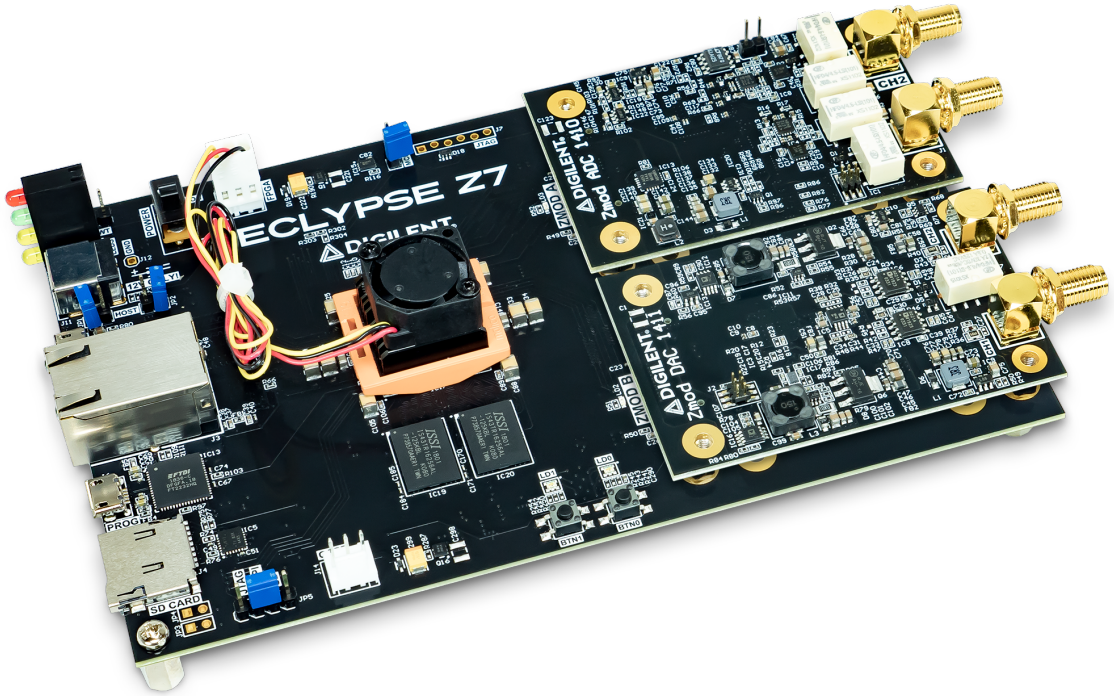

Abbildung 1.2: Bild des Eclypse-Z7 mit ADC und DAC
©Digilent Inc. [2]

Hauptgrund für die Auswahl dieses Boards ist der, im Vergleich zu anderen FPGA Mezzanine Card (FMC) basierten Boards, relativ günstige Preis, sowie das Vorhandensein von vielen IP-Cores der Firma Digilent welche die Ansteuerung der einzelnen Hardware-Komponenten vereinfachen und dem großen Umfang von verfügbaren Referenzmaterialien.

Der von dem Board verwendete ADC *(ZModScope 1410-105, AD9648BCPZ-105)* erfüllt alle gestellten Anforderungen.[3]

Bei dem verwendeten FPGA handelt es sich um einen XC7Z020-1CLG484C mit Dual-Core Cortex-A9 Prozessor mit $666\,\mathrm{MHz}$, sowie $1\,\mathrm{GiB}$ externem DDR3-DRAM[2]

Ein DAC wäre ebenfalls vorhanden, wird jedoch nicht genutzt.

2 Entwurf und Design

2.1 FPGA-Design

Nach der Festlegung der Anforderungen wurde mit der Konzeption und Entwurfsphase des gesamt Systemes sowie des FPGA-Designs begonnen.

Hierbei wurde zuerst anhand der Anforderungen die Architektur des zu erstellenden FPGA-Designs festgelegt.

Die Kommunikation zwischen den einzelnen Modulen, besonders für den Datenaustausch, soll zum größten Teil mit einer AXI-Stream basierten Schnittstelle umgesetzt werden. Für die, hauptsächlich zur Steuerung verwendete, Kommunikation zwischen FPGA-Design und Rechenkern sollen AXI-Lite basierte Registerbänke verwendet werden.

In dem folgenden Entwurf nicht betrachtet werden die weiterhin notwendigen AXI-Infrastruktur Blöcke welche, für die Kommunikation mit dem Rechenkern weiterhin noch benötigt werden, jedoch nicht selbst umgesetzt werden. Eine Gesamtübersicht des Designs (mit AXI-Infrastruktur) kann im Kapitel 3.1 gefunden werden.

Es handelt sich um eine klassische QPSK-Empfängerarchitektur mit direkter digitaler Abwärtsmischung.

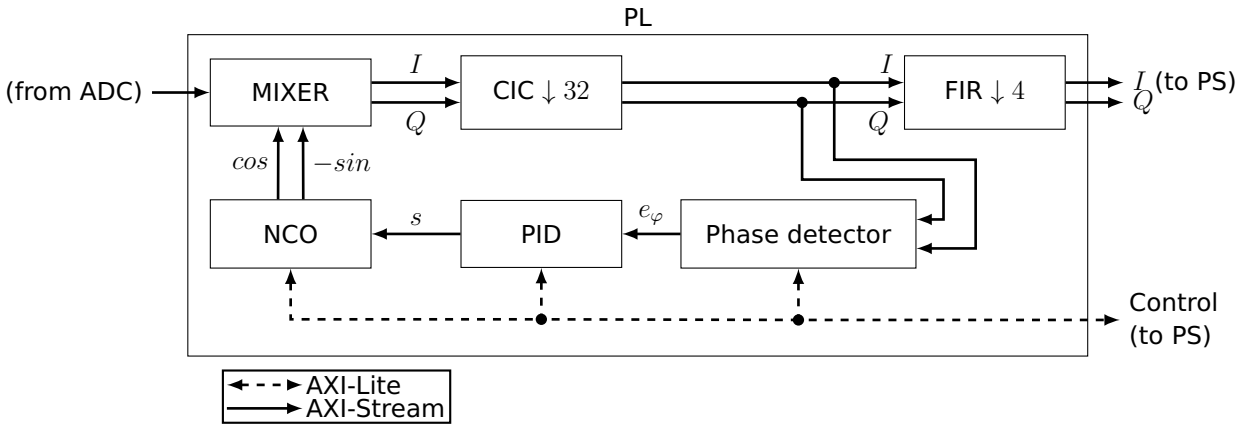

Abbildung 2.1: Architektur des FPGA-Designs (ohne AXI-Infrastruktur)

Die Hauptkomponenten des FPGA-Designs werden in den folgenden Abschnitten noch etwas näher erklärt.

2.1.1 Mischer (Mixer)

Der Mischer hat die Aufgabe den von dem ADC abgetasteten Datenstrom mit den, vom lokalen Oszillator erzeugten, Trägern zu multiplizieren und so das Nutzsignal in das Basisband zu verschieben. Die besondere Schwierigkeit liegt hier dabei, dass der Mischer mit der Taktrate des ADCs f_{ADC} betrieben werden muss.

Für den Empfang von QPSK-modulierten Daten ist es notwendig einen komplexen Mischer *(sogenannter I/Q-Mischer)* zu verwenden. Als Eingang dienen zwei vom lokalen Oszillator (NCO) erzeugten Referenzsignale *(Sinus/Kosinus)*, sowie der ADC-Datenstrom ($x[n]$), welche dann wie folgt Multipliziert werden:

$$I[n] = x[n] \cdot cos(2\pi \frac{f}{f_{ADC}} \cdot n) \tag{2.1}$$

$$Q[n] = x[n] \cdot -sin(2\pi \frac{f}{f_{ADC}} \cdot n) \tag{2.2}$$

Es entsteht ein komplexer Datenstrom welcher dann für die weitere Verarbeitung genutzt werden kann:

$$\underline{y}[n] = I[n] - j \cdot Q[n] = x[n] \cdot e^{j2\pi \frac{f}{f_{ADC}} \cdot n} \tag{2.3}$$

Zusätzlich zu dem Signal im Basisband entsteht bei der Mischung immer auch eine Kopie des Signals bei dem doppelten der Trägerfrequenz.[4] Die nicht erwünschte Kopie wird anschließend durch die Verwendung eines Tiefpassfilters eliminiert. Diese Aufgabe übernimmt hier der CIC-Dezimierungsfilter.

Ein- und Ausgabeschnittstellen des Mischers sollen als AXI-Stream realisiert werden, wobei es notwendig ist jeden Taktzyklus einen neuen Datenwert im Mischer zu verarbeiten.

2.1.2 Lokaler Oszillator (NCO)

Der lokale Oszillator hat die Aufgabe die von dem Mischer verwendeten Referenzsignale zu erzeugen.

Umgesetzt ist der lokale Oszillator als Numerisch gesteuerter Oszillator (NCO). Die notwendige Sinus-/Kosinusignale werden hierbei durch die sogenannte direkte digitale Synthese (DDS) erzeugt.

Dabei handelt es sich effektiv um einen Zähler (Phasen-Akkumulator, n-Bit breit) von welchem anschließend die untersten m-Bit verwendet werden um die Ausgangsdatenwerte in einer Lookup-Tabelle nachzuschlagen. Anhand der Schrittweite s um welche der Phasen-Akkumulator jeden Taktzyklus erhöht wird lässt sich die Grundfrequenz der ausgegebenen Sinusschwingungen einstellen.

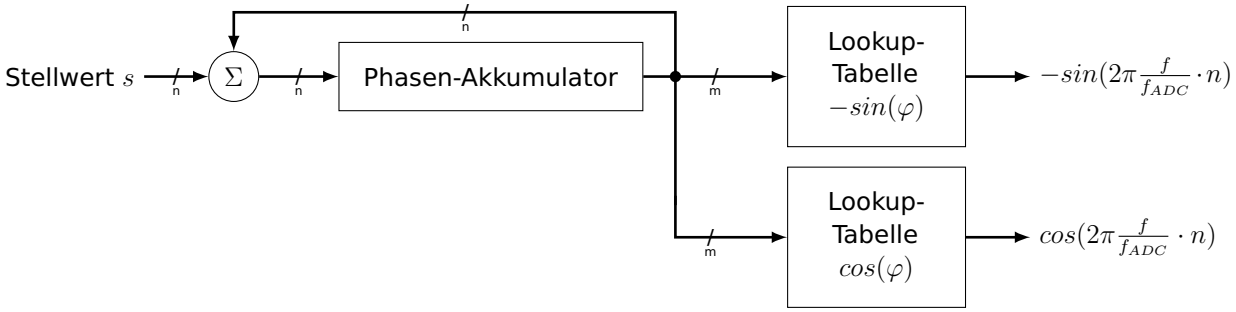

Abbildung 2.2: Schematischer Aufbau des lokalen Oszillators.

Die Breite des Phasen-Akkumulator, die Ausgangswertbreite sowie die Anzahl der Stützpunkte der Lookup-Tabelle haben einen großen Einfluss auf die Güte der erzeugten Schwingung. [1]

Der Zusammenhang zwischen Frequenzstellwert s und der Ausgangsfrequenz der erzeugten Sinusschwingung $\frac{f}{f_{ADC}}$ ist abhängig von der Akkumulator Bitbreite n:

$$\frac{f}{f_{ADC}} = \frac{s}{2^n} \tag{2.4}$$

2.1.3 CIC-Dezimierungsfilter

2.1.4 FIR-Kompensationsfilter

Der FIR-Kompensationsfilter hat die Aufgabe der weiteren Abtastratenreduzierung mit der notwendigen Bandbreiten Begrenzung. Er wirkt zusätzlich als Kompensationsfilter für den vorgeschalteten CIC-Filter um dessen Schwankungen der Amplitudenkennlinie in Passband zu kompensieren.

Die Filterkoeffizienten wurden mit Matlab anhand des CIC-Filter Frequenzganges und der gewünschten Gesamtfrequenzcharakteristik ermittelt.

TODO

Aus Zeit- und Effizienzgründen soll hier der FIR-Filter Compiler[6] von Xilinx verwendet werden.

2.1.5 Phasen-Komparator

Der Phasen-Komperator ermittelt aus den komplexen Basisbandsignalen (I/Q) einen Phasenfehler anhand dessen die Frequenz des lokalen Oszillators später so angepasst werden soll dass er möglichst exakt dem Oszillator des Senders folgt.

Diese Ermittlung des Phasenfehlers erfolgt anhand der in einer Costas-Loop verwendeten Technik zur Ermittlung des Phasenfehlers.[4] Abhängig von der gewählten

Modulationsart (BPSK/QPSK) werden die folgenden Rechenschritte für die Ermittlung des Phasenfehlers genutzt:

Für **BPSK**:

$$e_\varphi = sign(I) \cdot Q \tag{2.5}$$

Für **QPSK**:

$$e_\varphi = sign(I) * Q - sign(Q) * I \tag{2.6}$$

Auf die genaue Herleitung dieser Zusammenhänge soll hier verzichtet werden. Eine genauere analytische Betrachtung [5] könnte später für die Auslegung der Parameter des PID-Reglers anhand des Streckenmodelles verwendet werden.

2.1.6 PID-Regler

Der PID-Regler wandelt die vom Phasen-Komperator ermittelte Phasenabweichung der Empfangssymbole und den von außen vorgegebenen Soll-Frequenzpunkt in ein Frequenzstellsignal für den lokalen Oszillator um. Er versucht so sie Frequenz- und Phasendifferenz zwischen dem lokalen Oszillator und dem Oszillator des Senders auszuregeln.

Der (im FPGA) zeitdiskret Implementierte Regler setzt eine Übertragungsfunktion zweiter Ordnung um: $F(z) = \frac{A \cdot z^2 + B \cdot z + C}{z^2 - 1}$, deren Parameter A, B, C von außen eingestellt werden sollen. Diese flexible Umsetzung des Reglers soll es ermöglichen den Regler später für unterschiedliche Anwendungsfälle einfach zur Laufzeit anpassen zu können.

Anhand der Bilinear-Transformation (Tustin-Methode) können die zeitdiskreten Reglerparameter aus den Parametern eines zeitkontinuierlich PID-Reglers (K, T_n, T_v) und der Abstastfrequenz $T_A = \frac{32}{f_{ADC}}$ abgeleitet werden:

$$A = K \cdot (1 + \frac{T_A}{2 \cdot T_n} + \frac{2 \cdot T_v}{T_A}) \tag{2.7}$$

$$B = K \cdot (\frac{T_A}{T_n} + \frac{4 \cdot T_v}{T_A}) \tag{2.8}$$

$$C = K \cdot (\frac{T_A}{2 \cdot T_n} + \frac{2 \cdot T_v}{T_A} - 1) \tag{2.9}$$

Die Implementierung erfolgt später anhand der Differenzengleichung welche aus der Übertragungsfunktion abgeleitet werden kann:

$$y[n] = A \cdot x[n] + B \cdot x[n-1] + C \cdot x[n-2] + y[n-2] \tag{2.10}$$

3 Implementierung

3.1 FPGA-Design

3.2 Software

Literaturverzeichnis

[1] L. Cordesses. Direct digital synthesis: a tool for periodic wave generation (part 1). *IEEE Signal Processing Magazine*, 21(4):50–54, 2004.

[2] Digilent Inc. Eclypse-z7 reference manual. https://digilent.com/reference/programmable-logic/eclypse-z7/reference-manual. [Online; Abgerufen am: 22.02.2024].

[3] Digilent Inc. Zmod scope reference manual. https://digilent.com/reference/zmod/scope/reference-manual?s[]=zmodscope. [Online; Abgerufen am: 22.02.2024].

[4] Qasim Chaudhari. Costas loop for carrier phase synchronization. https://wirelesspi.com/costas-loop-for-carrier-phase-synchronization/. [Online; Abgerufen am: 27.02.2024].

[5] Xiao-ou Song. Analysis and implementation of modified costas loop for qpsk. In *2015 International Conference on Intelligent Networking and Collaborative Systems*, pages 394–397, 2015.

[6] Xilinx Inc. *LogiCORE IP FIR Compiler v5.0 Data Sheet (DS534)*, 03 2011.

Abkürzungsverzeichnis

FPGA Field-programmable gate array
SDR Software defined radio
ADC Analog-to-digital converter
BPSK Binary Phase-Shift Keying
QPSK Quadratur Phase-Shift Keying
CPU Central Processing Unit
NCO Numerically Controlled Oscillator
AXI Advanced eXtensible Interface
DMA Direct Memory Access
APSoC All programmable System on a Chip
BRAM Block random access memory
CLB Configurable logic block
DSP Digital Signal Processing
IOB Input/Output-Buffer
DAC Digital-to-analog converter
CIC Cascaded-Integrator-Comb
FIR Finite impulse response
PID Propertional-Integral-Differential
NCO Numerically controlled oscillator
DDS Direct digital synthesis

Abbildungsverzeichnis

Tabellenverzeichnis