Sascha Müller zum Hagen

Evaluierung eines Interfaces für industrielle Hochgeschwindigkeitskameras

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Informations- und Elektrotechnik
Studienrichtung Informationstechnik
am Studiendepartment Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Jochen Schneider
Zweitgutachter: Prof. Dr. Franz Schubert

Abgegeben am 24. August 2007

Sascha Müller zum Hagen

Thema der Diplomarbeit

Evaluierung eines Interfaces für industrielle Hochgeschwindigkeitskameras

Stichworte

Fast CameraLink, Phy, TLK3114, Bitfehlerrate

Kurzzusammenfassung

Die hier vorliegende Diplomarbeit beschäftigt sich mit einem neuen Standard für die Datenübertragung von der Kamera zum PC. Dieser ist erfolgreich getestet und untersucht worden. Bei diesen Tests ist der Phy „TLK3114SC“ der Firma Texas Instruments verwendet worden. Die schnellere Datenübertragung führt zu einer höheren Übertragungsfrequenz (ca. 1,6 GHz). Die damit einhergehenden Nichtlinearitäten der Bauelemente haben berücksichtigt werden können. Wellenwiderstand, Signalausbreitungsgeschwindigkeit und die maximal mögliche Leitungslängendifferenz, die nicht zu einer zu großen Laufzeitdifferenz zwischen den Signalen führt, sind berechnet worden. Die mit der schnellen Datenübertragung einhergehende größere Fehleranfälligkeit ist durch die Einführung einer Kodierung des Signals äußerst klein gehalten worden.

Sascha Müller zum Hagen

Title of the paper

Evaluation of an interface for industrial high-speed cameras

Keywords

Fast CamerLink, Phy, TLK3114, Bit Error Rate

Abstract

In the following thesis (diploma) a new standard for the data communication from the camera to the PC is considered. This standard is tested and examined successfully. During these tests the Phy „TLK3114SC“ of Texas Instruments is used. The rather high data communication induces a rather high transmission frequency (approx. 1.6 GHz). The induced non-linearities of the elements have been taken into account on. The characteristic impedance, signal propagation speed and the maximal possible difference of the length of the wires are calculated. This difference is such, that the trip-time-difference of the signals does not become too large. The rather fast communication of signals induces a rather great of rate of signal-error. This rate is kept small by introducing a coding of the signals.

Inhaltsverzeichnis

Einleitung

Weltweit werden heutzutage in diversen industriellen Anwendungen digitale Kameras mit nachgeschalteter elektronischer Bildverarbeitung eingesetzt, um Steuerungs- und Regelungsaufgaben zu optimieren oder diese erst zu ermöglichen. Die Bildverarbeitung wird dabei üblicherweise von einem Industrie-PC abgewickelt, der über eine kabelgebundene Datenübertragungsstrecke mit der Kamera verbunden ist, auf der die Bild- und Steuerungsdaten der Kamera übertragen werden.
Die zunehmende Komplexität der Prozesse, deren stetig steigende Ablaufgeschwindigkeit sowie neue hochauflösende Kameras bedingen dabei eine immer höhere Bandbreite der Datenübertragungsschnittstelle zwischen Kamera und PC.
Die Bandbreite der derzeit etablierten Schnittstelle nach „CameraLink“-Standard erschöpft sich bei maximal 5,44 $^{\text{GBit}}/_{\text{sec}}$ und stellt damit schon heute den Engpass der gesamten Datenverarbeitungskette dar. CameraLink lässt sich damit für zukünftige Hochgeschwindigkeitsanwendungen nicht länger sinnvoll verwenden.

Ziel dieser Diplomarbeit ist es, einen neuen Standard für die Datenübertragung von der Kamera zum PC zu testen und zu untersuchen. Dieser neue Standard wird im Folgenden immer als „Fast CameraLink“[1] bezeichnet und verwendet den Phy „TLK3114SC“ der Firma Texas Instruments.
Insbesondere gilt es dabei die Schwierigkeiten, die beim Übergang von CameraLink auf Fast CameraLink entstehen können, genauer zu untersuchen. Diese werden im Folgenden in I und II in Kürze skizziert:

I Zur Erzielung einer Datenübertragungsrate von 3,125 $^{\text{GBit}}/_{\text{sec}}$ wächst beim Übergang von CameraLink auf Fast CameraLink die Übertragungsfrequenz auf der differentiellen Leitung von 85 MHz auf ca. 1,6 GHz. Bei solch hohen Frequenzen entstehen unter anderem zwei Probleme:
Erstens muss überprüft werden, ob bei den Bauteilen (inkl. der Leitung) nichtlineare Eigenschaften auftreten und diese nennenswerte Auswirkungen haben. Dazu werden die Bauteile mit Ersatzschaltbildern modelliert, bei denen jedes einzelne Element des Ersatzschaltbildes als ideal angenommen werden kann (siehe Kapitel 1).
Zweitens ist beim Platinenaufbau folgendes zu beachten: Alle Leitungen besitzen einen spezifischen Wellenwiderstand. Außerdem muss (nach Berechnung der Signalausbreitungsgeschwindigkeit) die Länge der Leitungen bestimmt werden, um

[1] Es gibt zwar schon einen Namen, aber dieser wird erst Anfang des vierten Quartals diesen Jahres veröffentlicht.

starke Laufzeitunterschiede zwischen gewissen Signalen zu verhindern (siehe Kapitel 2).

II Auf Grund des erheblichen Anstiegs der Datenrate beim Übergang von CameraLink auf Fast CameraLink wird das übertragene Signal leichter anfällig für Übertragungsfehler. Zur Verringerung dieser Fehleranfälligkeit findet eine Kodierung des Signals statt.
Zusätzlich wird die Taktinformation in das kodierte Signal eingebettet und somit mitübertragen; d.h., dass für die Taktübertragung keine extra Leitung zur Synchronisation erforderlich ist, wodurch die Fehleranfälligkeit noch weiter abfällt.

Der Hauptteil dieser Diplomarbeit umfasst fünf Kapitel. Kapitel 1 erläutert die Grundlagen, die für das Verständnis dieser Arbeit erforderlich sind. In Kapitel 2 werden die Planung der Tests und der damit verbundene Platinenaufbau beschrieben. Kapitel 3 beinhaltet die Realisierung der erstellten Platine sowie die Berechnung für die verwendeten Bauteile (Schaltregler, etc.). Im vierten Kapitel werden die für das Programmieren des FPGAs notwendigen Komponenten des Programms vorgestellt. Kapitel 5 stellt den genauen Testaufbau und die Ergebnisse der Tests dar.

1 Grundlagen

1.1 Bilddatenübertragung

Zur Datenübertragung von der Kamera an den PC haben sich bei der industriellen Bilddatenübertragung die folgenden Standards im Laufe der Zeit etabliert:

Standard	Bruttoübertragungsgeschwindigkeit
USB 2.0	480 $^{\mathrm{MBit}}/_{\mathrm{sec}}$ [12]
IEEE 1394 a	800 $^{\mathrm{MBit}}/_{\mathrm{sec}}$ [11]
IEEE 1394 b	1600 $^{\mathrm{MBit}}/_{\mathrm{sec}}$ [11]
GigE	1000 $^{\mathrm{MBit}}/_{\mathrm{sec}}$ [14]
CameraLink	5,44 $^{\mathrm{GBit}}/_{\mathrm{sec}}$ [10]

Tabelle 1.1: Etablierte industrielle Bilddatenübertragungsstandards und deren Datenübertragungsgeschwindigkeiten

Einzig und allein beim CameraLink-Standard ist die Brutto- gleich der Nettodatenübertragungsgeschwindigkeit. Bei allen anderen Standards werden die Bilddaten in Packeten versendet.
Im Vergleich dazu erreicht Fast CameraLink eine Datenübertragungsrate von 3,125 $^{\mathrm{GBits}}/_{\mathrm{sec}}$ pro differentiellem Pärchen[1]. Zur Steigerung der Datenübertragungsrate sind in dem hier verwendeten Standard folgende Kombinationen von Lanes vorgesehen:

[1] Ein differentielles Pärchen entspricht einer Lane.

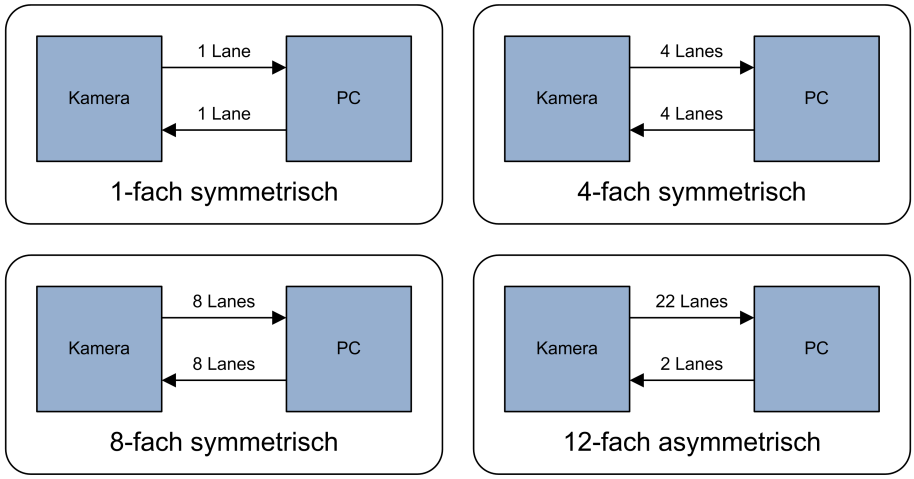

Bild 1.1: Vorgesehene Kombinationen von differentiellen Pärchen

Insofern können von der Kamera beim 12-fachen asymmetrischen Versenden Daten bis zu 68,75 $^{\mathrm{GBit}}/_{\mathrm{sec}}$ verschickt und maximal 6,25 $^{\mathrm{GBit}}/_{\mathrm{sec}}$ empfangen werden.

1.2 Leitungseigenschaften

Auf Grund der Tatsache, dass die Datenübertragung über eine Kupferleitung erfolgt, soll hier kurz auf das Ersatzschaltbild einer solchen Leitung eingegangen werden. Dieses Ersatzschaltbild ist für das Verständnis der bei den Tests verwendeten Störglieder erforderlich.
In Bild 1.2 wird ein Ersatzschaltbild einer einfaches Leitung [19, Seite 1194] für das Übertragen eines Signals dargestellt. Anhand dieses Ersatzschaltbildes ist ersichtlich, dass jede Leitung eine Tiefpasscharakteristik bei hohen Frequenzen aufweist.
Bei der asymmetrischen Signalübertragung werden eine Signal- und eine Masseleitung

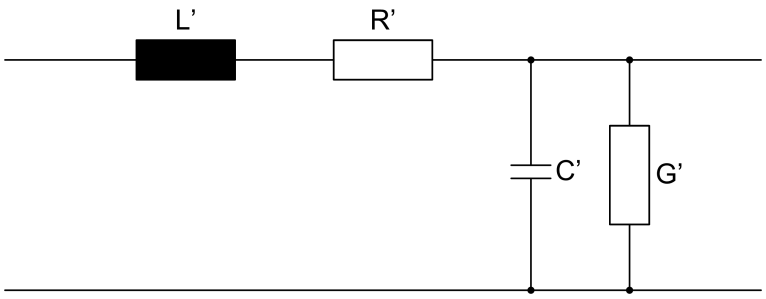

Bild 1.2: Ersatzschaltbild einer einfachen (asymmetrischen) Leitung

im Kabel mitgeführt. Die Masseleitung ist kein Signalträger und wird daher als ideal

angenommen. Da aber bei der differentiellen Datenübertragung keine Masseleitung mitgeführt wird, sondern zwei Signalleitungen für das Versenden der Daten zuständig sind, muss das Ersatzschaltbild angepasst werden. Dies impliziert, dass der Widerstandsbelag R' und der Induktivitätsbelag L' der Leitung in beiden Signalleitungen auftritt. Somit ergibt sich ein neues Ersatzschaltbild, wie es in Bild 1.3 abgebildet ist.

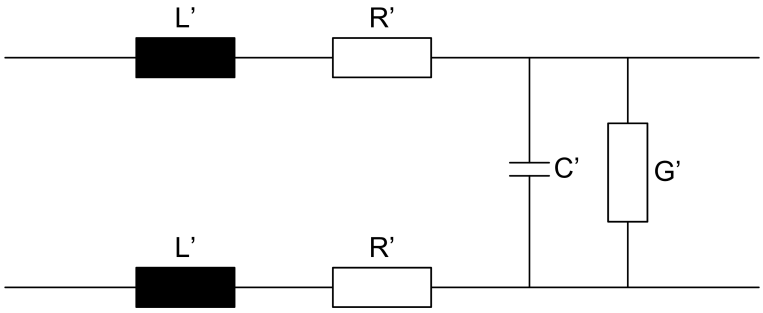

Bild 1.3: Ersatzschaltbild einer differentiellen Leitung

1.3 Eigenschaften von Bauelementen

Bei hohen Frequenzen verhalten sich reale passive Bauteile nicht mehr ideal. In Bild 1.4 wird das Ersatzschaltbild eines realen/ komplexen Widerstandes und eines realen/ komplexen Kondensators dargestellt.
Der komplexe Widerstand verfügt auf Grund seines internen Aufbaus über eine parasi-

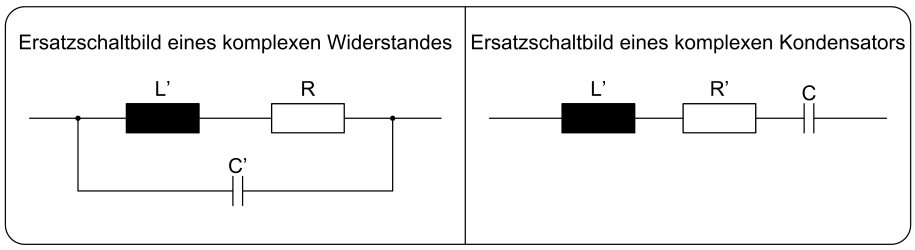

Bild 1.4: Ersatzschaltbild eines komplexen Widerstandes und eines komplexen Kondensators

täre Kapazität C' und eine parasitäre Induktivität L'. In [19, Seite 1326] wird die Formel für diesen komplexen Widerstand wie folgt angegeben:

$$Z_R(j\omega) = \frac{R + j\left(\omega \cdot \left(L' - C' \cdot R^2\right) - \omega^3 \cdot L'^2 \cdot C'\right)}{\left(1 - \omega^2 \cdot L' \cdot C'\right)^2 + \omega^2 \cdot C'^2 \cdot R^2} \tag{1.1}$$

Je größer die Frequenz ist, desto größer wird der imaginäre Anteil des komplexen Widerstandes und dieser verhält sich dann entweder kapazitiv oder induktiv. Da die parasitären

Anteile des komplexen Widerstands abhängig von Bauform und Fertigungsprozess sind, ist es erforderlich, im Datenblatt des Herstellers nachzusehen, wie sich die komplexen Widerstände bei höheren Frequenzen verhalten. In Bild 1.5, das dem Datenblatt [22, Seite 6, oberes Bild] entnommen ist, wird der komplexe Widerstand als Funktion der Frequenz dargestellt.
Unter Berücksichtigung des Graphen in Bild 1.5 lassen sich die komplexen Widerstände

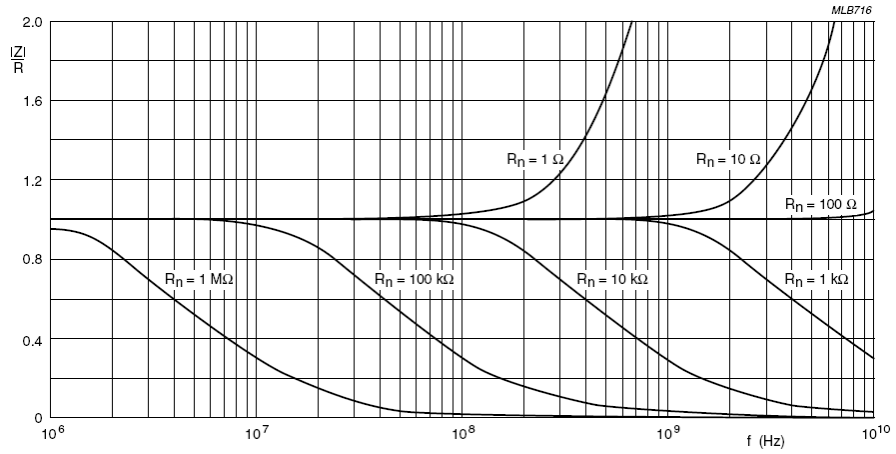

Bild 1.5: Verhalten der komplexen Widerstände in Abhängigkeit von der Frequenz

auch bei höheren Frequenzen für die Filter- oder Dämpfungsglieder verwenden.
Der Kondensator besitzt auf Grund seines internen Aufbaus ebenfalls parasitäre Anteile. Laut [19, Seite 1330] berechnet sich der komplexe Widerstand für den Kondensator bei hohen Frequenzen folgendermaßen:

$$Z_C(j\omega) = \frac{1 + j\omega \cdot C \cdot R' + (j\omega)^2 \cdot L' \cdot C}{j\omega \cdot C} \tag{1.2}$$

Hier gilt wie bei Formel 1.1 von der letzten Seite, dass bei höheren Frequenzen der imaginäre Anteil des Kondensatorwiderstandes zunimmt und dessen kapazitive Wirkung verändert. In diesem Fall verhält sich der Kondensator ab einer bestimmten Frequenz induktiv.
Die Frequenzabhängigkeit des komplexen Widerstands des Kondensators ist durch die obere Kurve in Bild 1.6 auf der nächsten Seite veranschaulicht. Zusätzlich wird im selben Bild durch die untere Kurve gezeigt, inwiefern der parasitäre ohmsche Widerstand R' in schwacher Form von der Frequenz abhängt. Diese Gerade liegt fast parallel zur Abszisse. Bild 1.6 auf der nächsten Seite ist aus den Werten von [17] erstellt worden.
Im Folgenden werden die Kapazitäten betrachtet, die sich in einem Bereich von 0,2 pF bis 1,5 pF befinden. Wenn man die beiden Kurven aus Bild 1.6 verlängert, schneiden sich diese bei einer gewissen Frequenz f_s.
Für $f \leq f_s$ gilt, dass der komplexe Widerstand eines Kondensators kapazitiv ist.

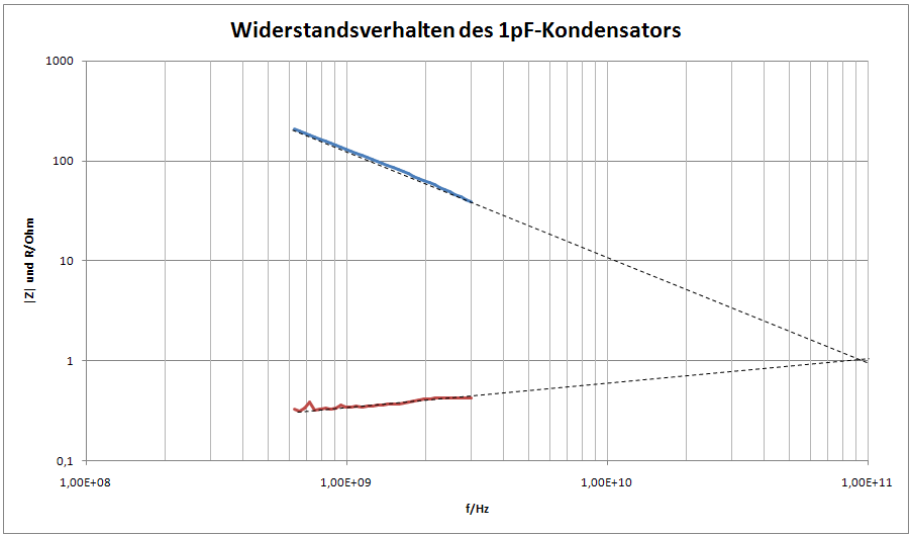

Bild 1.6: Verhalten des realen Kondensators in Abhängigkeit von der Frequenz

Der komplexe Widerstand eines Kondensators verhält sich in einem Frequenzbereich bis mindestens 12 GHz kapazitiv.[2]
Dieser Frequenzbereich umfasst alle relevanten Frequenzen des Signals von Fast CameraLink.
Mit „relevanten“ Frequenzen ist hier folgendes gemeint: Man betrachtet beispielsweise die Frequenzen eines Rechtecksignals mit einer Grundschwingungsfrequenz von maximal $f_{max} = 1,6\,\text{GHz}$. Dies entspricht der maximalen Übertragungsfrequenz von Fast CameraLink beim Senden einer Null-Eins-Folge. Relevant sind hier alle Frequenzen bis zur siebten Oberwellenfrequenz - für $f = f_{max}$ wären das 11,2 GHz.
Der nächste Fouriersummand ist die neunte Oberwelle. Diese Frequenz passt nicht mehr in den Frequenzbereich von 12 GHz und muss zunächst vernachlässigt werden. Die Amplitude dieses Summanden beträgt aber nur noch 1/9 der Grundwellenamplitude. Auch die Grundschwingungsfrequenz (1,6 GHz) der Signale zählt zu den relevanten Frequenzen.
Der Frequenzbereich $f \leq 12\,\text{GHz}$ könnte höchstwahrscheinlich noch weiter ausgedehnt werden. In Abschnitt 5.3 auf Seite 65 ist ersichtlich, dass sich die neunte Oberwellenfrequenz von 14,4 GHz ebenfalls miteinbeziehen ließe.

Die Tatsache, dass sich der komplexe Widerstand bis mindestens 12 GHz kapazitiv verhält, lässt sich wie folgt plausibel machen:
Die zwei Kurven in Bild 1.6 gelten lediglich für Frequenzen bis zu 3 GHz, da das Datenblatt nur bis dahin reicht.[3]
Setzt man die beiden Kurven, die Geraden sind, im logarithmischen Maßstab als Ge-

[2] Diese Tatsache wird weiter unten plausibel gemacht.
[3] alle Grenzwerte gehen nur bis 3 GHz

raden fort, so erhält man einen Schnittpunkt an der Stelle $f = f_s$ bei ca. 90 GHz. Das Fortsetzen in Form einer Gerade bis kurz vor dem Schnittpunkt ist gerechtfertigt, da man aus dem Datenblatt von größeren Kapazitäten weiß, dass dort ebenfalls Geraden bis kurz vor dem Schnittpunkt vorliegen.[4] Außerdem sind die Kurven aus Bild 1.6 Geraden.
In Bild 1.7 wird ein solches Verhalten eines 100 pF-Kondensators der gleichen Bauform demonstriert. Auch dort sind die Geraden im logarithmischen Maßstab aufgetragen.
Wenn man nun noch berücksichtigt, dass die Fortsetzung der beiden Kurven von Bild 1.6 ungefähr vor dem Schnittpunkt von Geraden auf gekrümmte Kurven übergeht, dann gilt, dass der Schnittpunkt höchstwahrscheinlich oberhalb von 22 GHz liegt. Das bedeutet, dass der komplexe Widerstand eines Kondensators für einen Frequenzbereich unterhalb von 22 GHz kapazitiv ist. Um ganz sicher zu gehen, wird dieser Bereich auf weniger als 12 GHz eingeschränkt.

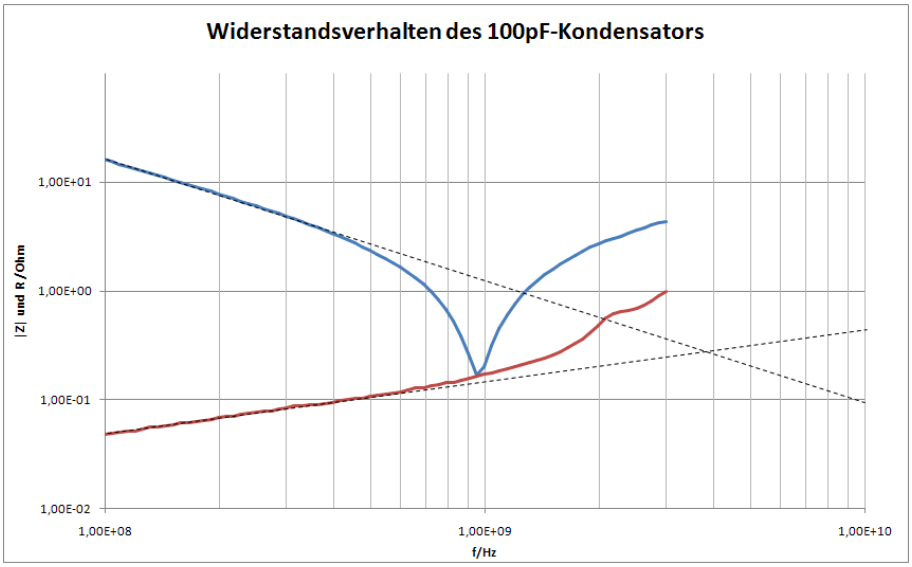

Bild 1.7: Verhalten des realen 100 pF-Kondensators in Abhängigkeit von der Frequenz

[4] Bei Datenblättern größerer Kondensatoren liegen die Schnittpunkte in einem Bereich unter 3 GHz.

2 Planung

In diesem Kapitel geht es darum, die bereits vorhanden bzw. die noch zu erstellenden Module zu erläutern. Der Testaufbau wird festgelegt und die Anforderungen an die noch zu erstellenden Platinen werden spezifiziert.

2.1 Verfügbare/ anzufertigende Module

2.1.1 Baugruppe (Kamera)

Als Baugruppe/ Kamera, bei der der Phy später zum Einsatz kommen soll, wird die A400 verwendet. Diese Kamera lässt sich mit einem 4 Mpix-CMOS-Bildsensor anfertigen, der 200 $^{\text{Bilder}}/_{\text{sec}}$ an den PC sendet. Bei einer 10 Bit-breiten Auflösung pro Pixel ergibt sich damit die folgende maximale Datenraten, die an den PC übertragen werden muss:

$$(2352 \cdot 1726)\,\frac{\text{Pixel}}{\text{Bild}} \cdot 10\frac{\text{Bit}}{\text{Pixel}} \cdot 200\frac{\text{Bilder}}{\text{Sekunde}} = 8,119\,^{\text{GBit}}/_{\text{sec}} \tag{2.1}$$

Im Übrigen schafft diese Baugruppe bestmögliche Voraussetzungen für die Erfüllung des SSTL_2 Class 1-Standards[1] beim FPGA, das sich auf dem Processing-Board befindet.
Für den Aufbau einer solchen Baugruppe sind drei Platinen erforderlich. In Bild 2.1 werden Aufbau und Verbindungen dieser Platinen dargestellt.
Auf dem Sensor-Board befinden sich sowohl der Sensor als auch die Beschaltung, die für

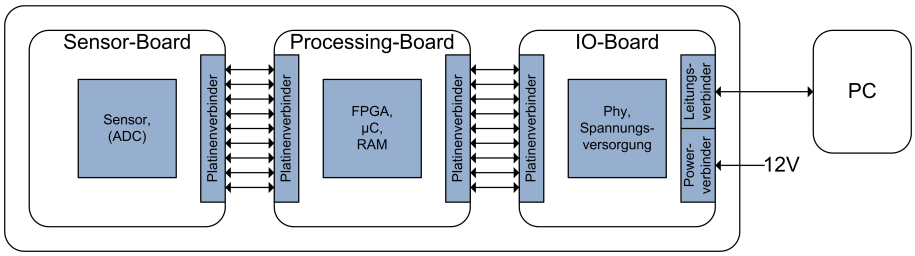

Bild 2.1: Baugruppe/ Kamera

den Sensor benötigt wird. Dieser Sensor übergibt die Daten an das Processing-Board und gegebenenfalls überarbeitet das auf dem Processing-Board befindliche FPGA diese Daten. Außerdem können im FPGA zusätzlich noch Bildkorrekturen durchgeführt werden.

[1] vgl. Abschnitt 3.1

Im nächsten Schritt findet eine Weitergabe der Bilddaten an das IO-Board statt. Auf diesem Board befindet sich die Beschaltung, die zur Versorgung aller Platinen mit den jeweiligen Spannungen und zur Versendung der Daten vom FPGA an den PC benötigt wird.
Zur Verwendung und Untersuchung des Phys muss das IO-Board neu erstellt werden. Da das Processing-Board ebenfalls für die Tests benötigt wird, ist es zur Betreibung der Platinen erforderlich, alle notwendigen Spannungen auf dem IO-Board zu erzeugen.

2.1.2 Kabel

Zur Datenübertragung an den PC wird das Kabel 74506-3107 [3] von der Firma Molex[2] verwendet. Dieses Kabel eignet sich für die hier vorliegenden Anforderungen am besten, da es für einen Standard erstellt worden ist, bei dem die Datenübertragungsrate einen ebenso hohen Wert wie dieser Phy besitzt.
Zur Inbetriebnahme des Kabels müssen zuerst die Filterglieder, die sich im Stecker des Kabels befinden, entfernt werden. Diese Filterglieder finden bei diesem Standard keine Verwendung.

2.1.3 Evaluierungs-Board

Zusätzlich wird ein Evaluierungs-Board von der Firma Texas Instruments verwendet. Auf diesem Evaluierungs-Board befinden sich ein FPGA von Xilinx, ein LC-Display, diverse Taster und LEDs, die Spannungsversorgung und der Phy. Der gleiche Phy wie auf diesem Evaluierungs-Board wird auch auf dem IO-Board zum Einsatz kommen. In Bild 2.2 auf der nächsten Seite ist die Verbindung der einzelnen Bauelemente abgebildet. LC-Display, Taster und LEDs werden vom FPGA gesteuert und abgefragt. FPGA und Phy sind über zwei Schnittstellen miteinander verbunden. Die erste Schnittstelle ist die Datenschnittstelle. Bei dieser werden die Daten vom FPGA an den Phy übergeben, damit dieser die Daten über das Kabel versenden kann. Sobald der Phy die Daten empfängt, findet eine Weitleitung der Daten an das FPGA statt. Die zweite Schnittstelle ist die Konfigurationsschnittstelle. Über diese Schnittstelle kann das FPGA den Phy einstellen. Die vom Phy kommenden Daten werden über SMA-Verbinder mit einer weiteren Platine verbunden, auf der sich der für die Anbindung an das Kabel erforderliche Stecker befindet. Da es eine solche Platine nicht gibt, ist auch diese noch zu erstellen.

2.1.4 Störglieder

Zur Störung des Phys bei seiner Übertragung ist das Erstellen einer Platine mit Störgliedern erforderlich. Dabei muss darauf geachtet werden, dass auch wie auf dem IO-Board

[2] www.molex.com

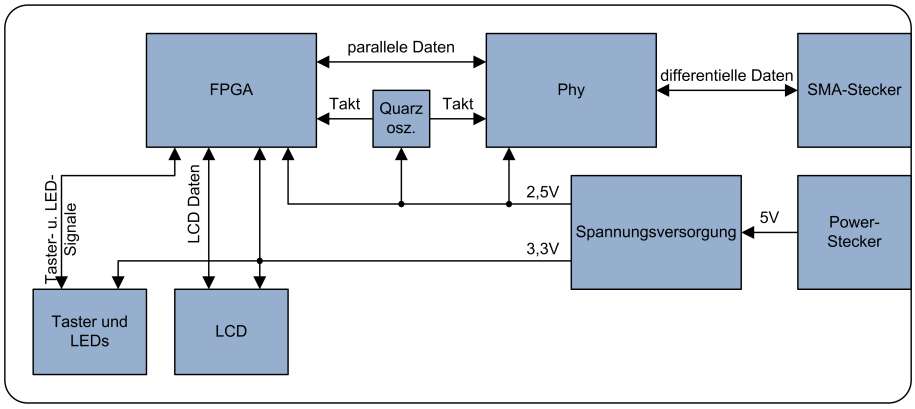

Bild 2.2: Aufbau des Evaluierungs-Boards

der Wellenwiderstand der differentiellen Leitungen eingehalten wird. Zur Erstellung immer anderer Störglieder ist auf dieser Platine ein Raster erstellt worden, mit dem die Störglieder aufgebaut werden können. In Bild 2.3 kann man die fertige Platine mit dem Raster für die einzelnen Störglieder sehen.

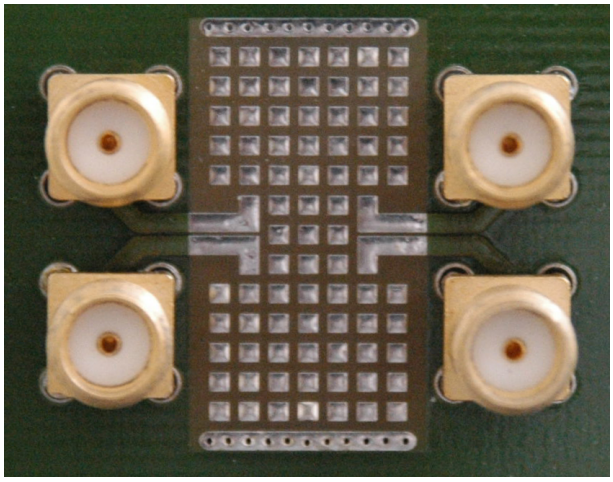

Bild 2.3: Platine mit SMA-Verbindungssteckern und Raster zum Aufbauen der einzelnen Störglieder

2.1.5 Adapter-Platine

Zur Verwendung der verschiedenen Störglieder mit dem Kabel ist es erforderlich, eine Adapter-Platine zu erstellen. Diese Platine beinhaltet den Stecker für das Kabel und SMA-Verbinder zur Anschließung der Störglieder. Da die Platine keine zusätzlichen Störungen erzeugen soll, muss auf den Wellenwiederstand der Leitung geachtet werden.

Mit dem Programm HyperLynx [15] sind Abstand und Breite der einzelnen Leitungen bestimmt worden. Somit werden für den Abstand der differentiellen Leitungen 150 μm und für die Breite der Leitungen 350 μm angesetzt. Außerdem ist es erforderlich, die Leitungen auf dieser Platine, die vom Kabel-Verbinder zu den SMA-Steckern führen, nach Möglichkeit mit gleichen Längen zu versehen. Hierbei nimmt man eine zulässige Toleranz[3] zwischen den einzelnen Leitungen von 1 mm an.

2.2 Testkonzepte

Der folgende Abschnitt handelt von der Bestimmung der Tests und somit insbesondere von der Spezifizierung des genauen Aufbaus und den Anforderungen an die Platine. Bei den Tests geht es hauptsächlich darum, die Bitfehlerrate des Phys und dessen Verhalten zu untersuchen. Dabei ist die Bitfehlerrate durch das Verhältnis der falsch empfangenen Bits zu den insgesamt gesendeten Bits definiert.
Die späteren Tests des Phys sind in drei Testschritte unterteilt:

- Test 1:
  Dämpfungstest
- Test 2:
  Tiefpasstest
- Test 3:
  Preemphasistest

Dämpfungstest
Bei diesem Test soll durch das Hinzuschalten verschiedener Dämpfungsglieder die Bitfehlerrate ermittelt werden. Der Vergleich eines Dämpfungsglieds mit einer Leitungslänge ist erlaubt, da jede Leitung eine Dämpfung des Signals bewirkt.

Tiefpasstest
Dieser Test dient zur Ermittlung der Bitfehlerrate mittels Hinzuschalten verschiedener Tiefpassfilter. Da jede Leitung - wie schon in Abschnitt 1.2 beschrieben - eine Tiefpasscharakteristik besitzt, hängt die Leitungsdämpfung nicht nur von der Länge, sondern auch von der zu übertragenden Frequenz ab. Daher kann ein Tiefpassfilter eine frequenzanhängige Leitungsdämpfung simulieren.

Preemphasistest
Hierbei schaltet man die Preemphasis ab und verwendet ein Dämpfungsglied und einen Tiefpassfilter, bei denen noch Bitfehler ermittelt werden können. Bei diesem Test wird also das Verhalten des Phys in Zusammenarbeit mit der Preemphasis untersucht.

[3] vgl. Abschnitt 2.4

2.3 Spannungsversorgung

Auf der Platine müssen verschiedene Spannungen erzeugt und bereitgestellt werden. Es ist erforderlich, diese Spannungen mit Hilfe von Schaltreglern aus der Versorgungsspannung von 12 V herzustellen. Vor der Spannungserzeugung muss die Eingangsspannung gefiltert werden. Dies ist notwendig, um einerseits eine Störungsübertragung von den Schaltreglern auf das eingespeiste Netz zu verhindern und um andererseits einer Übertragung eventueller Störungen aus dem Netz auf die IO-Platine entgegenzuwirken.
Die Platine besitzt eine 5 V-Spannungsversorgung, die aber nicht benötigt wird. Diese Spannungsversorgung kann noch bei späteren Veränderungen auf dem Processing- oder Sensor-Board verwendet werden.
Die 3,3 V- und die 2,5 V-Spannungsversorgungen sind für den Phy und das FPGA erforderlich. Da der Phy eine sehr hohe Stromaufnahme hat und auch nur mit 2,5 V versorgt wird, ist es erforderlich bei der Erzeugung dieser Spannung auf die Strombelastbarkeit der verwendeten Bauteile zu achten.

2.4 Platinenkonzept

Wie schon in Abschnitt 2.1 auf Seite 9 beschrieben, müssen auf der Platine des IO-Boards die Spannungsversorgung, der Platinensteckverbinder, die Beschaltung der digitalen Ein- und Ausgänge, der Fast CameraLink-Stecker, der Spannungsversorgungsstecker und der Phy Platz finden. Der Platinenverbinder für das Processing-Board hat auf der Platine schon eine feste Stelle, da er auf die Oberseite der Platine gehört und somit die Lage der anderen Stecker bestimmt: Diese müssen auf die Unterseite der Platine gesetzt werden. In Bild 2.4 auf der nächsten Seite ist zu sehen, dass der Platinenverbinder auf die linke Seite der Platine zu setzen ist und die anderen Stecker auf der rechten Seite zu platzieren sind. Dies liegt an der Tatsache, dass die fertige Platine später beim Einbauen in eine Kamera auf der rechten Seite mit der Kamerarückwand fest verbunden werden muss und auf der linken Seite nur durch den Platinenverbinder zusammengehalten wird. Mit dieser Vorgehensweise ist es möglich den Toleranzen der mechanischen Fertigung des Gehäuses entgegenzuwirken. Außerdem muss für die Region mit den digitalen Ein- und Ausgängen ein gesonderter Bereich auf der Platine entstehen, da diese mit Optokopplern geschützt werden. Dieser Bereich sollte sich so dicht wie möglich an dem Spannungsversorgungsstecker, der auch die digitalen Ein- und Ausgänge beinhaltet, befinden.
Desweiteren ist bei der Spannungsversorgung darauf zu achten, dass die Bauteile, die für die Spannung des Processing-Boards bestehen, alle auf der Unterseite der Platine liegen. Dies hat den Vorteil, dass die Schaltregler die Signale auf den anderen Lagen nicht stören.
Angesichts der Tatsache, dass die Signale für die differentielle Datenübertragung einen Wellenwiderstand von 100 Ω und die Signale der parallelen (asymmetrischen) Datenübertragung einen Wellenwiderstand von 50 Ω haben müssen, ist es erforderlich, beim Lagenaufbau die Abstände zwischen den einzelnen Lagen und das Dielektrikum zu be-

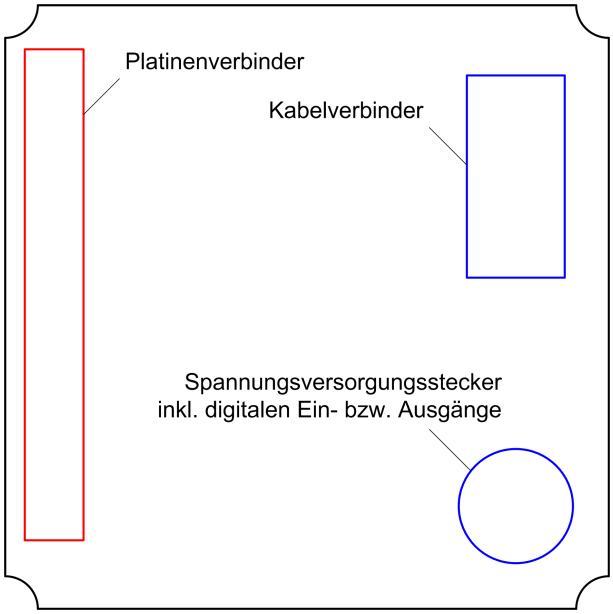

Bild 2.4: Anordnung der Stecker auf der fertigen Platine

rücksichtigen. Zur Einhaltung der Wellenwiderstände der Leitungen werden außerdem die kritischen Signale; d.h. die Signale, bei denen der Wellenwiderstand einzuhalten ist; auf die verschiedenen Lagen verteilt. Aus diesem Grund ist der Lagenaufbau - wie in Bild 2.5 auf der nächsten Seite dargestellt - verwendet worden. Auf die beiden Innenlagen, MidLayer 2 und 3, werden die Signale gelegt, die für die parallele Datenübertragung zu verwenden sind. Die differentiellen Signale belegen den MidLayer 4. Mit dem Programm HyperLynx [15] sind die Abstände zwischen den einzelnen Lagen und den einzelnen Leitungen bestimmt worden. Diese Abstände dienen dazu, bei den differentiellen Leitungen auf MidLayer 4 einen Wellenwiderstand von 100 Ω und bei den parallelen Leitungen einen Wellenwiderstand von 50 Ω zu erzeugen. Auf diese Weise ergibt sich der in Tabelle 2.1 auf der nächsten Seite beschriebene Lagenaufbau.

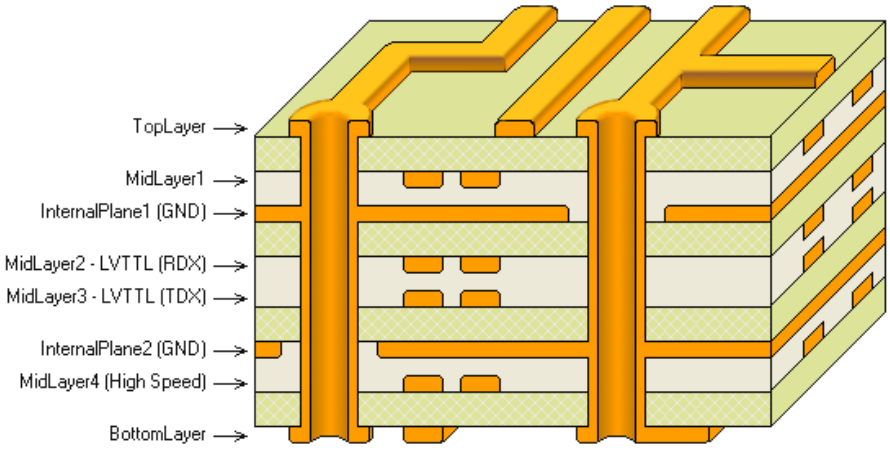

Bild 2.5: Lagenaufbau der Platine

Lagenname	Dicke
TopLayer	43 μm
FR4	60 μm
MidLayer 1	35 μm
Prepreg	150 μm
InternalPlane1 (GND)	35 μm
FR4	100 μm
MidLayer2 (RXD)	35 μm
Prepreg	660 μm
MidLayer3 (TXD)	35 μm
FR4	100 μm
InternalPlane2 (GND)	35 μm
Prepreg	150 μm
MidLayer4 (HighSpeed)	35 μm
FR4	60 μm
BottomLayer	43 μm

Tabelle 2.1: Abstände für den Lagenaufbau der Platine

Außerdem muss auf die Länge der einzelnen Leitungen geachtet werden, da sich ansonsten die Signale auf den verschiedenen Leitungen mit unterschiedlicher Geschwindigkeit ausbreiten. Andernfalls könnte es zu Störungen kommen, da die einzelnen Signale nicht mehr zeitgleich eintreffen. Infolgedessen muss für die differentiellen und parallelen Leitungen ein Leitungslängenausgleich durchgeführt werden.
Die Ausbreitungsgeschwindigkeit eines Signals auf der Platine hängt vom Dielektrikum ab und lässt sich durch Formel 2.2 auf der nächsten Seite folgendermaßen beschreiben

[19, Seite 1198]:

$$v = \frac{c_0}{\sqrt{\mu_r \epsilon_r}} \tag{2.2}$$

In der obigen Formel kann für μ_r 1 eingesetzt werden, da in den Leitungen im Allgemeinen keine magnetischen Stoffe vorhanden sind. Setzt man nun ϵ_r gleich 4,3 und die Lichtgeschwindigkeit c_0 gleich $2{,}9979 \cdot 10^8$ m/s, so erhält man eine Ausbreitungsgeschwindigkeit von:

$$v = \frac{2{,}9979 \cdot 10^8 \,\mathrm{m/s}}{\sqrt{4{,}3}} = 0{,}1445727 \,\mathrm{mm/ps} \tag{2.3}$$

Da im Datenblatt keine Verzögerungszeiten für die parallelen und differenziellen Leitungen angegeben sind, ist es erforderlich, bei den parallelen Leitungen auf die Setup- und Hold-Zeit zurückgreifen. Für das Senden liegt diese bei 480 ps und für das Empfangen bei 960 ps. Um allerdings definitiv unter diesem Wert zu bleiben, ist es empfehlenswert, nur 5 % des kleineren Wertes der Setup- und Hold-Zeit zu verwenden. Unter Berücksichtigung dieser Tatsache ergibt sich dann für das Senden eine maximale Zeit von 24 ps, was einen Leitungslängenunterschied von 3,46974 mm zur Folge hat. Für das Empfangen resultiert aus der oben genannten Annahme eine Maximalmalzeit von 48 ps, wodurch eine Leitungslängendifferenz von 6,9394 mm entsteht. Somit wird für den Leitungslängenausgleich eine maximale Leitungslängendifferenz von 3 mm verwendet.
Auf Grund der Tatsache, dass das FPGA ebenfalls eine Verzögerungszeit besitzt, die beim Senden der Daten vom Flip Flop zum Pad eintritt, muss diese ebenfalls in der Berechnung der Gesamtverzögerungszeit berücksichtigt werden. Verwendet man den Ausgang als einfachen digitalen Ausgang, so wird im Datenblatt des FPGAs [8, Seite 232] eine Verzögerungszeit vom Ausgangs-Flip Flop zum Pad von circa 170 ps angenommen. Mit dieser Angabe, der in Formel 2.3 errechneten Zeit und der maximalen Leitungsdifferenz von 3 mm ergibt sich eine maximale Verzögerungszeit von:

$$\begin{aligned} t_{max} &= \frac{l_{Differenz}}{v_{Leitung}} + t_{FPGA} \\ &= \frac{3\,\mathrm{mm}}{0{,}1445727 \,\mathrm{mm/ps}} + 170\,\mathrm{ps} \\ &= 190{,}75\,\mathrm{ps} \end{aligned} \tag{2.4}$$

Die errechnete maximale Verzögerungszeit ist immer noch deutlich geringer als die Setup- und Hold- Zeit. Demzufolge werden bei einem Leitungslängenausgleich von bis zu 3 mm keinerlei Störungen in Bezug auf verspätet ankommende Signale auftreten.
Bei den differenziellen Leitungen muss ansonsten nichts weiter berücksichtigt werden, da der Takt in den Daten integriert ist und für jede Lane mitgesendet wird. Man sollte lediglich darauf achten, dass bei den differentiellen Leitungen jedes Paar ungefähr die gleiche Länge hat. Hierbei ist eine Leitungslängentoleranz von 3 mm - ebenso wie bei den parallelen Leitungen - zulässig.

3 Hardware

In Bild 3.1 wird der Gesamtaufbau der Platine inkl. der Kommunikation nach außen dargestellt.
Durch die Verwendung des Processing-Boards, auf dem sich das FPGA, der Mikrokon-

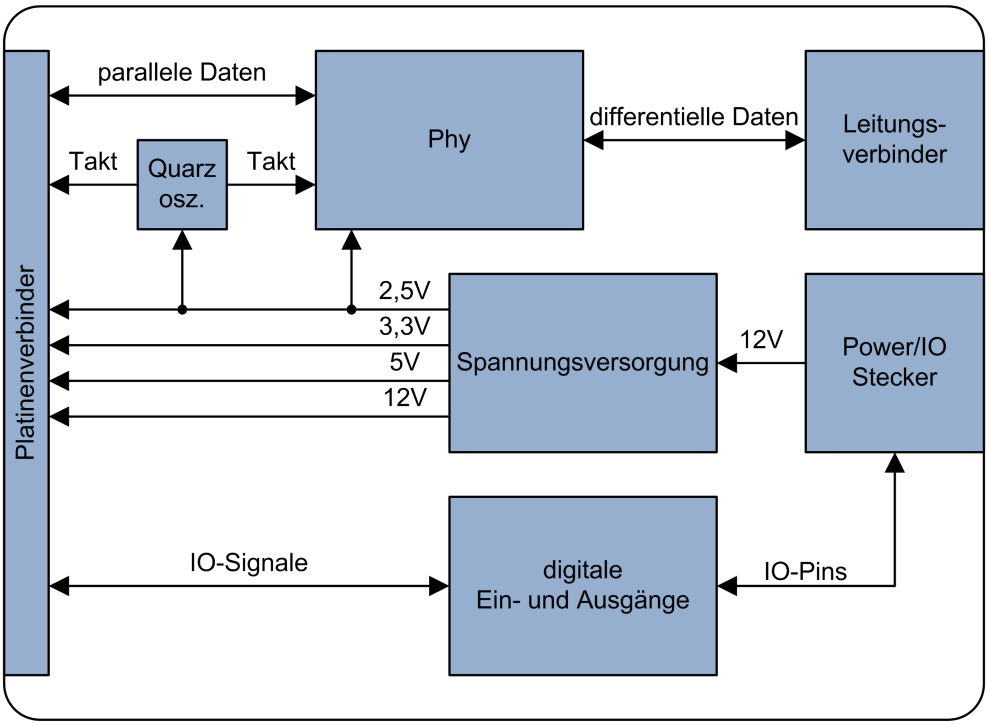

Bild 3.1: Blockschaltbild der IO-Platine

troller und die Rams befinden, werden Kommunikation und Bereitstellung der Spannungen über den Platinensteckverbinder realisiert. Alle Signale und erzeugten Spannungen sind somit an diesen Verbinder angeschlossen. Die Belegung des Verbinders wird durch den Aufbau des Processing-Boards festgelegt.
Infolgedessen muss die Spannungsversorgung auf der Platine erzeugt und bereitgestellt werden. Außerdem sind die digitalen Ein- und Ausgänge noch aufzubereiten.

3.1 Phy

Der Phy[1], der sich im ISO/OSI-Schichtenmodell [21] in der Bitübertragungsschicht befindet, erhält die Daten vom FPGA, bereitet diese auf und überträgt sie. Die Daten, die der Phy vom FPGA erhält, sind parallel und die Daten, die der Phy versendet, seriell.

3.1.1 Parallele Datenübertragung

Zur Absicherung der Datenübernahme vom FPGA erhält der Phy zu den übergebenen Daten einen Takt. In Bild 3.2 wird die parallele Datenübertragung zum Phy dargestellt. Der Phy übernimmt die Daten bei steigender und fallender Flanke des Takts TCA (Double-

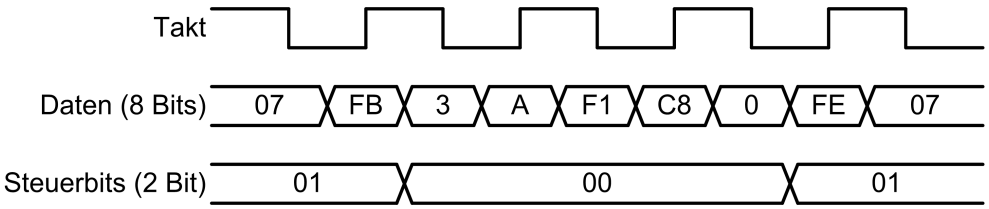

Bild 3.2: Parallele Datenübertragung zum Phy

leDataRate). Anschließend werden die Daten aufbereitet und differentiell versendet.
Sobald der Phy Daten empfängt, übergibt er diese an das FPGA. Die Datenübertragung zwischen FPGA und Phy läuft in beiden Richtungen auf die gleiche Art und Weise ab.
Zusätzlich zu den eigentlichen Nutzdaten müssen Steuerkommandos versendet werden. Diese Steuerkommandos enthalten Informationen darüber, ob ein Packet mit Nutzdaten beginnt oder endet. Das Steuerkommando *0x1FB* steht für Nutzdaten zu Beginn eines Packets und das Kommando *0x1FE* für Nutzdaten am Ende eines Packets.
Darüber hinaus lassen sich über die Steuerkommandos zusätzliche Informationen bezüglich des gerade gesendeten Bytes ausgeben. Zur Unterscheidung dieser Steuerkommandos von den Nutzdaten werden zusätzlich zwei Steuerbits mitgeführt. Anhand der Zustände dieser Steuerbits kann eine Entscheidung, ob es sich um Nutzdaten oder Steuerkommandos handelt, getroffen werden.
Da die parallele Datenübertragung mit einer Frequenz von 156,25 MHz erfolgt, ist es erforderlich, die Datenleitung mit einem geeigneten Widerstand abzuschließen. Dies wird durch die Erfüllung des „SSTL_2 Class 1“-Standards erwirkt.
Bei diesem Standard ist es erforderlich, dass der Leitungswiderstand, wie in Bild 3.3 auf der nächsten Seite dargestellt, 50 Ω beträgt. Außerdem muss der Ausgang ebenfalls einen 50 Ω-Widerstand in Reihe besitzen. Dieser Widerstand lässt sich über den „IO-Standard“ des einzelnen Pins vom FPGA einstellen. Beim Phy ist dieser Widerstand intern fest vorhanden. Ein derartiges Vorgehen lässt sich mit der Vermeidung von Reflektionen auf den Leitungen begründen.
Außerdem ist beim Legen der einzelnen Leitungen - wie schon in Abschnitt 2.4 auf Seite

[1] Physikal Layer: integrierter Baustein zum Senden und Empfangen der Daten

13 beschrieben - darauf zu achten, dass diese die maximale Leitungsdifferenz von 3 mm zwischen Platinensteckverbinder und Phy nicht überschreiten. Bild 3.3 aus [18] veranschaulicht den Aufbau einer solchen Datenübertragung.
Damit der Phy-Eingang die Signale richtig interpretieren kann, muss der Kompara-

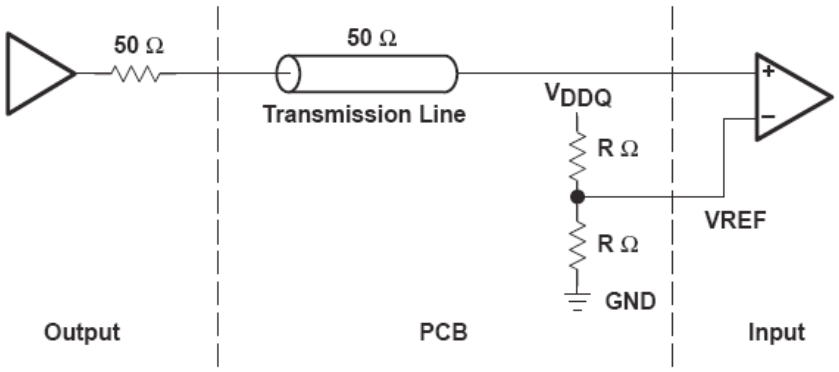

Bild 3.3: SSTL_2 Class 1-Standard

tor eine Referenz-Spannung von der halben Signalspannung besitzen. Diese Referenz-Spannung wird beim Phy über einen Widerstandsteiler von zwei gleichgroßen Widerständen erzeugt. Da der Phy den „HSTL Class 1“-Standard versteht, erkennt er an Hand der Referenz-Spannung, ob es sich um den HSTL- oder SSTL-Standard handelt.

3.1.2 Differentielle Datenübertragung

Zur schnellen und sicheren Datenversendung ist es nötig, die Nutzdaten zu kodieren. Aus diesem Grund verschlüsselt der Phy die Nutzdaten und Steuerkommandos in einen zehn Bit breiten Code, bevor er diese seriell versendet.
Durch die acht zu zehn Bit-Kodierung [20], die IBM entwickelt hat, ist es möglich, dem Datenstrom den Takt zu entnehmen. Der Vorteil besteht darin, dass zwischen Empfänger und Sender keine zusätzliche Taktleitung vorhanden sein muss und sich diese daher nicht auf einen festen Takt einigen brauchen. Desweiteren wird bei einer acht zu zehn Bit-Kodierung darauf geachtet, dass der Gleichspannungspegel auf der differentiellen Leitung null ist. Dies erfolgt auf die Art und Weise, indem genauso viele Einsen wie Nullen pro zehn Bit gesendet werden.
Die acht zu zehn Bit-Kodierung setzt sich aus einer drei zu vier und einer fünf zu sechs Bit-Kodierung zusammen. Die oberen drei Bit werden zu einem vier Bit-Code und die unteren fünf Bit zu einem sechs Bit-Code kodiert und danach zusammengeführt. In Tabelle A.2 auf Seite 76 ist die drei zu vier Bit-Kodierung und in Tabelle A.1 auf Seite 75 die fünf zu sechs Bit-Kodierung dargestellt.
Zudem kann der Phy für die differentielle Datenübertragung eine sogenannte „Preemphasis“ aktivieren. Diese Preemphasis erhöht die Amplitude und erreicht somit ein besseres

Signal-Rausch-Verhältnis. Außerdem wirkt die Preemphasis der Tiefpasscharakteristik des Kabels entgegen. In Bild 3.4 wird das Ausgangsverhalten einer differentiellen Leitung bei aktivierter Preemphasis gezeigt.
Zudem ist dieses Verhalten nur möglich, wenn bei zwei aufeinanderfolgenden Einsen

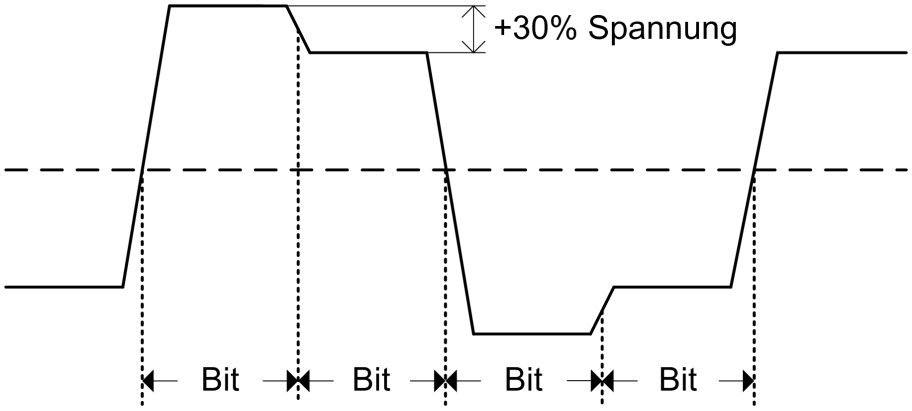

Bild 3.4: Ausgangsspannung einer differentiellen Leitung mit aktivierter Preemphasis

auch zwei aufeinander folgenden Nullen erzeugt werden, damit das Spannungspotential auf der differentiellen Leitung null bleibt.

3.1.3 Konfigurationsschnittstelle

Zur Durchführung einiger Einstellungen im Phy - wie zum Beispiel der Aktivierung der Zustandsautomaten für die Entzerrung (deskew) der ankommenden differentiellen Daten - besitzt dieser eine Konfigurationsschnittstelle. Diese Schnittstelle lässt sich über das FPGA ansprechen und wird als MDI[2]-Schnittstelle bezeichnet.
Bei der MDI-Schnittstelle werden Takt und Daten über zwei Leitungen übertragen. Zum Lesen und Schreiben der Konfigurationseinstellungen vom Phy verläuft die Datenübertragung bidirektional. Da FPGA und Phy simultan Daten senden könnten, arbeiten beide passiv: Dabei ist ein Pull-Up-Widerstand an die Datenleitung angeschlossen und FPGA und Phy schalten diese Datenleitung auf den Low-Pegel. Insofern kann bei FPGA und Phy die Treiberstufe durch ein Fehlverhalten nicht zerstört werden. Beim Takt ist kein Pull-Up-Widerstand erforderlich, da dieser vom FPGA bereitgestellt und generiert wird. Der Phy kann diesen Takt nur lesen.
Der notwendige Datenstrom muss den in Bild 3.5 auf der nächsten Seite dargestellten Ablauf aufweisen:
Der Phy agiert als Empfänger und die Rolle des Senders wird vom FPGA übernommen. Jeder Datenstrom beginnt mit einem sogenannten SOF (Start of Frame). Dies ist das einleitendende Kommando und alle am Bus angeschlossenen Teilnehmer lauschen auf die noch folgenden Kommandos. Hiernach wird das R/W (Read/Write) -Kommando gesen-

[2]Management Data Interface

SOF	R/W	Addr.	Reg.-Addr.	Turn Around	DATA

Bild 3.5: Aufbau eines MDI-Datenstroms

det. Dieses gibt an, ob auf dem Empfänger gelesen oder geschrieben werden soll. Hieran schließt das „Address“-Kommando an, gefolgt von dem „Reg.-Address“-Kommando, welches die Adresse des Empfängers und des Registers beinhaltet. Nun fährt das FPGA mit einem Wechsel (Turn-Around) fort, um zu signalisieren, dass die Steuerkommandos beendet sind. Zum Schluss werden noch die Daten gesendet bzw. empfangen, je nachdem was vorher vereinbart worden ist.
Sieht man sich hierzu die einzelnen Bitströme im Bild 3.6 an, so lässt sich der genauere Unterschied zwischen dem Senden und dem Empfangen der Daten erkennen.
Das R/W-Kommando lautet beim Senden „0 1“ und beim Empfangen „1 0“. Zudem

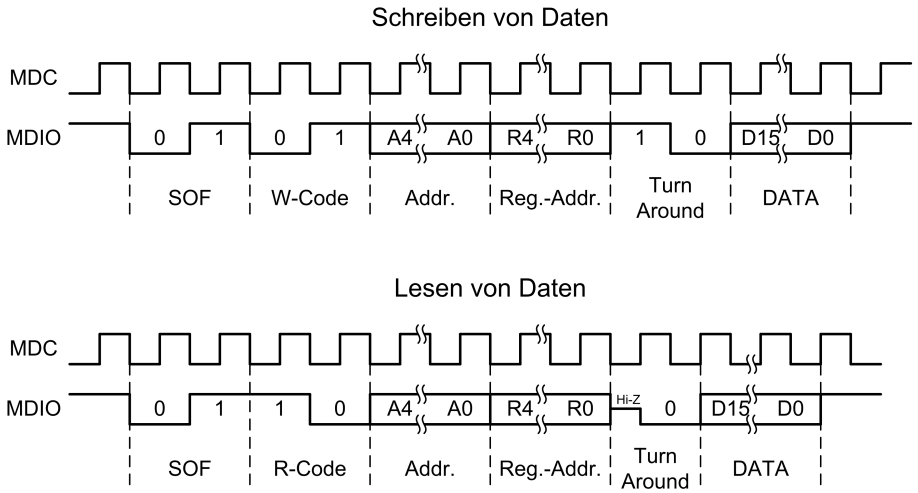

Bild 3.6: Schreiben und Lesen der Register über die MDI-Schnittstelle

ist noch ersichtlich, dass die Daten während der Steuerkommandos bei der steigenden Flanke vom Takt anliegen müssen. Erst das „Tourn-Around“ ändert gegebenenfalls die Datenflussrichtung. Im Falle, dass der Sender Daten an den Empfänger schicken muss, übergibt dieser die Daten bei jeder steigenden Flanke vom Takt an den Empfänger. Wenn allerdings vom Empfänger gelesen werden soll, überträgt der Empfänger bei jeder fallenden Flanke die Daten an den Sender.
Angesichts der Tatsache, dass der Phy auch noch einen erweiterten Adressraum mit einer 16 Bit breiten Adresse besitzt und zwei verschiedene Betriebsmodi kennt, um einige andere Standards zu erfüllen, wird an dieser Stelle kurz darauf eingegangen, auf welche Weise diese erweiterten Register geschrieben bzw. gelesen werden:
Zum Ansprechen einer erweiterten Adresse ist es allerdings erforderlich, geringfügige Änderungen im Bitstrom vorzunehmen. In Bild 3.7 auf der nächsten Seite ist der Aufbau

der drei Bitströme zu sehen.
Der Start-Code lautet nun nicht mehr „0 1“, sondern „0 0“. Damit kann der Phy er-

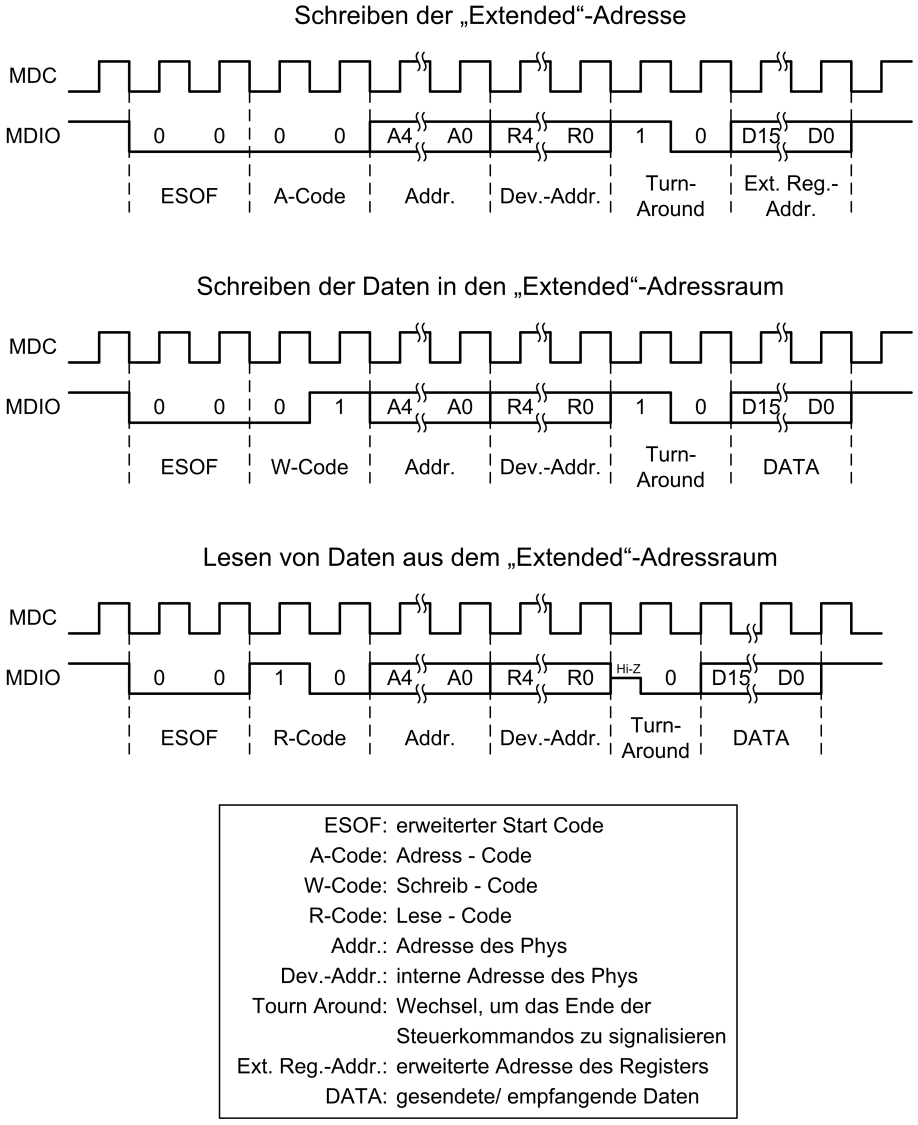

Bild 3.7: Schreiben und Lesen der erweiterten Register über die MDI-Schnittstelle

kennen, dass es sich nun um ein Kommando für den erweiterten Adressraum handelt. Außerdem wird als „Reg.-Address“ die Adresse der Betriebsart ausgewählt.
Zum Auslesen bzw. Hineinschreiben von Daten aus einem Register ist es wichtig, zunächst dem Phy mitzuteilen, um welches Register es sich handelt. Danach kann das FPGA die Daten aus dem Phy mit einem Schreib- bzw. Lesezugriff auslesen bzw. hineinschreiben. Dies geschieht auf folgende Weise:
Zuerst sendet das FPGA den Startcode „0 0“, dann die Register-Adresse der Betriebs-

art und zum Schluss die Adresse des erweiterten Registers in dem Datenbereich an den Phy. Dieser übernimmt die Daten und ist nun informiert darüber, dass es sich bei dem nächsten Schreib- bzw. Lese-Kommando um das zuvor gesendete Register handelt. Beim Schreiben/ Lesen wird „0 0“ ebenfalls als Startcode verwendet. Die Register-Adresse ist wieder die der Betriebsart. Der restliche Ablauf findet analog zum gewöhnlichen Schreiben bzw. Lesen statt.

3.2 FPGA

Das FPGA befindet sich - wie schon in Abschnitt 2.1 auf Seite 9 beschrieben - auf dem Processing-Board und wird über den Platinensteckverbinder mit dem IO-Board verbunden.
Der für den Phy notwendige Referenz-Takt ist zum FPGA zu leiten. Da sich jedoch Phy und FPGA auf unterschiedlichen Platinen befinden, muss der differentielle Takt in einen Single-ended-Takt umgewandelt werden, bevor dieser zum FPGA geführt wird. Die Umwandlung eines differentiellen Taktes in einen Single-ended Takt erfolgt mittels eines Wandlerbausteins. Dies ist notwendig, da es ansonsten zu Reflektionen auf der Leitung kommen kann, die den Takt überlagern und diesen stören. In Bild 3.8 auf der nächsten Seite ist die Schaltung abgebildet, die aus einem differentiellen Takt des Quarzoszillators einen Single-ended Takt erzeugt. Bei der Erstellung des Layouts muss darauf geachtet werden, dass der Abstand zwischen dem Quarzoszillator und dem Phy ungefähr genauso groß wie der Abstand zwischen dem Wandlerbaustein und dem Quarzoszillator ist. Außerdem müssen die vom Quarzoszillator zum Phy und zum Wandlerbaustein führenden Leitungen in etwa die gleiche Länge haben, um Störungen durch Reflektionen auf den Leitungen zu vermeiden.
Bei der Inbetriebnahme des Phys muss darauf geachtet werden, dass die Pinbelegung des Processing-Boards mit der des IO-Boards übereinstimmt. Beim Processing-Board ist zusätzlich noch ein Widerstand umzubestücken, damit die beiden IO-Bänke am FPGA, die für die Kommunikation zu verwenden sind, mit einer Spannung von 2,5 V betrieben werden können. Außerdem ist es erforderlich, bei der Pinbelegung der IO-Bank darauf zu achten, dass bei dem Signal, das für den zu sendenden Takt (TCA) benötigt wird, mindestens zwei Pins an jeder Seite frei sind. Diese ersten beiden Pins werden direkt mit GND und die anderen beiden benachbarten Pins mit 2,5 V verbunden. Dadurch ergibt sich eine Pinbelegung, wie sie in Tabelle 3.1 auf der nächsten Seite beschrieben ist.

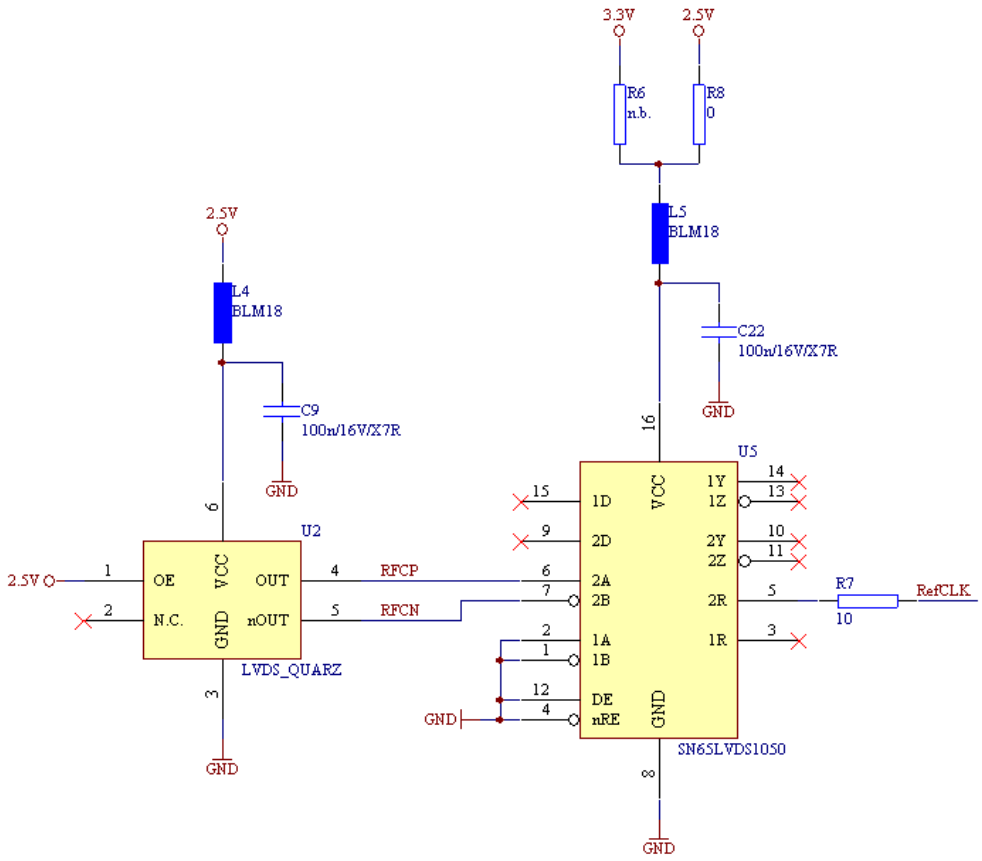

Bild 3.8: Beschaltung für den Takt des Phys und des FPGAs

Pin-Name des FPGAs	Pinbelegung
AA6	GND
Y6	2,5 V
A10	TCA
Y7	2,5 V
R4	GND

Tabelle 3.1: Pinbelegung des FPGAs für den Takt TCA

Diese Beschaltung der IO-Pins muss erwähnt werden, da der Takt von einem gewöhnlichen IO-Pin des FPGAs erzeugt wird, um dieselben Verzögerungen wie bei den Daten zu erhalten. Mit Hilfe der beiden Spannungen direkt am TCA-Pin ist es dem FPGA möglich, die steilen Flanken, die der Takt besitzen sollte, ordnungsgemäß hervorzubringen. Die zwei benachbarten Masse-Pins sollen den Takt so gut wie möglich abschirmen, damit dieser keine Störungen auf den anderen Signalleitungen vom FPGA verursacht.

Der Empfangstakt RCA wird auf einen dedizierten Takteingang gegeben, um den Takt auf eine PLL übertragen zu können. Die PLL „rastet“ sich dann auf diesen Takt ein. Das FPGA kann intern mit dem Takt aus der PLL weiterarbeiten.

3.3 Spannungsversorgung

12 V-Spannungsfilterung
Auf der Platine müssen die 5 V-, die 3,3 V- und die 2,5 V-Spannungen aus der einzuspeisenden 12 V-Spannung erzeugt werden[3]. Damit die Platine keine Störungen auf das eingespeiste 12 V-Netz zurückführt und andererseits keine Störungen vom eingespeisten Netz erhält, muss man die Eingangsspannung filtern. In Bild 3.9 ist der Aufbau des Filters für die 12 V-Spannung zu sehen.
Das Tiefpass-Filter, welches aus $L8$ (10 μH) und $C44$ (100 nF) besteht, filtert zuerst die

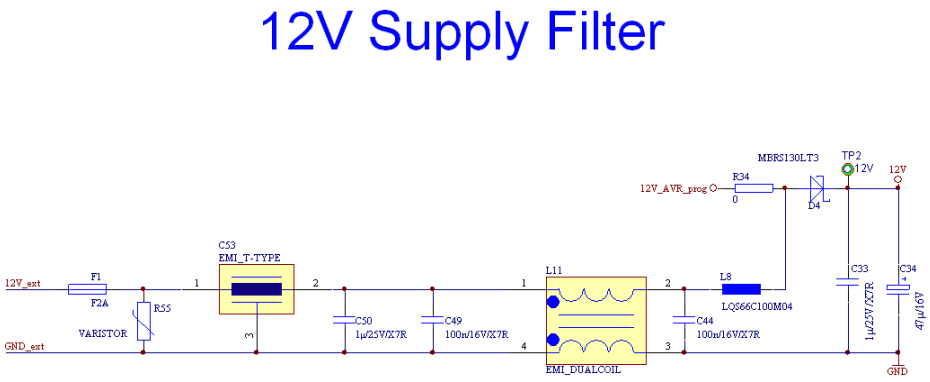

Bild 3.9: Filterung der eingespeisten 12 V-Spannung

Störungen bis zu einer Grenzfrequenz f_g von

$$\begin{aligned} f_g &= \frac{1}{2\pi\sqrt{L \cdot C}} \\ &= \frac{1}{2\pi\sqrt{10 \cdot 10^{-6}H \cdot 100 \cdot 10^{-9}F}} \\ &= 159,155\,\text{kHz} \end{aligned} \tag{3.1}$$

heraus. Die Gleichtakt-Spule[4] $L11$ ist für die Filterung der Störungen, die im Gleichtakt auftreten, zuständig. Dieses Filter sondert jede Gleichtaktstörung der Spannung aus. Durch das Tiefpass-Filter $C53$ wird die Spannung von hohen Frequenzen befreit. Im Datenblatt [6, Seite 1] ist zu sehen, dass die Grenzfrequenz des Filters bei circa 3 MHz

[3] siehe auch Abschnitt 2.3 auf Seite 13
[4] engl.: Common Mode Choke

liegt. Kleinere Frequenzen werden von den Kondensatoren $C49$ und $C50$ herausgefiltert. Der Varistor $R55$ schützt die gesamte Platine vor Überspannungen und die Diode $D4$ sichert diese vor einem falschen Anschließen der Versorgungsspannung. Insofern wird die angeschlossene Netzspannung von Störungen, die von der Platine oder aus dem Netz kommen, befreit und die Platine kann vor Überspannungen und vor dem Verpolen geschützt werden.

5 V-Spannungserzeugung

Die 5 V-Spannungsversorgung, die für spätere Änderungen bereitgestellt wird, ist mit dem Schaltregler $U6$ (LT1376 von TI) erzeugt worden. Bild 3.10 zeigt die Beschaltung des Schaltreglers, die erforderlich ist, um eine Spannungsversorgung von 12 V in eine 5 V-Spannung umzusetzen. Der grundlegenden Aufbau der Schaltung ist größtenteils

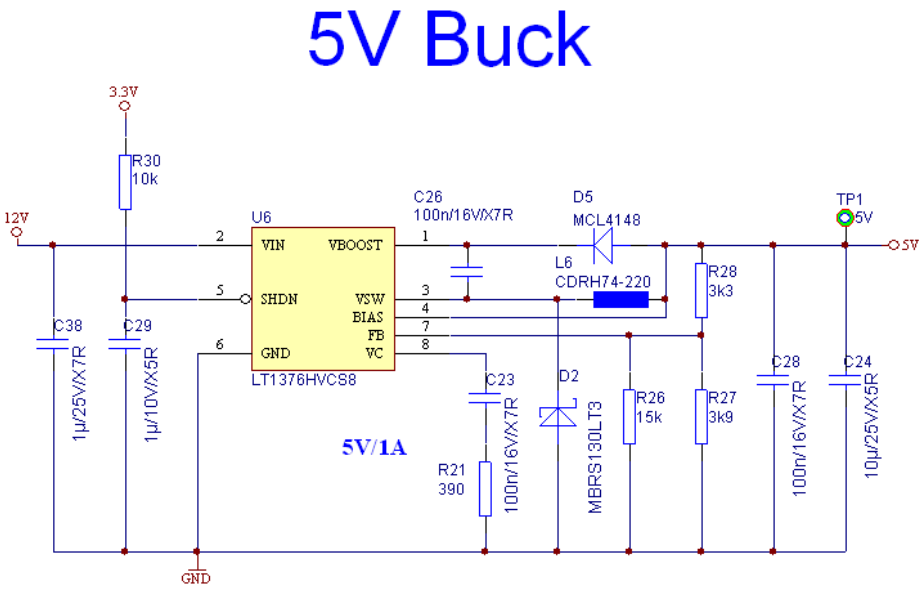

Bild 3.10: Erzeugung der 5 V-Spannung

aus dem Datenblatt [4, Seite 1] übernommen worden. Damit der Schaltregler eine Spannung einstellen kann, muss am FP-Eingang (Feedback) die Ausgangsspannung über einen Spannungsteiler zurückgeführt werden und der Mittelabgriff des Spannungsteilers eine Spannung von 2,24 V aufweisen. Auf diese Weise ist es dem Schaltregler möglich, eine Spannung einzustellen und diese zu halten. Da hier eine Spannung von 5 V am Ausgang anliegen soll und die abzugreifende Spannung bekannt ist, ist es erforderlich, einen Widerstandsteiler zu ermitteln, der eine abzugreifende Spannung von 2,24 V einstellt. Im Datenblatt [4, Seite 9] wird die Formel für die Berechnung des Widerstandsteilers angegeben mit

$$R28 = \frac{R26||27\,(V_{OUT} - 2,24\,\mathrm{V})}{2,24\,\mathrm{V}}. \tag{3.2}$$

Zur Erlangung möglichst genauer Werte werden für das Widerstandsnetzwerk drei Widerstände verwendet. $R28$ besteht aus einem und $R26||27$ aus zwei parallelgeschalteten Widerständen. Um nun die Berechnung etwas zu vereinfachen, wird $R28 = 3,3\,\mathrm{k\Omega}$ gewählt. Somit kann man die Formel 3.2 wie folgt nach $R26||27$ umstellen:

$$R26||27 = \frac{R28 \cdot 2,42\,\mathrm{V}}{V_{OUT} - 2,42\,\mathrm{V}} \tag{3.3}$$

Mit $V_{OUT} = 5\,\mathrm{V}$ ergibt sich für $R26||27$

$$\begin{aligned} R26||27 &= \frac{3,3\,\mathrm{k\Omega} \cdot 2,42\,\mathrm{V}}{5\,\mathrm{V} - 2,24\,\mathrm{V}} \\ &= 3,095\,\mathrm{k\Omega}. \end{aligned} \tag{3.4}$$

Mittels systematischen Probieren erhält man eine Widerstandskombination für $R26||27$ die aus einem $15\,\mathrm{k\Omega}$- und einem $3{,}9\,\mathrm{k\Omega}$-Widerstand besteht. Mit diesen Widerständen berechnet sich $R26||27$ wie folgt:

$$\begin{aligned} R26||27 &= \frac{3,9\,\mathrm{k\Omega} \cdot 15\,\mathrm{k\Omega}}{3,9\,\mathrm{k\Omega} + 15\,\mathrm{k\Omega}} \\ &= 3095,238\,\Omega \end{aligned} \tag{3.5}$$

Setzt man obiges $R26||27$ und $R28 = 3,3\,\mathrm{k\Omega}$ in die Formel 3.2 ein und formt diese nach V_{OUT} um, so ergibt sich für die theoretische Ausgangsspannung

$$\begin{aligned} V_{OUT} &= \frac{R1}{R2} \cdot 2,42\,\mathrm{V} + 2,42\,\mathrm{V} \\ &= 5,000\,\mathrm{V}. \end{aligned} \tag{3.6}$$

Zur Abwendung eines gleichzeitigen Starts aller Schaltregler bei der Spannungserzeugung - was im Übrigen zur Folge hätte, dass alle Einschaltströme zur gleichen Zeit anfallen würden - sind einige der Schaltregler voneinander abhängig. Um dies realisieren zu können, besitzt der hier verwendete Schaltregler einen Abschalt-Pin (SHDN), bei dem eine Spannung größer als 1 V anliegen muss, bevor dieser mit der Umsetzung beginnt. Damit zuerst die 3,3 V-Spannung erzeugt wird, ehe die 5 V-Spannungsumsetzung anfängt, ist es erforderlich, den SHDN-Pin des 5 V-Schaltreglers mit einem RC-Glied zu beschalten, das von der 3,3 V-Spannung gespeist wird. Im abgeschalteten Zustand ist der Kondensator entladen. Erst beim hinzuschalten der Spannungsversorgung der Platine kann der Kondensator geladen werden, nachdem der 3,3 V-Schaltregler mit der Umsetzung beginnt. Beim Aufladen des Kondensators verhält sich die Spannung am Kondensator in Abhängigkeit von der Zeit wie folgt:

$$U_c\,(t) = U \cdot \left(1 - e^{-\frac{t}{R \cdot C}}\right) \tag{3.7}$$

Wird diese Formel nach t umgestellt, so ergibt sich die Aufladezeit des Kondensators:

$$\begin{aligned} \frac{U_c(t)}{U} &= \cdot\left(1 - e^{-\frac{t}{R\cdot C}}\right) \\ e^{-\frac{t}{R\cdot C}} &= 1 - \frac{U_c(t)}{U} \\ -\frac{t}{R\cdot C} &= ln\left(1 - \frac{U_c(t)}{U}\right) \\ t &= -ln\left(1 - \frac{U_c(t)}{U}\right)\cdot R\cdot C \end{aligned} \tag{3.8}$$

Mit $U_c(t) = 1\,\mathrm{V}$, $U = 3,3\,\mathrm{V}$, $R = 10\,\mathrm{k\Omega}$ gewählt und $C = 1\,\mu\mathrm{F}$ lässt sich für die Aufladezeit t der folgende Wert bestimmen:

$$\begin{aligned} t &= -ln\left(1 - \frac{1\,\mathrm{V}}{3,3\,\mathrm{V}}\right)\cdot 10\,\mathrm{k\Omega}\cdot 1\,\mu\mathrm{F} \\ &= 3,610\,\mathrm{ms} \end{aligned} \tag{3.9}$$

Infolgedessen beginnt der Schaltregler für die 5 V-Spannung erst circa 3 ms Zeitversätzt nach Anliegen der 3,3 V. Auf diese Weise hat der Schaltregler für die 3,3 V-Spannung genügend Zeit, um diese Spannung einzustellen und die davon abhängenden Bauteile mit genügend Strom zu versorgen.

3,3 V-Spannungserzeugung

Für die 3,3 V-Spannungsversorgung ist derselbe Schaltregler wie für die 5 V-Spannungsversorgung verwendet worden. Aus diesem Grund können dieselben Formeln wie bei dem 5 V-Schaltregler vorausgesetzt werden. In Bild 3.11 auf der nächsten Seite ist die Beschaltung des Schaltreglers für die Erzeugung der 3,3 V-Spannung detailliert abgebildet. Damit sich am Ausgang des Schaltreglers eine Spannung von 3,3 V einstellen kann, muss am FB-Eingang auf Grund der analogen Vorgehensweise zum 5 V-Schaltregler eine Spannung von 2,42 V anliegen. Mit Formel 3.3, $V_{OUT} = 3,3\,\mathrm{V}$ und $R33 = 1,8\,\mathrm{k\Omega}$ erhält man für $R36$ einen Widerstand von

$$\begin{aligned} R36 &= \frac{1,8\,\mathrm{k\Omega}\cdot 2,42\,\mathrm{V}}{3,3\,\mathrm{V}\cdot 2,42\,\mathrm{V}} \\ &= 4,95\,\mathrm{k\Omega}. \end{aligned} \tag{3.10}$$

Setzt man $R36 = 4,7\,\mathrm{k\Omega}$ in die Formel 3.6 ein, so ergibt sich für V_{OUT} ein Wert von

$$\begin{aligned} V_{OUT} &= \frac{1,8\,\mathrm{k\Omega}}{4,7\,\mathrm{k\Omega}}\cdot 2,42\,\mathrm{V} + 2,42\,\mathrm{V} \\ &= 3,346\,\mathrm{V}. \end{aligned} \tag{3.11}$$

Damit der Schaltregler erst bei einer Eingangsspannung von mehr als 10 V mit der Umsetzung beginnt, wird der SHDN-Eingang wie im Datenblatt [4, Seite 15] ausgeführt beschaltet. Im Datenblatt [4, Seite 15] ist die Formel für R_{HI} in Abhängigkeit von R_{LO},

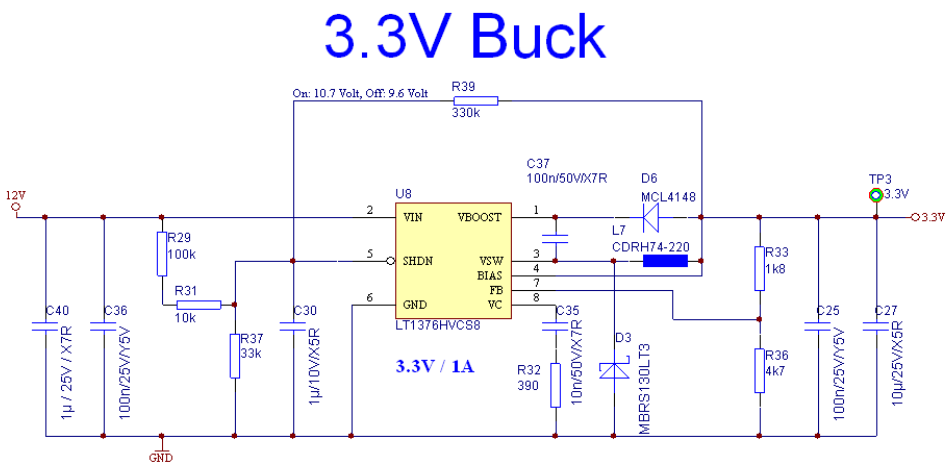

Bild 3.11: Erzeugung der 3,3 V-Spannung

V_{IN} (Abschaltspannung), ΔV (Differenz zwischen Ein- und Abschaltspannung), V_{IN} und V_{OUT} wie folgt angegeben:

$$R_{HI} = \frac{R_{LO}[V_{IN} - 2,38\,\mathrm{V}\left(\frac{\Delta V}{V_{OUT}} + 1\right) + \Delta V]}{2,38\,\mathrm{V} - R_{LO} \cdot 3,5\,\mu\mathrm{A}} \tag{3.12}$$

Mit $V_{IN} = 9\,\mathrm{V}$, $\Delta V = 1\,\mathrm{V}$, $V_{OUT} = 3,3\,\mathrm{V}$ und $R_{LO} = 33\,\mathrm{k\Omega}$ gewählt erhält man für R_{HI} schließlich

$$\begin{aligned} R_{HI} &= \frac{33\,\mathrm{k\Omega}[9\,\mathrm{V} - 2,38\,\mathrm{V}\left(\frac{1\,\mathrm{V}}{3,3\,\mathrm{V}} + 1\right) + 1\,\mathrm{V}]}{2,38\,\mathrm{V} - 33\,\mathrm{k\Omega} \cdot 3,5\,\mu\mathrm{A}} \\ &= 111,04\,\mathrm{k\Omega}. \end{aligned} \tag{3.13}$$

Angesichts der Tatsache, dass ein derartiger Widerstand nicht existiert, ist dieser durch eine Reihenschaltung aus zwei Widerständen, 100 kΩ und 10 kΩ, ersetzt worden. Ein solches Vorgehen sollte eigentlich vermieden werden, da sich die Toleranzen der Widerstände addieren, doch an dieser Stelle ist dies nicht so gravierend. Mit den oben genannten, neuen Widerstandswerten lässt sich nun die Ein- und Abschaltspannung berechnen. Dazu muss die Formel 3.12 nach V_{IN} wie folgt umgestellt werden:

$$\begin{aligned} \frac{R_{HI}}{R_{LO}} &= \frac{V_{IN} - 2,38\,\mathrm{V}\left(\frac{\Delta V}{V_{OUT}} + 1\right) + \Delta V}{2,38\,\mathrm{V} - R_{LO} \cdot 3,5\,\mu\mathrm{A}} \\ \frac{R_{HI}}{R_{LO}} \cdot (2,38\,\mathrm{V} - R_{LO} \cdot 3,5\,\mu\mathrm{A}) &= V_{IN} - 2,38\,\mathrm{V}\left(\frac{\Delta V}{V_{OUT}} + 1\right) + \Delta V \\ V_{IN} &= \frac{R_{HI}}{R_{LO}} \cdot (2,38\,\mathrm{V} - R_{LO} \cdot 3,5\,\mu\mathrm{A}) + 2,38\,\mathrm{V}\left(\frac{\Delta V}{V_{OUT}} + 1\right) - \Delta V \end{aligned} \tag{3.14}$$

Mit Hilfe der zuvor gewählten bzw. ermittelten Bedingungen, $R_{HI} = 110\,\text{k}\Omega$, $R_{LO} = 33\,\text{k}\Omega$, $\Delta V = 1\,\text{V}$ und $V_{OUT} = 3,3\,\text{V}$, ergibt sich für V_{IN} eine Spannung von

$$\begin{aligned} V_{IN} &= \frac{110\,\text{k}\Omega}{33\,\text{k}\Omega} \cdot (2,38\,\text{V} - 33\,\text{k}\Omega \cdot 3,5\,\mu\text{A}) + 2,38\,\text{V}\left(\frac{1\,\text{V}}{3,3\,\text{V}} + 1\right) - 1\,\text{V} \\ &= 9,649\,\text{V}. \end{aligned} \tag{3.15}$$

Folglich beginnt der Schaltregler erst mit der Umsetzung, wenn die Eingangsspannung $V_{On} = V_{IN} + \Delta V = 10,649\,\text{V}$ erreicht ist. Außerdem schaltet er sich ab, sobald die Eingangsspannung unterhalb von $V_{Off} = V_{IN} = 9,649\,\text{V}$ erreicht worden ist. Die Voraussetzung für das Gelingen dieser Vorgehensweise besteht allerdings darin, dass ein Widerstand, der sich vom Ausgangspin der 3,3 V-Spannung zum Eingangspin „SHDN“ erstreckt, mit einem Wert von

$$\begin{aligned} R_{FB} &= R_{HI} \cdot \frac{V_{OUT}}{\Delta V} \\ &= 110\,\text{k}\Omega \cdot \frac{3,3\,\text{V}}{1\,\text{V}} \\ &= 363\,\text{k}\Omega \end{aligned} \tag{3.16}$$

gesetzt wird. Die Realisierung dieses Widerstandes erfolgt mittels der Ersetzung durch einen 330 kΩ-Widerstand.
Somit sind die Widerstände $R37 = R_{LO} = 330\,\text{k}\Omega$, $R29 + R31 = R_{HI} = 110\,\text{k}\Omega$ und $R39 = R_{FB} = 330\,\text{k}\Omega$ zu verwenden.

2,5 V-Spannungserzeugung

Auf Grund der Tatsache, dass der Schaltregler LT1376 von Linear Technology nicht genügend Strom bereitstellen kann, ist für die 2,5 V-Spannungsversorgung ein anderer Schaltregler als der oben beschriebene verwendet worden: Bei dieser Sachlage kommt der Schaltregler TPS54350 von Texas Instrument zum Einsatz. Bild 3.12 auf der nächsten Seite stellt die gesamte Beschaltung dar, die benötigt wird, damit der Schaltregler bei einer Eingangsspannung von 12 V eine Ausgangsspannung von 2,5 V mit einem maximalen Strom von 2 A erzeugt.
Die grundsätzliche Beschaltung ist aus dem Datenblatt [5, Seite 1] übernommen worden, wobei die Widerstände, Spulen und Kondensatoren bzgl. ihrer Größe an die hier vorliegenden Anforderungen angepasst werden müssen. Damit der Schaltregler eine Spannung von 2,5 V am Ausgang erzeugen kann, ist der Ausgang über einen Spannungsteiler mit dem VSENSE-Eingang zu verbinden. Die Referenzspannung, die am VSENSE-Eingang anliegen soll, beträgt 0,891 V. Dieser Spannungsteiler lässt sich durch folgendes Verhält-

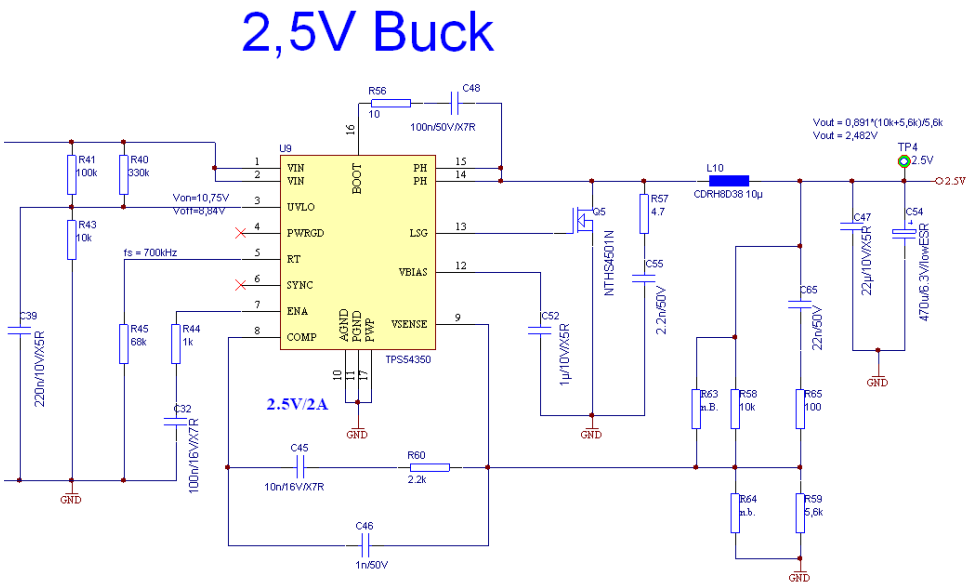

Bild 3.12: Erzeugung der 2,5 V-Spannung

nis beschreiben, wobei die Formel anschließend nach $R59$ auflöst wird:

$$\begin{aligned} \frac{0,891\,\mathrm{V}}{V_{OUT}} &= \frac{R59}{R59+R58} \\ R59+R58 &= R59 \cdot \frac{V_{OUT}}{0,891\,\mathrm{V}} \\ R59 \cdot \left(\frac{V_{OUT}}{0,891\,\mathrm{V}} - 1\right) &= R58 \\ R59 &= \frac{R58}{\frac{V_{OUT}}{0,891\,\mathrm{V}} - 1} \end{aligned} \tag{3.17}$$

Mit $R58 = 10\,\mathrm{k\Omega}$ gewählt und $V_{OUT} = 2,5\,\mathrm{V}$ erhält man einen Widerstandswert von

$$\begin{aligned} R59 &= \frac{10\,\mathrm{k\Omega}}{\frac{2,5\,\mathrm{V}}{0,891\,\mathrm{V}} - 1} \\ &= 5,54\,\mathrm{k\Omega}. \end{aligned} \tag{3.18}$$

Schließlich lässt sich V_{OUT} mit $R59 = 5,6\,\mathrm{k\Omega}$ ausrechnen und es ergibt sich dafür folgender Wert:

$$\begin{aligned} V_{OUT} &= 0,891\,\mathrm{V} \cdot \frac{R59+R58}{R59} \\ &= 0,891\,\mathrm{V} \cdot \frac{5,6\,\mathrm{k\Omega}+10\,\mathrm{k\Omega}}{5,6\,\mathrm{k\Omega}} \\ &= 2,48\,\mathrm{V} \end{aligned} \tag{3.19}$$

Zur Berechnung des Filters, das am COMP-Eingang anzuschließen ist, wird wie im Datenblatt [5, Seite 17] angegeben der „SWIFT Designer“ verwendet. Dieser kann auf der Homepage von Texas Instruments[5] kostenlos heruntergeladen werden. In Bild 3.13 ist die Berechnung des Filters und deren Einstellung abgebildet.
Hinsichtlich der Tatsache, dass nicht alle Kondensatoren und Widerstände mit den ex-

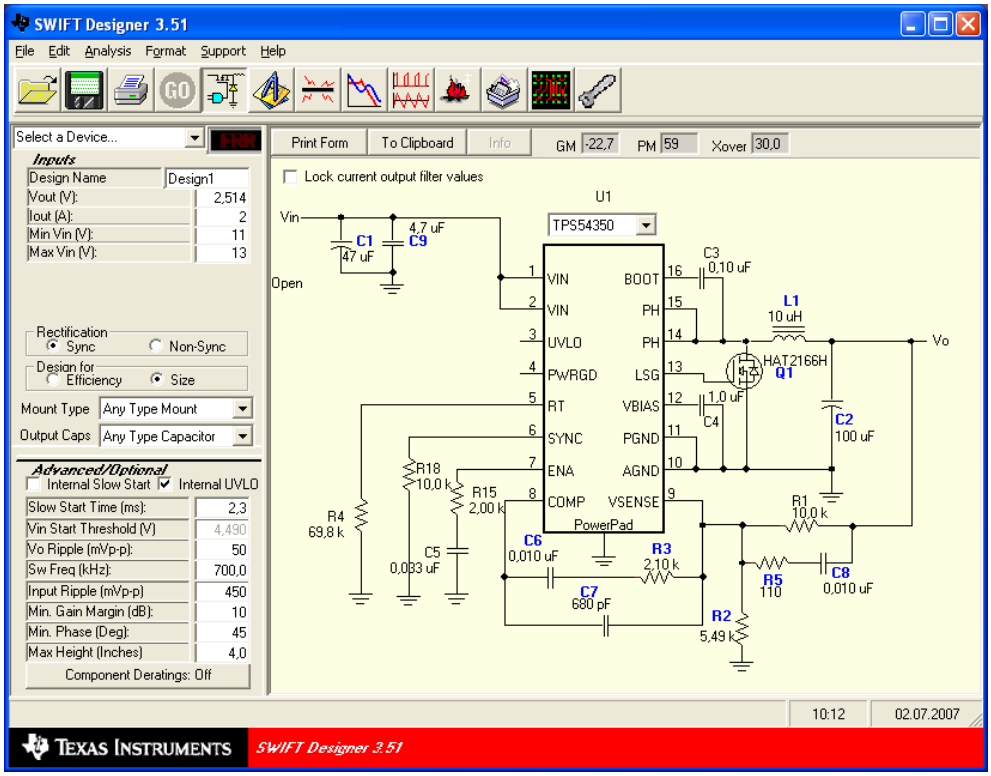

Bild 3.13: Berechnung des Filters für den COMP-Eingang des Schaltreglers

akten Werten zur Verfügung stehen, sind Kondensatoren und Widerstände verwendet worden, die so nahe wie möglich an den berechneten Werten liegen.
Damit der Schaltregler erst mit seiner Umsetzung beginnt, wenn eine bestimmte Eingangsspannung erreicht worden ist, muss der UVLO-Eingang mit einem Spannungsteiler beschaltet werden. Die Formeln, die V_{On} und V_{Off} bestimmen, werden im Datenblatt [5, Seite 7] wie folgt angeben:

$$(R40||R41) = \frac{V_{On} \cdot R43}{1,24\,\mathrm{V}} - R43 \tag{3.20}$$

$$V_{Off} = \frac{((R40||R41) + R43) \cdot 1,02\,\mathrm{V}}{R43} \tag{3.21}$$

[5] www.ti.com

Um zu erreichen, dass der Schaltregler erst bei einer Eingangsspannung V_{On} von 10,5 V mit der Umsetzung beginnt, wird $R43 = 10\,\text{k}\Omega$ gewählt und der Parallelwiderstand aus $R40$ und $R41$ mit Formel 3.20 folgendermaßen berechnet:

$$\begin{aligned}(R40||R41) &= \frac{10,5\,\text{V} \cdot 10\,\text{k}\Omega}{1,24\,\text{V}} - 10\,\text{k}\Omega \\ &= 74,68\,\text{k}\Omega \end{aligned} \tag{3.22}$$

Mit $R41 = 100\,\text{k}\Omega$ gewählt erhält man für $R40$

$$\begin{aligned}R40 &= \frac{1}{\frac{1}{(R40||R41)} - \frac{1}{R41}} \\ &= \frac{1}{\frac{1}{74,68\,\text{k}\Omega} - \frac{1}{100\,\text{k}\Omega}} \\ &= 329,0\,\text{k}\Omega. \end{aligned} \tag{3.23}$$

Der Widerstand, der am nahesten am theoretischen Wert liegt, beträgt $330\,\text{k}\Omega$ und wird hier daher verwendet. Mit den oben genannten Widerständen lässt sich nun die Einschaltspannung V_{On} und die Abschaltspannung V_{Off} exakt berechnen. Stellt man die Formel 3.20 nach V_{On} um und setzt die zuvor bestimmten Werte in die Formel ein, so ergibt sich für V_{On}

$$\begin{aligned}\frac{V_{On} \cdot R43}{1,24\,\text{V}} &= (R40||R41) + R43 \\ V_{On} \cdot R43 &= ((R40||R41) + R43) \cdot 1,24\,\text{V} \\ V_{On} &= \frac{((R40||R41) + R43) \cdot 1,24\,\text{V}}{R43} \\ V_{On} &= \frac{\left(\left(\frac{R40 \cdot R41}{R40+R41}\right) + R43\right) \cdot 1,24\,\text{V}}{R43} \\ V_{On} &= \frac{\left(\left(\frac{330\,\text{k}\Omega \cdot 100\,\text{k}\Omega}{330\,\text{k}\Omega+100\,\text{k}\Omega}\right) + 10\,\text{k}\Omega\right) \cdot 1,24\,\text{V}}{10\,\text{k}\Omega} \\ V_{On} &= 10,75\,\text{V}. \end{aligned} \tag{3.24}$$

Mit denselben Widerstandswerten in Formel 3.21 eingesetzt erhält man

$$\begin{aligned}V_{Off} &= \frac{\left(\left(\frac{330\,\text{k}\Omega \cdot 100\,\text{k}\Omega}{330\,\text{k}\Omega+100\,\text{k}\Omega}\right) + 10\,\text{k}\Omega\right) \cdot 1,02\,\text{V}}{10\,\text{k}\Omega} \\ &= 8,84\,\text{V}. \end{aligned} \tag{3.25}$$

Zur Verminderung der Störungen des eingespeisten Netzes durch den Schaltregler wird die Schaltfrequenz des Schaltreglers möglichst hoch eingestellt. Dabei lässt sich die Schaltfrequenz mit Hilfe des RT-Eingangs des Schaltreglers bestimmen. Im Datenblatt [5, Seite 15] wird der Widerstand, der an Masse anzuschließen ist, durch folgende Formel bestimmt:

$$R45\,(k\Omega) = \frac{46000}{f_s\,(kHz) - 35,9} \tag{3.26}$$

Damit der Schaltregler mit einer Frequenz von ungefähr 700 kHz arbeiten kann, muss ein Widerstand mit einem Wert von

$$\begin{aligned} R45 &= \frac{46000}{700 - 35,9}\text{k}\Omega \\ &= 69,266\,\text{k}\Omega \end{aligned} \tag{3.27}$$

angeschlossen werden. Für $R45$ wird ein Wert von 68 kΩ verwendet und somit ergibt sich für den Schaltregler eine Schaltfrequenz f_s von

$$\begin{aligned} R45\,(k\Omega) &= \frac{46000}{f_s\,(kHz) - 35,9} \\ f_s\,(kHz) - 35,9 &= \frac{46000}{R45\,(k\Omega)} \\ f_s &= \left(\frac{46000}{R45\,(k\Omega)} + 35,9\right)\text{kHz} \\ &= 712,4\,\text{kHz}. \end{aligned} \tag{3.28}$$

3.4 Digitale Ein- und Ausgänge

Zur späteren Verwendung dieses IO-Boards müssen auf der Platine noch digitale Ein- und Ausgänge untergebracht werden. Da es für diese Platine schon digitale Ausgänge gibt, wird im Folgenden nur auf die Beschaltung der digitalen Eingänge eingegangen. In Bild 3.14 auf der nächsten Seite ist der Schaltungsaufbau der digitalen Eingänge abgebildet.
Bei den digitalen Eingängen kommt der Optokoppler HCPL063L von Agilent zum Einsatz. Der n-Kanal selbstleitende MOS-Feldeffekttransistor mit einem Widerstand von 180 Ω sorgt für eine Stromquelle. Die Gate-Source-Spannung ist im Datenblatt [7, Seite 6] aus dem Graphen mit -1,8 V abzulesen. Somit stellt sich ein Strom von

$$\begin{aligned} I &= -\frac{V_{GS}}{R} \\ &= -\frac{-1,8\,\text{V}}{180\,\Omega} \\ &= 10\,\text{mA} \end{aligned} \tag{3.29}$$

ein, der den Optokoppler-Eingang nicht zerstört.
Zur Sicherstellung einer galvanischen Trennung der digitalen Ein- und Ausgänge muss auf der Platine ein kleiner separater Bereich eingerichtet werden. Bei diesem Bereich ist darauf zu achten, dass die Optokoppler gleichsam als Bücke dienen.

3.5 Inbetriebnahme

Zu Beginn der Inbetriebnahme ist die IO-Platine ohne ein Processing-Board an die 12 V-Spannung angeschlossen worden. Danach sind die einzelnen Spannungen zu überprüfen

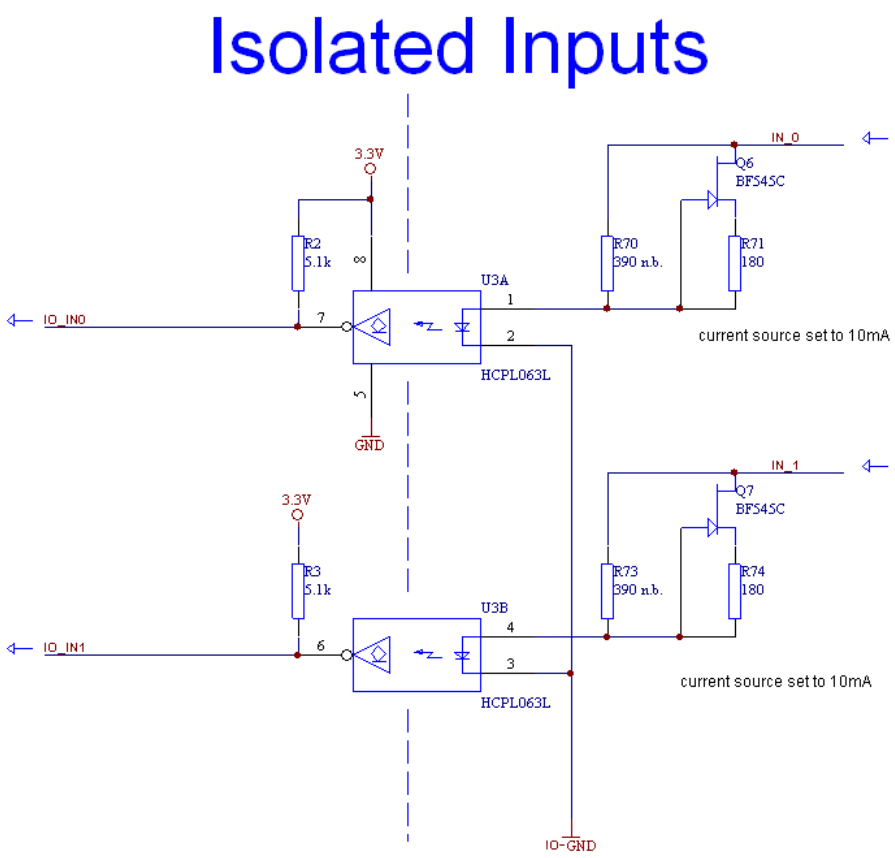

Bild 3.14: Beschaltung des HCPL063L für die digitalen Eingänge

gewesen. Am 12 V-Testpunkt ist eine Spannung von 11,67 V, am 5 V-Testpunkt eine Spannung von 4,987 V und am 3,3 V-Testpunkt eine Spannung von 3,355 V gemessen worden. Diese Spannungen haben sich alle im erwarteten Bereich befunden.
Beim 2,5 V-Testpunkt ist eine Spannungsschwankung von circa 2,26 V bis 2,36 V ermittelt worden. Da der Phy jedoch eine Eingangsspannung von mindestens 2,3 V [18, Seite 57] benötigt, hat sich die Spannung durch das Verändern des Widerstandes $R64 = 68\,\text{k}\Omega$ nach oben korrigiert auf ein Wert von 2,613 V.
Im nächsten Schritt ist das Processing-Board angeschlossen worden. Ein kleines Testprogramm hat die Kommunikation zwischen Phy und FPGA erfolgreich bestätigen können. Danach ist das FPGA mit dem zuvor auf dem Evaluierungs-Board getesteten Programm programmiert worden, um die Datenkommunikation zwischen Phy und FPGA zu testen. Leider hat diese Kommunikation nicht funktioniert.
Zur Erfüllung des SSTL_2 Class 1-Standards benötigt der Eingang des FPGAs - wie schon in Abschnitt 3.1 auf Seite 18 beschrieben - eine Referenzspannung von 1,25 V. Da die Referenzspannung der IO-Bank[6] des FPGAs an dieser Stelle auf Masse gelegt

[6]Mehrere digitale Ein- und Ausgänge werden zu einer sogenannten IO-Bank zusammengefasst.

gewesen ist, hat der SSTL_2-Standard nicht erfüllt werden können. Zur Erfüllung dieses Standards müsste das Processing-Board neu erstellt werden. Leider ist aus zeitlichen Gründen ein Redesign des Processing-Boards in dieser Diplomarbeit nicht mehr möglich. Um das IO-Board dennoch verwenden zu können, wird dieses als Repeater in den Test miteinbezogen. Dabei werden die Daten vom Evaluierungs-Board versendet, empfangen und ausgewertet.

4 Software

Die gesamte Software für die beiden Programme Quartus II und Xilinx ISE ist in VHDL geschrieben worden. Gute Erläuterungen für die Sprache VHDL finden sich in [9]. Zum einfachen Testen des Programms werden die einzelnen Module in so genannten „Komponenten“ zusammengefasst. Für jede Komponente wird außerdem eine sogenannte „Testbench“ geschrieben, um diese vor ihrer Inbetriebnahme zu überprüfen.
Jede Komponente verwendet mindestens die folgenden Bibliotheken:

- IEEE.STD_LOGIC_1164.ALL
- IEEE.STD_LOGIC_ARITH.ALL
- IEEE.STD_LOGIC_UNSIGNED.ALL
- work.Component_Pkg.all

Diese ersten drei Bibliotheken sind standardisiert und werden in [9] näher beschrieben. Die letzte Bibliothek ist projektspezifisch und enthält die Definition aller verwendeten Komponenten. Dadurch entfällt eine explizite Deklaration der Komponente in den einzelnen VHDL-Dateien. Außerdem verwenden einige Komponenten noch selbsterstellte Bibliotheken, um die Übersichtlichkeit des Quellcodes zu steigern.
Sobald ein Zustandsautomat mehrere Zustände besitzt, wird er mit einem selbstdefinierten und aussagekräftigen Namen realisiert. Für Signale im Allgemeinen gibt es lediglich die Konvention, dass Wortteile im Signalnamen mit einem Großbuchstaben beginnen müssen. Schnittstellensignale bekommen außerdem ein „o“, wenn es sich um ein Ausgang, ein „i“, wenn es ich um ein Eingang oder ein „b“, wenn es sich um ein bidirektionales Signal handelt, vorangestellt. Werden nur einzelne Bits eines Signals beschrieben oder gelesen, so sind diese bei der Signalbeschreibung aufgeführt.
Ein häufig verwendeter Zustandsautomat wird in Listing 4.1 exemplarisch dargestellt. Dabei handelt es sich um einen Moore-Automaten. Dieser Zustandsautomat lässt den Ausgang `oLED` mit halber Taktgeschwindigkeit blinken, solange das Signal `iReset` ungleich null und das Signal `iLED_Blink` gleich eins ist:

Listing 4.1: Beispiel eines Zusantdsautomaten

```
1 LED_Blink : process (iReset, iClock)
2 begin
3   if iReset = '1' then
4     State <= LED_ON;
5     oLED <= '0';
6
```

```
7     elsif RISING_EDGE(iClock) then
8       oLED <= '0';
9
10      case State is
11        when IDLE =>
12          if iLED_Blink = '1' then
13            State <= LED_ON;
14          else
15
16        when LED_ON =>
17          oLED <= '1';
18          if iLED_Blink = '1' then
19            State <= LED_OFF;
20          else
21            State <= IDLE;
22          end if;
23
24        when LED_OFF =>
25          if iLED_Blink = '1' then
26            State <= LED_OFF;
27          else
28            State <= IDLE;
29          end if;
30      end case;
31    end if;
32  end process LED_Blink;
```

4.1 Top-Level

Das Top-Level-Design bindet alle für die Funktion notwendigen Komponenten ein. In Bild 4.1 auf der nächsten Seite werden sowohl der Aufbau des Top-Level-Designs als auch die Verbindung der einzelnen eingebundenen Komponenten dargestellt. Der für das Ansteuern der **DATA_Control_Inst**-Komponente erforderliche Takt ist der Empfangstakt `iRCA`. Aus diesem Takt sind durch die PLL drei Takte entstanden. Mittels der Takte `Clock_0` und `Clock_90` werden die Daten versendet. Der Takt `Clock_FX` ist für das Empfangen der Daten zuständig. Er kann unabhängig von den anderen beiden Takten verschoben werden, falls es Probleme bei der Datenübername geben sollte.
Der Takt `iClock` wird durch die Taktteiler **ClkDiv_Control**-Komponente in drei Instanzen auf drei unterschiedliche Takte verlangsamt. Diese langsameren Takte sind den Komponenten zur Verfügung gestellt worden.
Die **PHY_Control_Inst**-Komponente steuert den Ablauf des Phys und kommuniziert

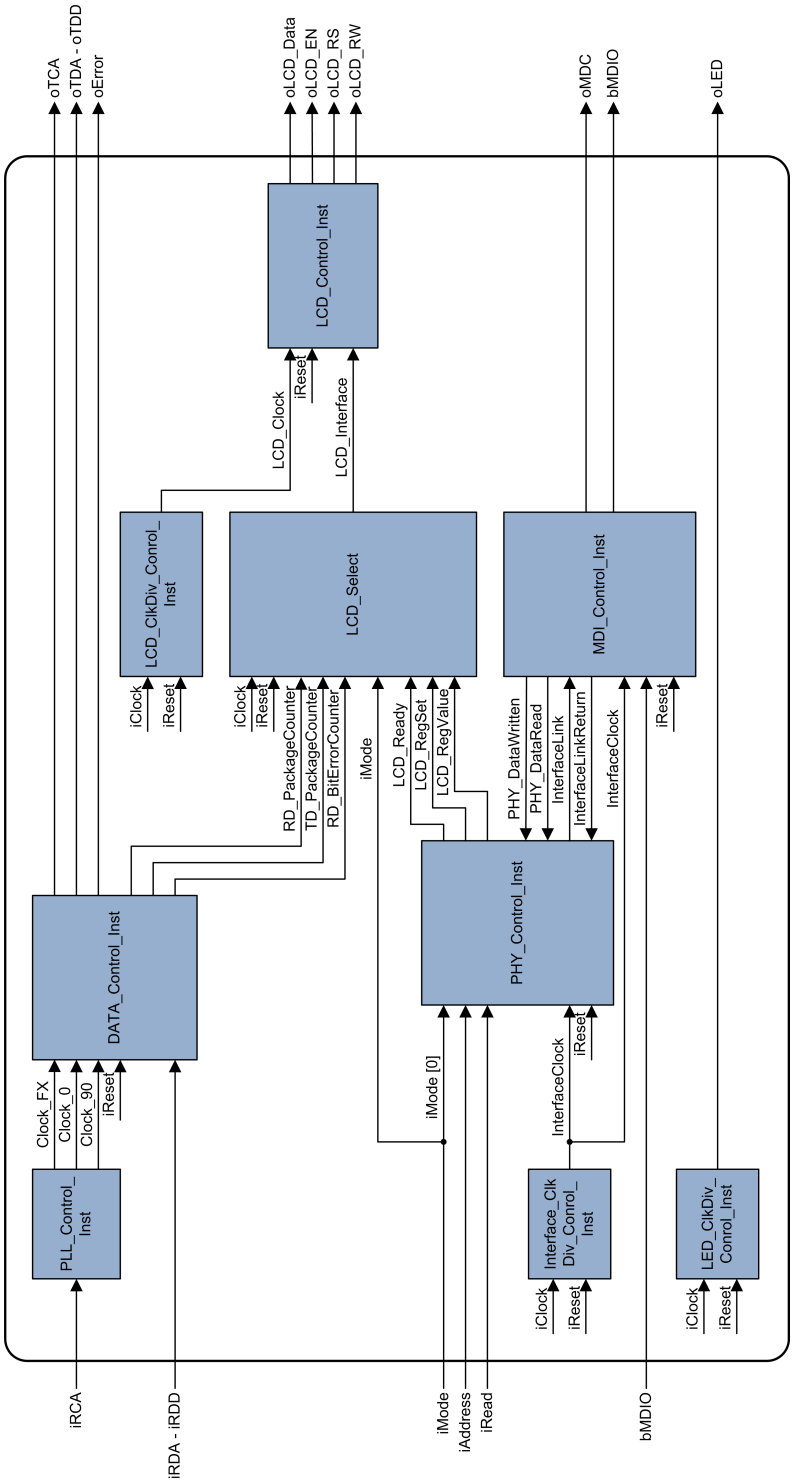

Bild 4.1: Blockschaltbild der Top_Level-Komponente

mit dem Phy über die **MDI_Control_Inst**-Komponente.
Im **LCD_Select**-Prozess wird die Information der **DATA_Control_Inst**- und der **PHY-_Control_Inst**-Komponente empfangen. In diesem Prozess entscheidet sich mittels des Signals `iMode`, welche Werte auf dem LC-Display erscheinen. Die **LCD_Control_Inst**-Komponente übernimmt die Daten des **LCD_Select**-Prozesses und gibt diese auf dem LCD aus.

4.2 Daten-Kontroller

Der Daten-Kontroller ist lediglich für die Zusammenführung der beiden Komponenten, **TD_Control** und **RD_Control**, zuständig. Damit lässt sich das Zusammenspiel der beiden getrennt voneinander entwickelten Kontroller einfach testen. Außerdem wird die Lesbarkeit des Codes im **Top_Level**-Design erhöht.

4.3 TD-Kontroller

Dieser Kontroller lädt die Daten aus dem ROM[1] und übergibt sie an den **DDR_Reg_TD**-Kontroller. Bild 4.2 auf der nächsten Seite stellt den Aufbau dieser Komponente dar. Für die Datenübertragung an den **DDR_Reg_TD**-Kontroller ist der Prozess **TD_SendData** zuständig. Zur Initialisierung des Phys verweilt der Prozess nach dem Einschalten für ungefähr eine halbe Millisekunde im Zustand ***INIT***. Danach wechselt er in den Zustand ***IDLE*** und löscht die beiden Zähler `TD_PackageCounter` und `TD_StateCounter`.
Im Zustand ***IDLE*** werden die Startadressen der einzelnen ROM-Komponenten gesetzt. Zur Entstehung verschiedener Signale auf den einzelnen Lanes beim Senden wählt man in diesem Zustand die Startadressen der ROM-Komponenten unterschiedlich. Es wird zwar einen Überlauf der ROM-Adresssignale geben; dieser kann allerdings durch das Setzen einer geschickt gewählten Größe des ROMs vernachlässigt werden. Dabei muss das ROM eine Größe aufweisen, die sich durch eine Exponentialfunktion zur Basis zwei mit einer natürlichen Zahl auszeichnet (z.B. 256, 512, 1024, etc.). Somit ist auf diese Weise jeder Adresse im ROM eine Zufallszahl zugeordnet.
Um der **DDR_Reg_TD**-Komponente genügend Zeit zur Erstellung des Pakets für den Phy zu verschaffen, wird im ***IDLE***-Zustand zwischen dem Senden zweier Pakete etwas gewartet. Dafür ist der Zähler „`StateCounter`“ zuständig. Erst wenn er einen Wert von vier überschreitet, wechselt der Prozess automatisch in den Zustand ***DATA***.
Im ***DATA***-Zustand werden die Daten aus dem ROM an die **DDR_Reg_TD**-Komponente übergeben, die daraufhin die Daten verschickt. Sobald der Adresszähler für Lane A den Maximalwert erreicht hat, wird überprüft, ob noch weitere Daten zu senden sind. Diese Überprüfung geschieht mit dem Zähler „`TD_PackageCounter`“. Falls weitere Daten gesendet werden sollen, geht der Prozess in den Zustand ***IDLE*** zurück und der Zähler

[1] Read Only Memory

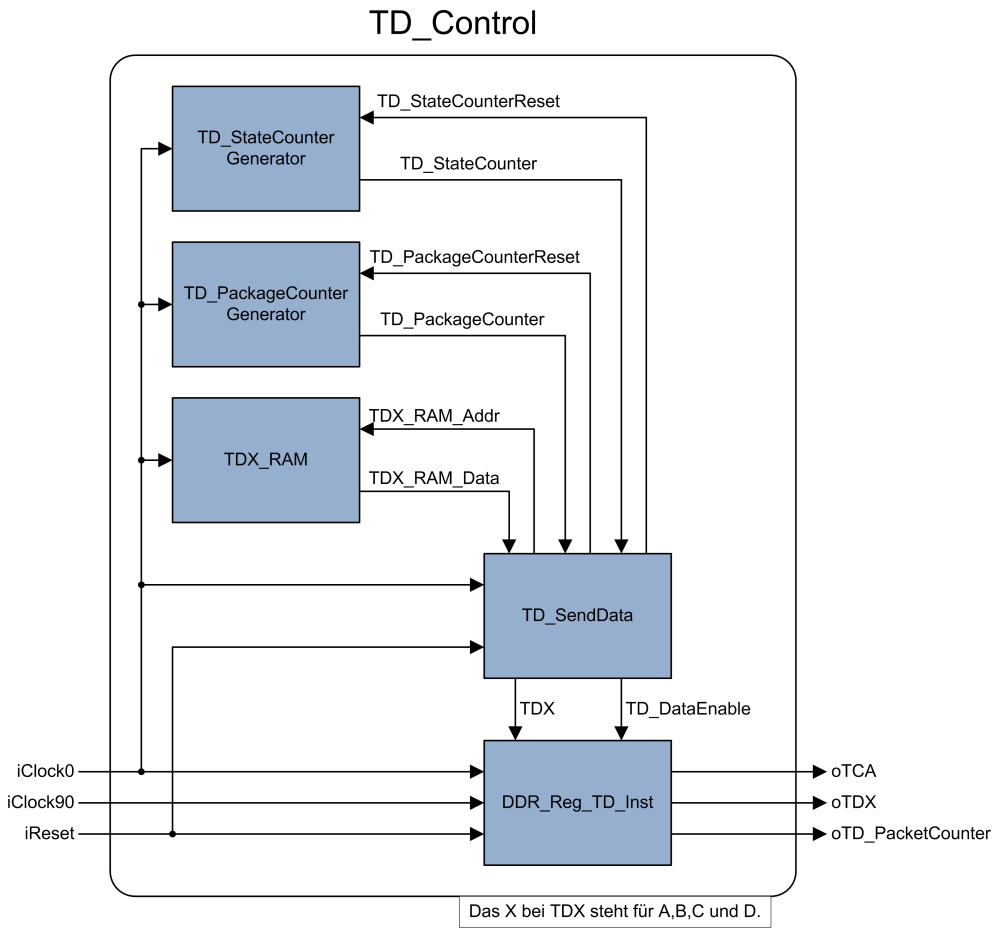

Bild 4.2: Blockschaltbild des TD-Kontrollers

`TD_PackageCounter` wird um Eins erhöht. Ist jedoch keine weitere Datenübertragung erforderlich, so setzt der Prozess in den Zustand ***TRANSMIT_END*** über und verharrt in diesem. Dadurch werden keine weiteren Nutzdaten, sondern nur noch die IDLE-Daten an den Phy übergeben. Gleichzeitig setzt der Zustand ***TRANSMIT_END*** das Signal `TD_SendEnd` auf eins, um zu signalisieren, dass das Senden beendet ist. Nur durch einen Reset-Impuls kann der Prozess wieder von Neuem beginnen und die Daten versenden.
Die Prozesse **TD_StateCounterGenerator** und **TD_PackageCounterGenerator** sind für das Hochzählen und Löschen der `TD_StateCounter`- bzw. `TD_PackageCounter`-Signale zuständig.
Die vier ROM-Komponenten haben die Aufgabe - wie weiter oben schon erwähnt - die zu sendenden Daten aus dem ROM auszulesen. Dabei enthalten alle ROM-Komponenten dieselben Daten.

4.4 DDR_Reg_TD-Kontroller

Der DDR_Reg_TD-Kontroller ist für das Senden der Daten zuständig. Auf Grund der Datenübertragung mit einer Taktgeschwindigkeit von 156,25 MHz bei fallender und steigender Flanke ist es nötig die **D**ouble**D**ata**R**ate[2]-Register in den IO-Zellen des FPGAs zu verwenden. Infolgedessen ist es erforderlich, bei jeder steigenden Flanke zwei Bytes an die DDR-Register der IO-Zellen zu übergeben. In Bild 4.3 wird der Aufbau des DDR_Reg_TD-Kontrollers dargestellt.
Hierbei ist der Prozess **SendData** von besonderer Bedeutung. Er verpackt die Daten

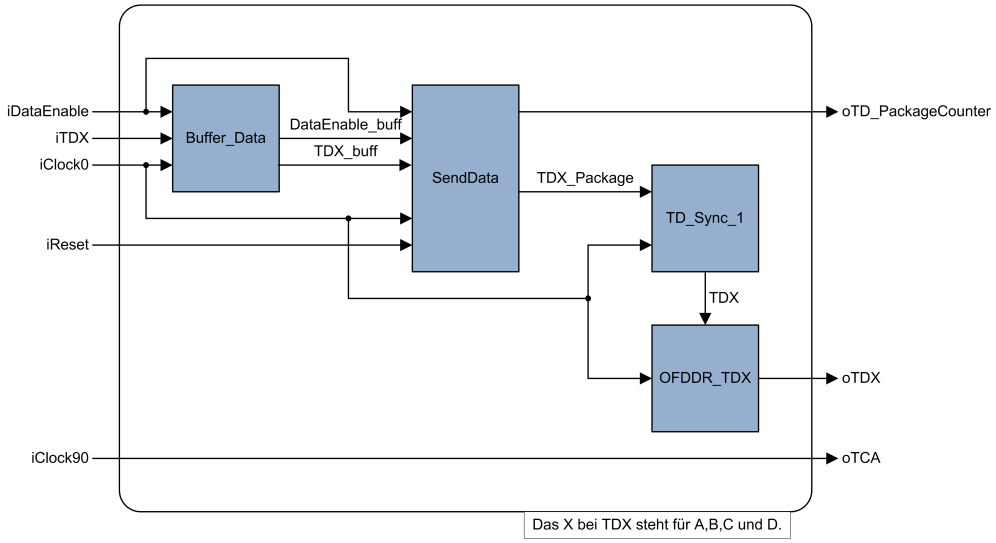

Bild 4.3: Blockschaltbild des DDR_Reg_TD-Kontrollers

in ein für den Phy verständliches Paket, damit er diese versenden kann. Dabei geht der

[2]doppelte Taktrate

Prozess **SendData** wie folgt vor: Nach einem Reset wechselt er in den Zustand ***IDLE***. In diesem Zustand verharrt der Prozess solange, bis das Signal `iDataEnable` logisch null ist und übergibt die IDLE-Kommandos an den Phy. Sobald das Signal `iDataEnable` logisch eins wird, geht der Prozess in den Zustand ***SOF*** über.
Im Zustand ***SOF*** erhält ausschließlich Lane A ein Start-Kommando, während alle anderen Lanes das IDLE-Kommando erhalten. Außerdem ist es erforderlich, in diesem Zustand bei allen Lanes die Checksumme zu löschen. Beim darauf folgenden Takt wechselt der Prozess automatisch in den Zustand ***DATA***.
Zur Ermöglichung einer einfachen Gestaltung der Ansteuerung dieser Komponente sind die zu sendenden Daten immer nur solange gültig, wie eine logische Eins beim Signal `iDataEnable` vorliegt. Da allerdings noch vor der Datenübertragung das Start-Kommando zu versenden ist, werden die Daten und das Signal `iDataEnable` zwischengespeichert; d.h. im Zustand ***DATA*** empfängt der Phy immer die zwischengespeicherten Daten. Außerdem wird in diesem Zustand die Checksumme über die Daten berechnet. Eine besonders einfache Berechnungsmethode - die im Übrigen hier angewendet worden ist - besteht in der XOR-Verknüpfung der zu sendenden zwei Bytes mit der vorherigen Checksumme. Sobald das zwischengespeicherte Signal `iDataEnable` den Wert Null annimmt, wechselt der Prozess in den Zustand ***CRC***.
In diesem Zustand wird die berechnete Checksumme an den Phy gesendet und der Prozess geht in den „***EOF***“-Zustand über.
An dieser Stelle sendet das FPGA nur auf der Lane A das Stop-Kommando an den Phy. Alle anderen Lanes erhalten das IDLE-Kommando. Zur Ermittlung der Anzahl der gesendeten Pakete wird der Zähler `TD_PackageCounter` um eins erhöht und daraufhin wechselt der Prozess in den Zustand ***IDLE***, um empfangsbereit für neue Packete zu sein.
Der Prozess **Buffer_Data** ist - wie schon oben angedeutet - für das Puffern der eingehenden Daten zuständig.
Ein weiteres Mal werden die zu sendenden Daten mit Hilfe des **TD_Sync_1**-Prozesses gepuffert, bevor diese zum DDR-Register der IO-Zelle gelangen. Auf diese Weise kann der Fitter die vorgegebenen Zeiten einhalten.

4.5 RD-Kontroller

Dieser RD-Kontroller ist für die Einbindung des **DDR_Reg_RD_Control**-Kontrollers und für die Überprüfung der empfangenen Daten zuständig. Die erhaltenen Daten müssen in doppelter Weise auf ihre Richtigkeit überprüft werden. Zuerst ist die Checksumme zu kontrollieren. Bei diesem Kontrollvorgang wird - wie in Abschnitt 4.4 von der letzten Seite beschrieben - am Ende der Daten immer die errechnete Checksumme angefügt. Dabei ist es nötig, die Checksumme aus den erhaltenen Daten zu berechnen und mit der empfangenen Checksumme zu vergleichen. Stimmt die empfangene mit der berechneten Checksumme überein, so hat eine fehlerfreie Übertragung dieses Packets stattgefunden. Die zweite Überprüfung untersucht die Daten auf eventuelle Bitfehler. Hierbei werden die erhaltenen Daten mit denen aus dem ROM verglichen. Dieser Vergleich erfolgt mittels

einer XOR-Verknüpfung der beiden Bytes. Ist das Ergebnis der Verknüpfung gleich null, so bestehen keine Bitfehler bei diesem empfangenen Byte. In Bild 4.4 wird der Aufbau einer solchen Komponente abgebildet.
Die Komponente **DDR_Reg_RD** empfängt und sortiert die Daten. Der Prozess **Recei-**

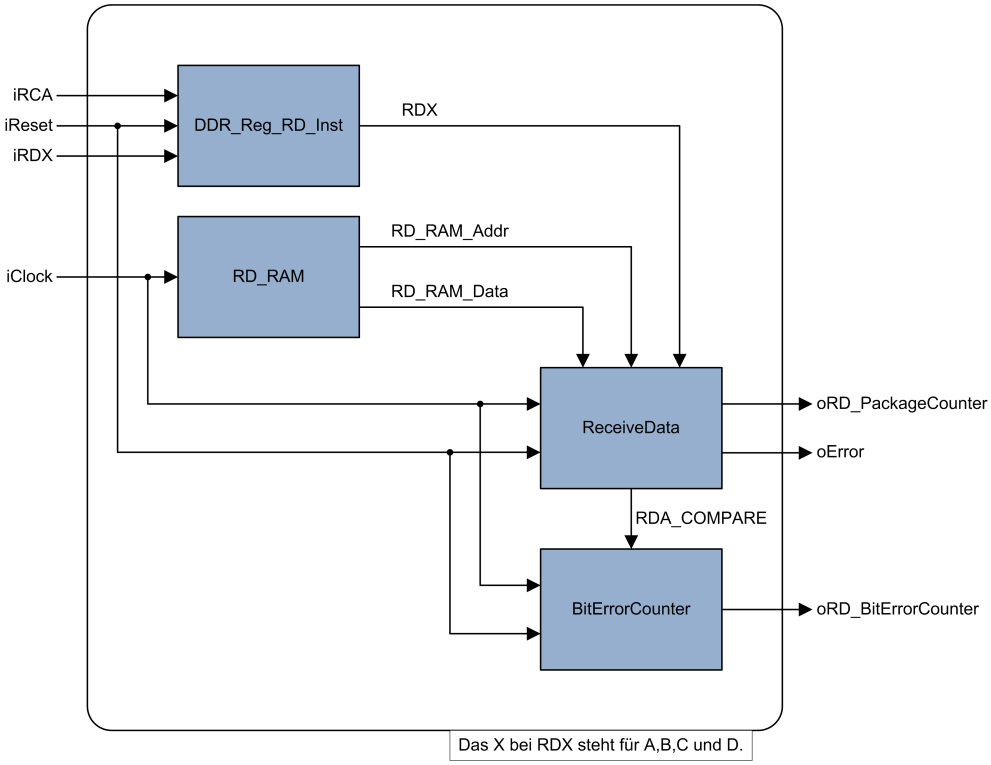

Bild 4.4: Blockschaltbild des RD-Kontrollers

veData wartet nach einem Reset so lange im Zustand ***IDLE*** und löscht die Checksummen-Signale, bis das Signal `RDA(8 downto 0)` den Wert des Start-Kommandos erkennt. Erst dann wechselt der Prozess in den Zustand ***DATA***.
In diesem Zustand berechnet sich die Checksumme der empfangenen Daten analog zur Checksumme des Sendevorgangs. Außerdem werden - wie schon oben beschrieben - die zwei erhaltenen Bytes mit den beiden Bytes aus dem ROM per XOR-Verknüpfung verbunden und das Ergebnis wird in dem Signal `RDA_COMPARE` gespeichert. Da die Anzahl der zu empfangenen Bytes durch die Rom-Größe festgelegt ist, verwendet man zum Erkennen des Zustandswechsels des Prozesses einen Zähler, der die gesamten ROM-Adressen durchläuft. Erst beim Erreichen der maximalen ROM-Adresse wechselt der Prozess in den nächsten Zustand.
Die letzten empfangenen Bytes werden im Zustand ***CALC_CRC*** in die Berechnung der Checksumme eingefügt, um im Zustand ***COMPARE_CRC*** die berechnete Checksumme aus den erhaltenen Daten mit der empfangenen Checksumme zu vergleichen. Bei Übereinstimmung wird der Zähler `RD_PackageCounter` um eins erhöht und das

Ausgangssignal `oError` auf Null gesetzt. Falls die beiden Checksummen jedoch nicht übereinstimmen, ist das Ausgangssignal `oError` auf Eins zu setzen und der Zähler `RD_PackageCounter` darf nicht erhöht werden. Nach diesem Checksummenvergleich wechselt der Prozess automatisch in den Zustand ***IDLE*** und wartet auf ein neues Startkommando.
Wie schon zuvor ausgeführt werden die empfangenen Bytes mit den gesendeten durch eine XOR-Verknüpfung miteinander verbunden und in dem Signal `RDA_COMPARE` gespeichert. Dieses Signal wird vom Prozess **BitErrorCounter** weiterverarbeitet und dabei jede Eins gezählt. Da die Berechnung mit einer Taktrate von 156,25 MHz stattfinden muss, sind die einzelnen Addierer verschachtelt. Zuerst wird jede Eins (Bitfehler) in einem eigenen Zähler erfasst und danach werden zwei nebeneinanderliegende Bits in einem neuen Zähler zusammengefügt. So entsteht aus den beiden Zählern `RD_BitErrorCounter_15` und `RD_BitErrorCounter_14` der zusammengefasste Zähler `RD_BitErrorCounter_15_14`. Diese Art der Verschachtelung muss fünfmal hintereinander durchgeführt werden, um den Gesamtzähler zu erhalten. Aus diesem Grund ist eine Wartezeit nach der XOR-Verknüpfung von mindestens fünf Takten pro Bitfehler erforderlich, bis der Bitfehler gezählt worden ist.

4.6 DDR_Reg_RD-Kontroller

Der DDR_Reg_RD-Kontroller ist für das Empfangen der Daten zuständig. Diese werden mit der doppelten Taktgeschwindigkeit vom `iRCA` empfangen. Zur internen Weiterverarbeitung der Daten ist es erforderlich, diese durch die DDR-Register der IO-Zellen des FPGAs auf die doppelte Datenbreite bei gleichem Takt zu bringen. Somit sind die Daten zur steigenden Flanke vom `iRCA` synchronisiert. Dies geschieht im Prozess **IFDDR_RDX**. Da das Empfangen eines Pakets nur durch das Datenwort *0x1FB* und nicht an Hand der Flanke vom `iRCA` ablesbar ist, müssen die Daten sortiert werden. Bild 4.5 auf der nächsten Seite stellt das Empfangen und Sortieren der Daten dar. Wie oben erwähnt ermöglicht der Prozess **IFDDR_RDX** ein Beförderung der Daten auf die doppelte Datenbreite bei gleichen Takt. Im Folgenden werden die Signale, die der Prozess **IFDDR_RDX** bei steigender Flanke übergibt, immer als „Low-Daten“ und die Signale, die der Prozess bei fallender Flanke abliefert, als „High-Daten“ bezeichnet. Die Prozesse **RD_Sync_0** bis **RD_Sync_2** speichern die empfangenen Daten bei jeder steigenden Flanke und geben diese bei der nächsten steigenden Flanke weiter. Damit ist es möglich, eine zeitliche Verzögerung der Daten zu den Eingangsdaten zu erhalten. In dieser Zeitspanne lässt sich der Ort des Startkommandos auffinden, um den Anfang des Packets zu ermitteln.
Hinter dem Prozess **RD_Sync_0** sind die Daten zur steigenden Flanke synchron. Gleich im Anschluss wird in dem Prozess **StartLowHigh** die Stelle des Startkommandos des ankommenden Packets ermittelt. Durch die beiden Signale `StartOnHigh` und `StartOnLow` lässt sich erkennen, ob sich das Startkommando im oberen oder unteren Datenbereich befindet. Je nach Lage des Startkommandos erhält entweder `StartOnHigh` oder `StartOnLow` eine logische Eins - dabei gilt allerdings, dass niemals beide Signale gleichzeitig

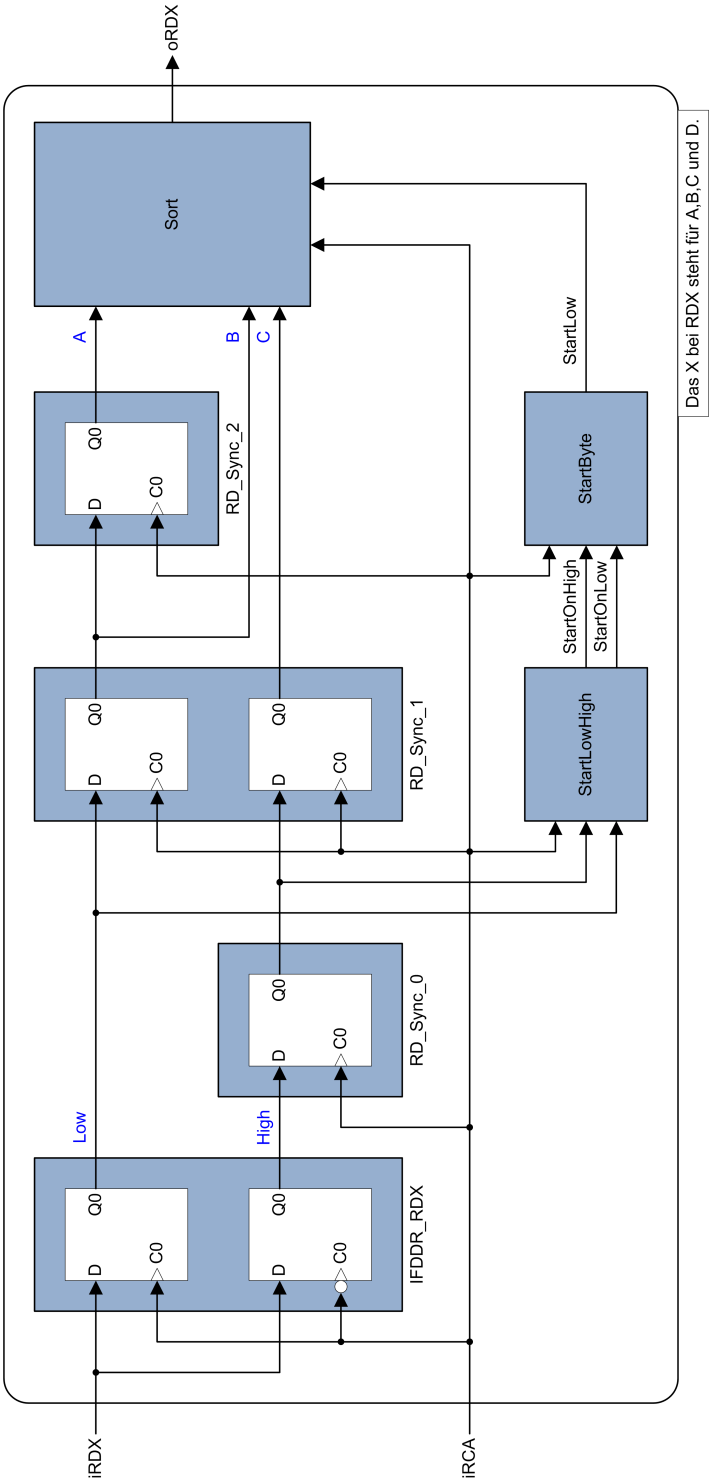

Bild 4.5: Blockschaltbild des DDR_Reg_RD-Kontrollers

eine logische Eins aufweisen können.
Zur Sortierung und Ausgabe der Daten im Prozess **Sort** werden im Prozess **StartByte** die beiden einzelnen Signalen zu einem Signal zusammengeführt.
Der Prozess **Sort** sortiert die Daten so, dass sich im Low-Bereich von `oRDX` das Start-Kommando befindet und die Nutzdaten im darauf folgenden High-Bereich beginnen. Ist `StartLow` logisch eins, so werden die Daten aus Bild 4.5 von der letzten Seite von C und B und ansonsten die Daten von A und B an `oRDX` übergeben.

4.7 ROM-Kontroller

Dieser Kontroller stellt die Daten bereit, die gesendet und mit den empfangenen Daten verglichen werden sollen. Im Listing 4.2 ist die Komponente abgebildet.

Listing 4.2: ROM_Control-Komponente

```
1  architecture Behavior of ROM_Control is
2  begin
3
4    process (iClock)
5    begin
6      if (RISING_EDGE(iClock)) then
7        oData <= RAM(conv_integer(iAddr));
8      end if;
9    end process;
10
11 end Behavior;
```

Für das ROM wird derselbe Takt wie beim Senden bzw. Empfangen der Daten verwendet. Diese ROM-Komponente erhält die Daten aus dem **ROM_Data_Pkg**-Packet, das zuvor mit dem Tool „Create ROM“ erstellt worden ist. Funktion sowie Quellcode dieses Programms sind in Anhang B ab Seite 84 beschrieben.

4.8 PHY-Kontroller

Der Phy-Kontroller ist für die Initialisierung und das Auslesen der Register vom Phy zuständig. Zur Versendung der Anforderungen an den Phy benötigt diese Komponente den MDI-Kontroller. In Bild 4.6 auf der nächsten Seite wird der Funktionsablauf des Prozesses **StateMachine_Control** dargestellt.
Nach dem Einschalten ist eine Initialisierung des Phys erforderlich. Dazu wird der Registerwert *0x7902* in das Register *16* eingetragen, damit die im Phy internen Kompensationszustandsautomaten aktiviert werden können. Mittels einer Änderung des nächsten

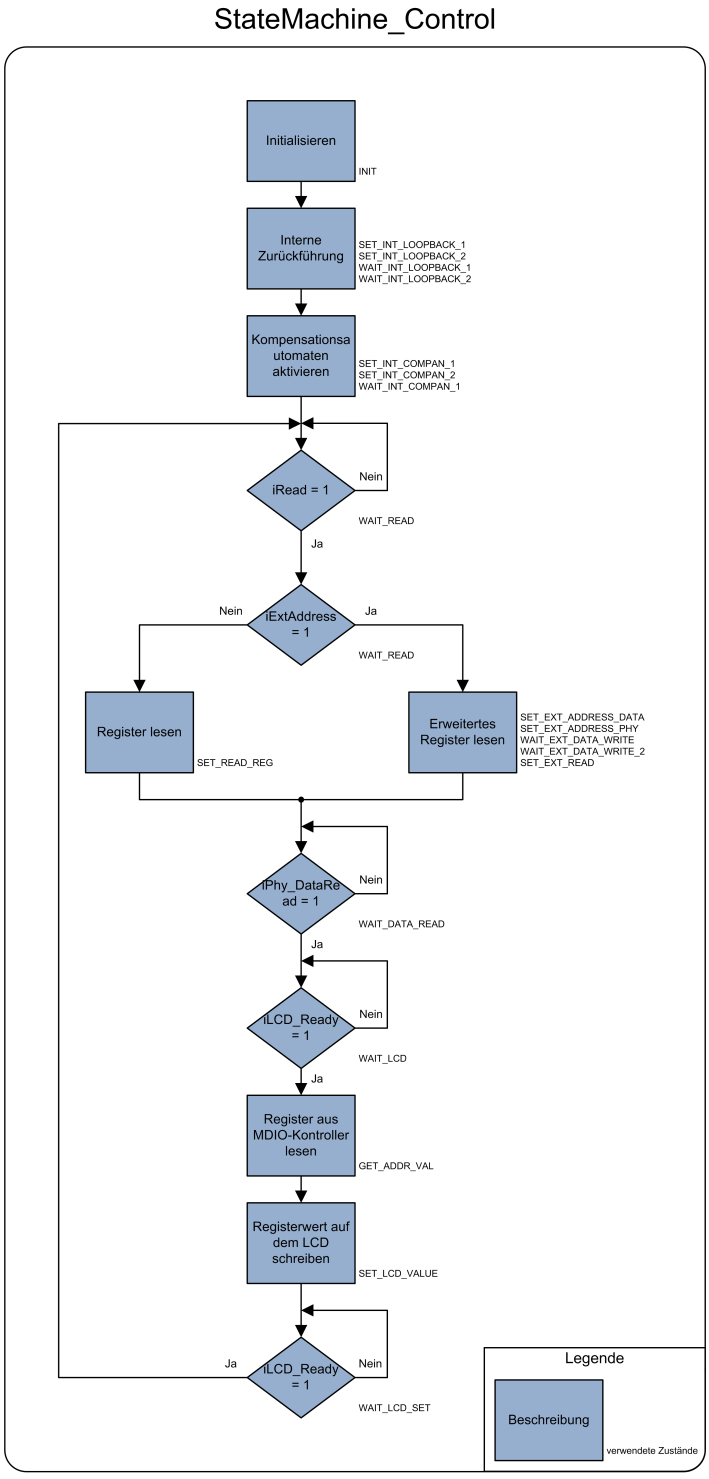

Bild 4.6: Flussdiagramm des Hauptprozesses beim Phy-Kontroller

Zustands in den Zustand ***INIT*** lässt sich eine Aktvierung der internen Zurückführung (Loopback) des Phys bei der Initialisierung durchführen.
Zur Notation eines Wertes in einem Register des Phys sind mehrere Zustände nötig. Zuerst ist es erforderlich, den Registerwert an den MDI-Kontroller zu übergeben. Danach muss der MDI-Kontroller die Adresse erhalten, in die der Registerwert eingetragen werden soll. Nun wartet der Prozess solange ab, bis das Signal `iPhy_DataWritten` logisch eins ist. Nach Eintragung des Registerwerts muss der Prozess noch einige Takte abwarten, um zu schnell hintereinander stattfindende Zugriffe auf den Phy zu verhindern.
Nach der Initialisierung wartet der Prozess solange im Zustand ***WAIT_READ***, bis `iRead` den logischen Wert „Eins“ annimmt. Erst dann wird ausgewertet, ob ein erweitertes oder ein herkömmliches Register ausgelesen werden soll.
Beim Auslesen eines einfachen Registers ist lediglich das Senden der auszulesenden Registeradresse an den MDI-Kontroller erforderlich. Falls dagegen ein erweitertes Register vorliegt, muss zuerst die erweiterte Registeradresse an den MDI-Kontroller gesendet werden, damit dieser die Adresse im Phy speichern kann. Erst danach kann der eigentliche Ausleseprozess des Registers im Phy geschehen.
Im nächsten Schritt wartet der Prozess zur Ausgabe des Registerwertes auf dem Display auf die Verfügbarkeit des LCD-Kontrollers und des LC-Displays. Sobald das LCD für eine Ausgabe bereit ist, finden Auslesung aus dem MDI-Kontroller und eine Übergabe des Werts an den LCD-Kontroller statt. Zum Schluss wird ein weiteres Mal abgewartet, bis der Wert auf dem LCD erschienen ist, bevor der Prozess in den Zustand ***WAIT_READ*** übergeht.

4.9 MDI-Kontroller

Die MDI-Schnittstelle[3] ist für die Kommunikation zwischen Phy und FPGA zuständig. Die Eingangsvektoren dieser Komponente sowie die interne Verarbeitung der einzelnen Prozesse sind in Bild 4.7 auf der nächsten Seite abgebildet.
Auf Grund der Bidirektionalität des Signals `bMDIO` ist dieses sowohl auf der Eingangs- als auch auf der Ausgangsseite sichtbar. Im VHDL-Code stellt es ein Signal dar und wird zum Schreiben und Lesen verwendet. Zum Lesen muss es in den Tri-State-Zustand gebracht werden, um denn einzulesenden Wert nicht zu verfälschen.
Die **generic**-Anweisung taucht nicht im Blockschaltbild auf. Diese Anweisung dient der Zuweisung der Kommandoadressen, die für eine Anfrage benötigt werden. In Tabelle 4.1 auf der nächsten Seite sind die drei Kommandoadressen und deren Zuweisung dargestellt. Die Bedeutung dieser Kommandoadressen wird noch etwas weiter unten erläutert.
Für die Datenannahme und -ausgabe des MDI-Kontrollers ist der Prozess **Request_Interface** zuständig. Dieser Prozess erhält von außen ein Signal vom Typ `InterfaceLinkType` und gibt ein Signal von der Art `InterfaceLinkReturnType` nach außen weiter. Beide Signal-Typen werden im Packet **DataInterface_Pkg** beschrieben. Listing 4.3 auf Seite 51 zeigt den Aufbau dieses Packets:

[3] vgl. Abschnitt 3.1 auf Seite 18

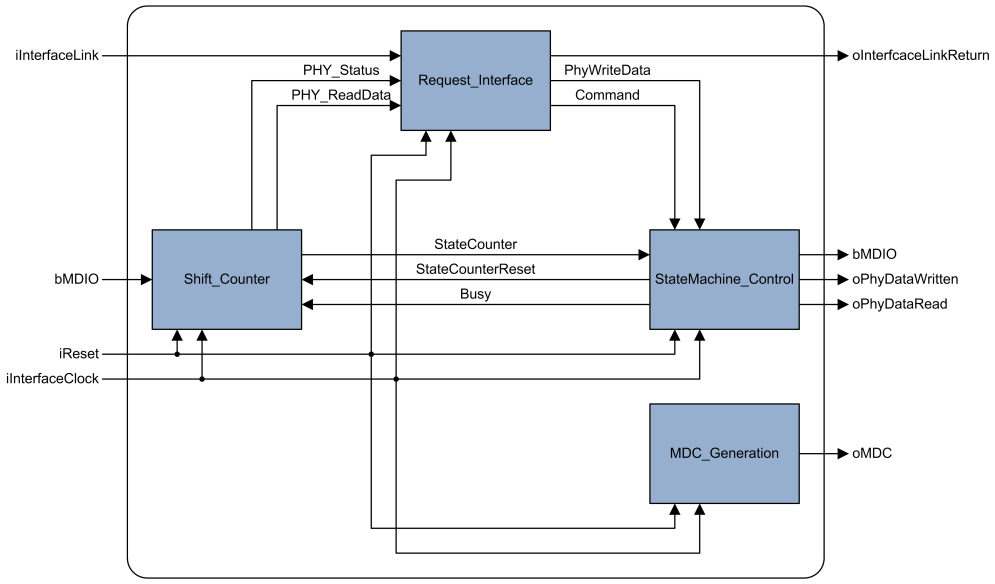

Bild 4.7: Blockschaltbild des MDI-Kontrollers

Kommandoadresse	Wert
ADDR_PHY_COMMAND	40
ADDR_PHY_DATA	41
ADDR_PHY_STATUS	42

Tabelle 4.1: Kommandoadressen und deren Zuweisungen

Listing 4.3: Schnittstelle für den MDI-Kontroller

```
1  package MDI_DataInterface_Pkg is
2
3    type MDI_InterfaceLinkType is
4    record
5      Address   : std_logic_vector(13 downto 0);
6      ReadEna   : std_logic;
7      WriteEna  : std_logic;
8      WriteData : std_logic_vector(31 downto 0);
9    end record;
10
11   type MDI_InterfaceLinkReturnType is
12   record
13     ReadData : std_logic_vector(31 downto 0);
14   end record;
15
16 end MDI_DataInterface_Pkg;
```

Diejenigen Kommandoadressen, die die Information bzgl. des weiteren Umgangs mit den Daten, die in das Signal `WriteData` eingetragen worden sind, enthalten, werden in das Signal `Address` hineingeschrieben. Bei der Übergabe der Adresse von `ADDR_PHY_COMMAND` ist es erforderlich, die Daten aus dem Signal `iInterfaceLink.WriteData` gemäß Tabelle 4.2 aufzuteilen. Wird die Adresse von `ADDR_PHY_DATA` übergeben, so müssen die Daten

`iInterfaceLink.Address`-Bits	Verwendung (internes Signal)
0 bis 4	PHY_RegAddress
8	ReadCommand
9	WriteCommand
10	ExtReadCommand
11	ExtWriteCommand
12	ExtAddressCommand

Tabelle 4.2: Aufteilung des Signals `iInterfaceLink.Address`

in das interne Signal `PHY_WriteData` gespeichert werden.
Ist die Adresse von `ADDR_PHY_STATUS` übergeben worden, so ist es erforderlich, den Status des MDI-Kontrollers in das Signal `oInterfaceLinkReturn.ReadData` abzulegen, so dass es sich im nächsten Schritt ausgelesen lässt.
Das Signal `WriteEna` kündigt einen Schreibbefehl und das Signal `ReadEna` einen Lesebefehl an. Der Signaltyp `InterfaceLinkReturnType` besitzt nur das Signal `ReadData`. Dieses beinhaltet entweder die vom Phy gelesenen Daten oder den Status des MDI-Kontrollers.

Beispielsweise sind zur Eintragung des Wertes *0x6140* in das Register *0* des Phys zwei nacheinander auszuführende Befehle nötig. Zuerst muss der zu schreibende Wert - in diesem Fall *0x6140* - in das Signal `iInterfaceLink.WriteData` abgelegt werden. Zusätzlich ist es erforderlich, in das Signal `iInterfaceLink.Address` die Adresse von `ADDR_PHY_DATA` zu schreiben und das Signal `iInterface.WriteEna` auf „Eins“ zu setzen. Bei der nächsten steigenden Flanke von `iInterfaceClock` wird der Wert des Signals `iInterfaceLink.Address` durch den Prozess **Request_Interface** intern gespeichert. Hiernach muss der eigentliche Schreibbefehl ausgeführt werden. Dafür wird die Kommandoadresse von `ADDR_PHY_COMMAND` in das Signal `iInterfaceLink.Address` abgelegt und die Adresse des Registers - in diesem Fall *0* - an das Signal `iInterfaceLink.WriteData(4 downto 0)` übergeben. Zum Schluss muss das Signal `iInterfaceLink.Address(9)` mit einer Eins beschrieben werden, damit der Prozess **Request_Interface** darüber informiert ist, dass es sich um einen einfachen Schreibbefehl handelt. Bei der nächsten steigenden Flanke von `iInterfaceClock` werden diese Daten übernommen und durch den Prozess **StateMachine_Control** an den Phy übergeben. Vor dem Lesen bzw. Schreiben neuer Daten muss das Signal `oPhy_DataWritten` eine Eins aufweisen. Erst dann ist das Register als beschrieben gekennzeichnet.
Beim Lesevorgang muss darauf geachtet werden, dass das Signal `iInterfaceLink.Address` den Wert der Kommandoadresse `ADDR_PHY_COMMAND` enthält. Außerdem ist die Adresse des zu lesenden Registers in das Signal `iInterfaceLink.WriteData(4 downto 0)` abzuspeichern. Ferner sind Bit 8 des Signals `iInterfaceLink.WriteData` und das Signal `iInterfaceLink.WriteEna` auf Eins zu setzen. Das Ende des Lesens des Registers beim Prozess **StateMachine_Control** wird durch das Signal `oPhy_DataRead` angegeben. Sobald dieses Signal eine Eins aufweist, hat ein erfolgreicher Lesevorgang des Registers stattgefunden. Indem die Kommandoadresse `ADDR_PHY_COMMAND` in das Signal `iInterfaceLink.Address` und eine logische Eins in `iInterfaceLink.ReadEna` geschrieben worden ist, kann nach der nächsten steigenden Flanke des `iInterfaceClock`s der Wert des Registers per Auslesen von `oInterfaceLinkReturn.ReadData` abgefragt werden.
Der Schreib- bzw. Lesevorgang eines erweiterten Registers verläuft analog zu dem eines einfachen Registers. Allerdings muss zuvor die Adresse des erweiterten Registers in den Phy geschrieben werden, um diesen über die Art des Registers zu informieren.
Der Prozess **StateMachine_Control** erkennt anhand des Kommandos (z.B. `ReadCommand`, `WriteCommand`, etc.), welcher boolesche Wert an das Schnittstellensignal `bMDIO` anzulegen ist. Dementsprechend durchläuft dieser Prozess die einzelnen Zustände und schreibt dabei die Daten seriell auf das Signal `bMDIO`. Zur Umgehung der Situation, pro Bit einen Zustand zu benötigen, dient der Prozess **Shift_Counter**, der bei jeder steigenden Flanke von `iInterfaceClock` das Signal `StateCounter` um eins erhöht. Durch das Hochzählen des `StateCounter`s kann der Prozess **StateMachine_Control** länger in einem Zustand bleiben und die einzelnen Werte nacheinander anlegen. Soll ein Registerinhalt vom Phy gelesen werden, so verweilt der **StateMachine_Control**-Prozess genauso lange wie beim Schreiben eines Registers in jedem Zustand. Das Lesen des Signals `bMDIO` wird allerdings in diesem Fall nicht in dem Zustand ***DATA*** des Prozesses **StateMachine_Control** abgehandelt, sondern im Prozess **Shift_Counter**. Eine derar-

tige Lösung ist erforderlich gewesen, da das simultane Schreiben und Lesen eines Signals in einem kombinatorischen Prozess nicht geleistet werden kann.
Der Prozess **MDC_Generation** dient zur Erstellung des Taktes. Das Signal `ClockEnable` ist ein Hilfssignal, das den Prozess **Shift_Counter** steuert.

4.10 LCD-Kontroller

Dieser Kontroller dient der Ansteuerung des LC-Displays. Mittels der Eingangsvektoren erhält der Kontroller die Werte, die auf dem LCD angezeigt werden sollen. Auf Grund der Tatsache, dass zwar das LC-Display, aber nicht diese Komponente in der Lage ist, alle Buchstaben und einige Sonderzeichen auszugeben, wird das LCD hier lediglich zur Ausgabe von Registerwerten oder Zählerständen verwendet. Aus diesem Grund werden nur vier 16 Bit-große Hexadezimalzahlen auf dem Display angezeigt. Außerdem kann diese Komponente nur beim Evaluierungs-Board zum Einsatz kommen. Im Gegensatz dazu werden beim IO-Boards die Signale nicht mit dem LCD, sondern mit dem Programm Quartus in „SignalTab“ angezeigt. Mit Hilfe von SignalTab ist eine Anzeige interner Signale möglich.
Zur einfacheren Steuerung des LC-Displays und zur Steigerung der Übersichtlichkeit des Quellcodes verwendet das LCD das „LCD_DataInterface_Pkg“, in dem die Schnittstellensignale sowohl zu einem Eingangs- als auch zu einem Ausgangssignal zusammengefasst worden sind.
In Listing 4.4 wird das Schnittstellenpacket für den LCD-Kontroller abgebildet:

Listing 4.4: Schnittstelle für den LCD-Kontroller

```
1  package LCD_DataInterface_Pkg is
2
3    type LCD_InterfaceType is
4    record
5      Set     : std_logic;
6      ValueA  : std_logic_vector (15 downto 0);
7      ValueB  : std_logic_vector (15 downto 0);
8      ValueC  : std_logic_vector (15 downto 0);
9      ValueD  : std_logic_vector (15 downto 0);
10   end record;
11
12   type LCD_InterfaceReturnType is
13   record
14     Ready   : std_logic;
15     Data    : std_logic_vector (7 downto 0);
16     RS      : std_logic;
17     RW      : std_logic;
```

```
18      EN      : std_logic;
19    end record;
20
21  end LCD_DataInterface_Pkg;
```

Im `LCD_InterfaceType` befinden sich alle Signale, die für das Schreiben von Werten auf dem Display benötigt werden. Die Daten, die in den Signalen `ValueA`, `ValueB`, `ValueC` und `ValueD` stehen, sind von der **LCD_Control**-Komponente immer nur dann übernommen worden, wenn das Signal `Set` eine logische Eins enthält. Zur Datenübermittlung der **LCD_Control**-Komponente an das LCD müssen alle für die Ansteuerung des LCDs nötigen Signale im `LCD_InterfaceReturnType` enthalten sein. Das einzige Signal, das nicht an das LCD weitergegeben wird, ist das Signal `Ready`. Dieses wird von der **PHY_Control**-Komponente benötigt. Weist `Ready` eine logische Eins auf, so steht die **LCD_Control**-Komponente für eine Datenaufnahme über die `LCD_InterfaceType`-Schnittstelle bereit.
In Bild 4.8 auf der nächsten Seite ist der Ablauf des Hauptprozesses **StateControl** demonstriert. Nach dem Einschalten des Boardes wird zuerst das LCD initialisiert. Dies geschieht in den Zuständen ***SET***, ***ENTRY_MODE***, ***DISPLAY*** und ***CLEAR***.
Im Zustand ***SET*** wird das Ansprechen des LCD auf 8 Bit sowie die Zeilenanzahl und die Zeichengröße bestimmt. Zeilenanzahl und Zeichengröße sind durch den mechanischen Aufbau des LC-Displays festgelegt. Somit ist es erforderlich, die Zeilenanzahl auf zweizeilig und die Zeichengröße auf 5x10 Punkte zu setzen.
Im Zustand ***ENTRY_MODE*** wird das Verhalten des LC-Displays nach dem Schreiben eines Zeichens eingestellt: Das LCD soll nach dem Schreiben eines Zeichens den internen Adresszähler für die Zeichen um Eins erhöhen und die schon auf dem LCD angezeigten Zeichen nicht mehr verschieben.
Der Zustand ***DISPLAY*** schaltet das Display ein. Außerdem stellt er das Blinken des Cursors und den Cursor selbst ab.
Bevor das LCD für das Schreiben von Werten bereit steht, werden im Zustand ***CLEAR*** alle Zeichen auf dem LCD gelöscht, der interne Adresszähler für die Zeichen wird auf Null gesetzt und der Prozess geht in den Zustand ***IDLE*** über.
Der Prozess bleibt in diesem Zustand solange, bis das Schnittstellensignal `Set` eine logische Eins anzeigt, um die Werte der Schnittstellensignale `ValueA`, `ValueB`, `ValueC` und `ValueD` zu übernehmen. Danach ist es erforderlich, den internen Adresszähler des Displays auf Null zu setzten und die ersten acht Zeichen auf dem Display darzustellen. Vor der Übergabe eines Zeichens an das Display muss überprüft werden, ob das Zeichen eine Zahl oder ein Buchstabe ist. Da - wie schon zuvor beschrieben - nichts anderes als Hexadezimalzahlen auf dem Display angezeigt werden, kann es sich dabei nur um die Buchstaben von A bis F handeln. Erkennt der Kontroller einen solchen Buchstaben, so wird dieser auf dem Display dargestellt. Nachdem nun die ersten acht Zeichen an das Display übergeben und angezeigt worden sind, ist es erforderlich, den internen Adresszähler des LCDs soweit zu erhöhen, dass alle eintreffenden Zeichen in der zweiten Zeile des LCDs angezeigt werden. Sobald alle Zeichen auf dem LCD angezeigt worden sind,

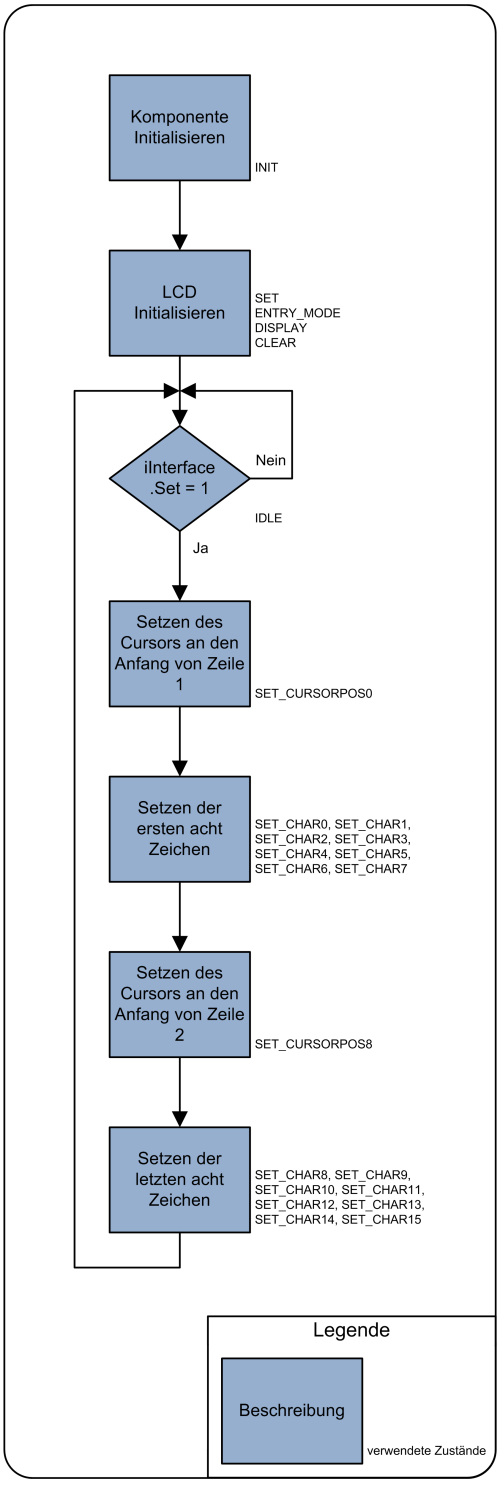

Bild 4.8: Flussdiagramm des **StateControl**-Prozesses beim LCD-Kontroller

wechselt der Prozess automatisch in den ***IDLE***-Zustand und wartet auf das Empfangen neuer Zeichen.
Zur Einhaltung der im Datenblatt [1, Seite 18 und folgende] angegeben Zeiten, wartet der Prozess **StateControl** bei jedem Zustand solange, wie es für diesen Zugriff nötig ist. Diese zeitliche Verzögerung wird mit Hilfe des vom Prozess **Counter** generierten Zählers `StateCounter` erwirkt. Zur Angabe eines absoluten Wertes bei jedem Zustand ist es nötig, den Zähler nach jedem Schritt in einen anderen Zustand mittels des Signals `StateCounterReset` zu löschen.

4.11 ClkDiv-Kontroller

Angesichts des ständigen Bedarfs an langsameren Takten im Design gibt es die **ClkDiv_Control**-Komponente. Mit Hilfe dieser Komponente ist es möglich, einen Takt zu verlangsamen. In Listing 4.5 wird die gesamte Komponente abgebildet:

Listing 4.5: ClkDiv_Control-Komponente

```
1  -- Der eigentliche Taktteilerprozess
2  Clock_Div : process (iClock, iReset)
3    variable CLK_VALUE : integer range 0 to CLK_DIV+1;
4  begin
5    -- Asynchroner Reset
6    if iReset = '1' then
7      intClock <= '0';
8      CLK_VALUE := 0;
9    elsif RISING_EDGE(iClock) then
10     CLK_VALUE := CLK_VALUE + 1;
11
12     -- Wenn der Maximalwert erreicht ist, dann setze die
13     -- Variabel zurück und toggle das Taktsignal
14     if CLK_VALUE = CLK_DIV then
15       intClock <= not intClock;
16       CLK_VALUE := 0;
17     end if;
18   end if;
19 end process Clock_Div;
20
21 -- Ausgabe des internen Taktsignal
22 oClock <= intClock;
```

Die Komponente besteht aus nur einem Prozess. Dieser zählt die Variable `CLK_VALUE` bei jeder steigenden Flanke von `iClock` hoch. Wenn die Variable `CLK_VALUE` den Maximalwert erreicht hat, wird das Signal `intClock` invertiert und der Wert der Variable

`CLK_VALUE` gelöscht. Bei jeder Veränderung des internen Signals `intClock` ändert das Ausgangssignal `oClock` seinen Wert entsprechend. Der Wert `CLK_DIV` wird bei der Instanziierung der Komponente in der **generic**-Anweisung angegeben.
Eine derartige Taktteilung ist im Allgemeinen nicht zu empfehlen und sollte nur mit Bedacht eingesetzt werden. Der Fitter kann nämlich beim Erstellen nicht erkennen, dass es sich bei dem Ausgangssignal von `oClock` um einen langsameren Takt als den Eingangstakt handelt. Aus diesem Grund versucht der Fitter das Ausgangssignal mit einem Takt von 156,25 MHz zu legen. Dies könnte zu einem Fehler führen, da der Fitter nicht in der Lage ist, die vorgegebenen Zeiten einzuhalten.

5 Test und Testergebnisse

Die Tests dienen - wie schon in Abschnitt 2.2 auf Seite 12 angekündigt - zur Ermittlung der Bitfehlerrate des Phys bei Störungen auf der Leitung. Zur Überprüfung der Messaufbauten auf Fehler innerhalb der Übertragungsstrecke sind diese jeweils ohne Störglieder aufgebaut und vermessen worden.
Im Folgenden wird jeder der drei Versuchsaufbauten jeweils mit jedem der weiter unten beschriebenen Störungsglieder getestet:

Messaufbau 1

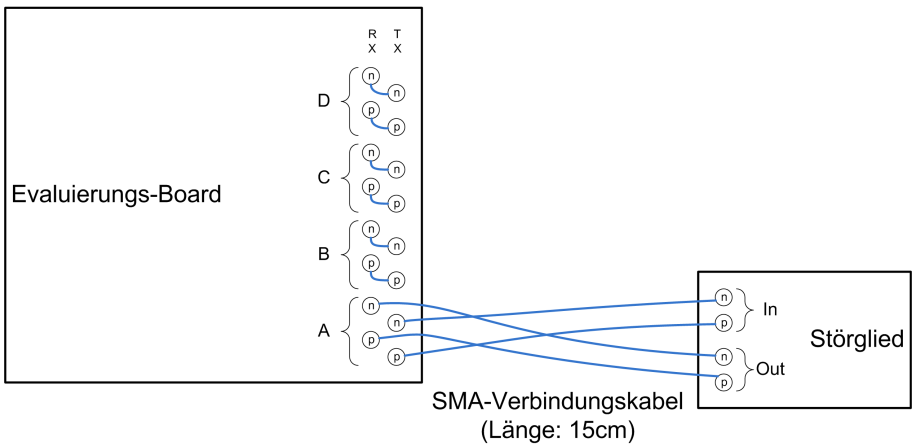

Bild 5.1: Messaufbau 1

Bei diesem Messaufbau gelangt das Signal vom Phy über die SMA-Verbinder direkt zum Störglied und dann zum Phy. Damit ist es möglich, das Verhalten des Phys beim Störglied ohne Kabel zu testen.

Messaufbau 2

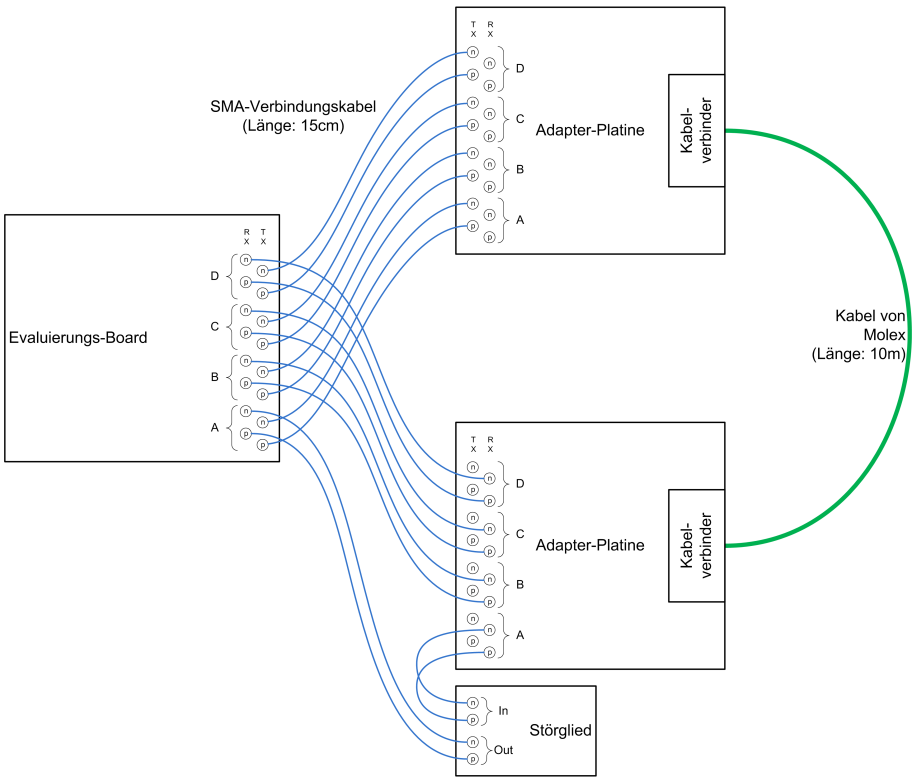

Bild 5.2: Messaufbau 2

In diesem Experiment werden die differentiellen Signale vom Phy zur Adapter-Platine und von da aus auf den Stecker geleitet, um die Signale per Kabel zu übertragen. Am Ende des Kabels befindet sich eine zweite Adapter-Platine, die die Signale über ein Störglied zum Phy zurückführt.

Messaufbau 3

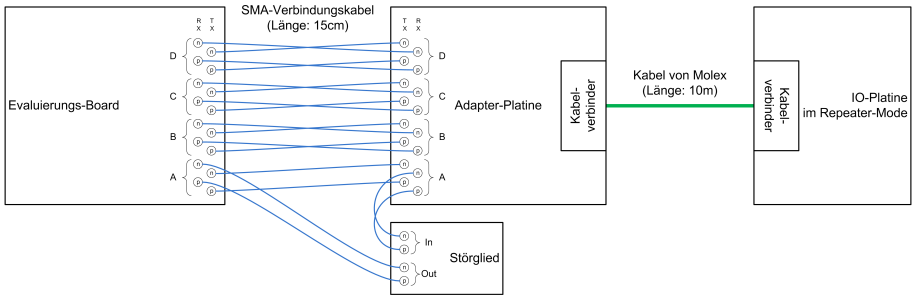

Bild 5.3: Messaufbau 3

Dieser Test dient zur Überprüfung der Kommunikation zwischen zwei Phys über ein Kabel. Die Signale werden vom Evaluierungs-Board zur Adapter-Platine übertragen. Die Adapter-Platine leitet das Signal über ein Kabel zur IO-Platine, die als Repeater arbeitet. Der Repeater empfängt das Signal, bereitet es auf und versendet es wieder. Vor einer Übertragung des Signals von der Adapter-Platine zum Phy ist es erforderlich, das Signal zuvor durch das Störglied zu schicken.

5.1 Messanforderungen

Zur Herstellung einer Vergleichbarkeit der Messergebnisse untereinander müssen die übertragenen Daten bei allen Tests identisch sein. Zu diesem Zweck erstellt das Tool „Create ROM“ ein ROM mit einer Größe von 1024 Bytes. Dieses Paket wird 20.971.519-mal versendet, was einem Datenvolumen von 20 GigaByte entspricht. Darüber hinaus findet jede Messung fünfmal statt, um exaktere Ergebnisse zu erzielen.
Der Phy versendet pro Sekunde circa 3 GigaBit, wobei durch die acht zu zehn Bit-Kodierung nur noch 2,4 GigaBit an reinen Nutzdaten enthalten sind. Diese 2,4 GigaBit entsprechen 292,969 kByte pro Sekunde. Somit benötigt der Phy für das Übertragen der 20 GigaByte ungefähr 71,58 Sekunden.
Außerdem werden die Daten auf jeder Lane versendet, wobei jedoch nur bei Lane A eine Störung durch Störglieder und eine Überprüfung der Daten stattfindet. Auf Grund der Tatsache, dass auf allen Lanes gleichzeitig gesendet wird, ist es erforderlich, eine mögliche Signalverstärkung der Daten durch ein Versenden von denselben Signalen in der Leitung zu verhindern. Dies geschieht durch die Addierung eines jeweils anderen Offsets beim Adresszähler für die Daten.
Das Versenden und Empfangen der Steuer-Kommandos wird in der Analyse der Bitfehler vernachlässigt.
Zur Gewährleistung einer späteren Kontrolle bzgl. der fehlerfreien Ankunft eines Packets ist es erforderlich, die CRC-Überprüfung von jedem Packet auszuwerten. Sobald Bitfehler erkannt worden sind, stimmt die CRC des Packets nicht mehr. Insofern kann man

annehmen, dass dieses Packet falsch empfangen worden ist. Ein Packet, bei dem die CRC-Überprüfung nicht stimmt, wird im Folgenden als „Fehlerpacket“ bezeichnet. Desweiteren werden zwischen jedem Packet fünf IDLE-Kommandos übertragen.

5.2 Dämpfungstest

Bei diesem Test wird - wie schon im Abschnitt 2.2 beschrieben - die differentielle Übertragungsstrecke mit einem Dämpfungsglied gestört. Ein solcher Test entspricht dem Testen von verschiedenen Leitungslängen, da jede Leitung auch eine Dämpfung besitzt. Im Messprotokoll des Kabels [2, Seite 23] ist verzeichnet, dass ein Bit ohne Störung bei einer Eingangsspannung von 1,0 V nur noch ein Ausgangsspannung ΔV von 443,2 mV besitzt. Somit ergibt sich für das Kabel mit einer Länge von 10 m eine Dämpfung von

$$\begin{aligned} D &= 20 \cdot log\left(\frac{U_e}{U_a}\right) \text{dB} \\ &= 20 \cdot log\left(\frac{1,0\,\text{V}}{0,4432\,\text{V}}\right) \text{dB} \\ &= 7,07\,\text{dB}. \end{aligned} \tag{5.1}$$

Dies entspricht einer Dämpfung von 0,707 $^{\text{dB}}/_{\text{m}}$.
In Abschnitt 1.2 auf Seite 4 wird das Ersatzschaltbild einer differentiellen Leitung dargestellt. Zur Konstruktion eines Dämpfungsgliedes für eine differentielle Leitung kann man wie folgt vorgehen:
Der Schaltungsaufbau eines einfachen asymmetrischen Dämpfungsglieds ist schon in [13, Seite 720] vorgegeben und dient dem im Folgenden beschriebenen Vorgehen als Vorlage. Da auch hier wie in Abschnitt 1.2 auf Seite 4 ein symmetrischer Leitungsaufbau erfolgen muss, wird der Widerstand R1 halbiert und gemäß Bild 5.4 aufgebaut.
Trotz der abgebildeten kleinen Veränderung des Filteraufbaus können dieselben Formeln

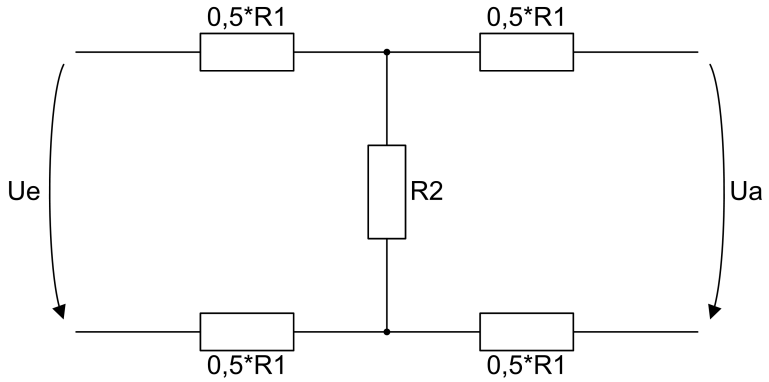

Bild 5.4: Symmetrisches Dämpfungsglied aus Widerständen

wie in [13, Seite 720] verwendet werden.
Zur Bestimmung einer Dämpfung D lässt sich der Spannungsfaktor $k = \frac{U_e}{Ua}$ wie folgt berechnen:

$$\begin{aligned} 20 \cdot log\left(\frac{U_e}{U_a}\right) &= D \\ 20 \cdot log\,(k) &= D \\ log\,(k) &= \frac{D}{20} \\ k &= 10^{\frac{D}{20}} \end{aligned} \tag{5.2}$$

Mit dem bekannten Wellenwiderstand Z von $100\,\Omega$ und dem in Formel 5.2 bestimmten Spannungsfaktor k können $R1$ und $R2$ wie folgt ermittelt werden:

$$R1 = Z \cdot \frac{k-1}{k+1} \tag{5.3}$$

$$R2 = Z \cdot \frac{2}{k - 1/k} \tag{5.4}$$

Mit diesen Formeln lässt sich für jede der verschiedenen Dämpfungen jeweils ein Dämpfungsglied erstellen. In Tabelle 5.1 sind die errechneten Widerstandswerte und die für den Aufbau verwendeten Widerstandswerte für Dämpfungen von 3 dB bis 18 dB dargestellt.

Dämpfung/dB	k	$R1/\Omega$ berechnet	$R2/\Omega$ berechnet	$\frac{R1/\Omega}{2}$ gewählt	$R2/\Omega$ gewählt
3	1,41	17,10	283,85	8,2	270
6	2,00	33,23	133,86	18	120
9	2,82	47,62	81,18	22	82
12	3,98	59,85	53,62	27	56
15	5,62	69,80	36,73	33	39
18	7,94	77,64	25,58	39	27

Tabelle 5.1: Berechnete Werte der Dämpfungsglieder

Zur Bestätigung der hier berechneten Werte wird das Dämpfungsglied mit dem Programm PSpice [16] aufgebaut und simuliert. Bild 5.5 auf der nächsten Seite zeigt den mit PSpice simulierten Schaltungsaufbau eines 3 dB-Dämpfungsglieds.
Die Widerstände $R1$ und $R2$ stellen zusammen mit der Spannungsquelle $V1$ den Sender dar und die Widerstände $R8$ und $R9$ mit dem Masse-Bezugspunkt dazwischen bilden den Empfänger. $R3$ bis $R7$ entsprechen der Beschaltung eines Dämpfungsglieds.
Die Spannungsquelle $V1$ generiert das Signal mit einer Frequenz von 1,6 GHz. Da sich der Masse-Bezugspunkt nicht direkt an der Spannungsquelle, sondern zwischen den Widerständen $R8$ und $R9$ befindet, wird am Ausgang von $V1$ eine Wechselspannung von

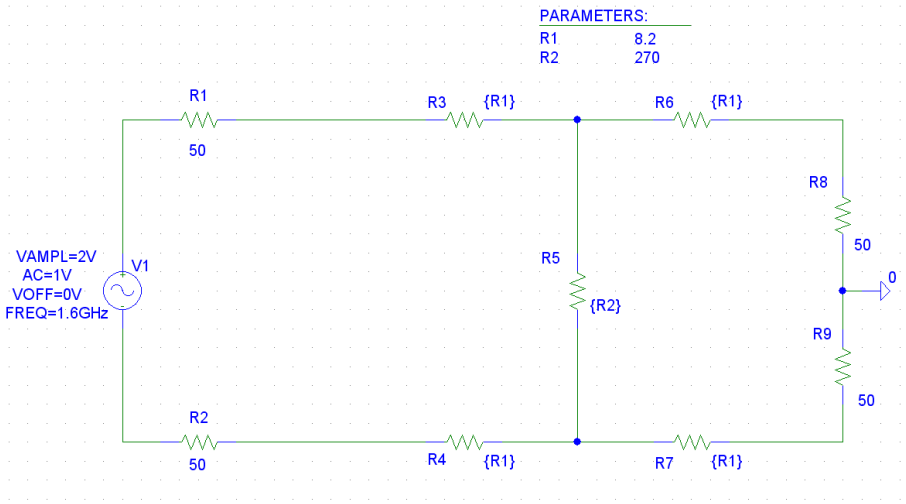

Bild 5.5: Schaltungsaufbau eines 3 dB-Dämpfungsglieds

-1 V bis +1 V eingestellt.
In Bild 5.6 wird die Simulation des Dämpfungsgliedes abgebildet.
Hier ist die Ausgangsspannung der Spannungsquelle in blau/ braun und das gedämpfte

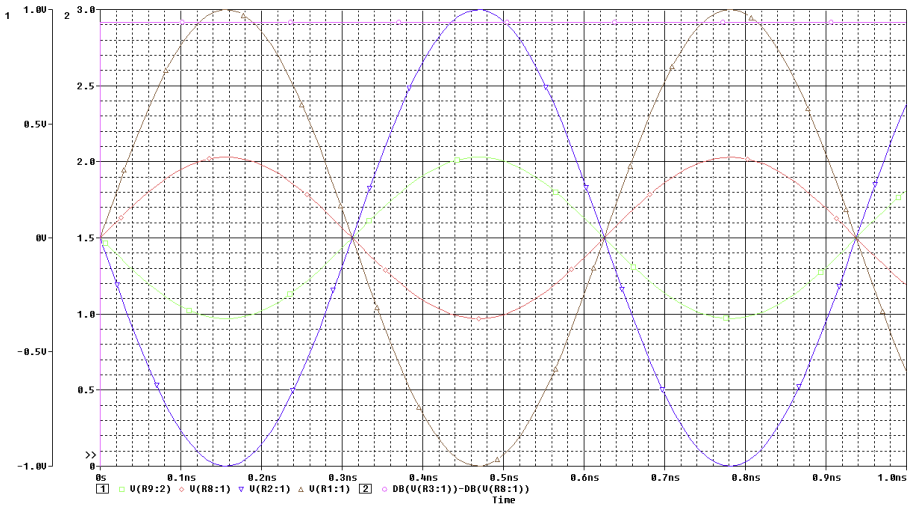

Bild 5.6: Simulation des 3 dB-Dämpfungsglieds

Signal hinter dem Dämpfungsglied in grün/ rot dargestellt. Die rosane Gerade zeigt die Dämpfung des Signals in dB an. Diese liegt bei circa 2,91 dB. Somit stimmt der berechnete mit dem simulierten Wert überein.
Durch Verändern der Parameter $R1$ und $C1$ lassen sich die anderen Dämpfungsglieder schnell und einfach simulieren. Alle Dämpfungsglieder sind mit der Simulation überprüft worden und stimmen mit den berechneten Werten überein.

Die Bitfehlerrate BER[1] lässt sich folgendermaßen bestimmen:

$$\begin{aligned} BER &= \frac{\text{Bitfehler}}{20971519\,\text{Pakete} \cdot 1024\,{}^{\text{Byte}}/_{\text{Paket}} \cdot 8\,{}^{\text{Bit}}/_{\text{Byte}}} \\ &= \frac{\text{Bitfehler}}{171798683648\,\text{Bit}} \end{aligned} \tag{5.5}$$

Da bei jeder Messung 171798683648 Bit übertragen werden, ist es theoretisch möglich, dass nach dem Versenden des letzten Bits ein Bitfehler auftritt, der nicht erkannt wird. Aus diesem Grund besitzt die kleinstmögliche Bitfehlerrate nicht den Wert Null, sondern einen Wert, der unterhalb von $\frac{1}{171798683648} = 5,82 \cdot 10^{-12}$ liegt. Zusätzlich wird aus den fünf Messungen der Mittelwert der Bitfehlerrate berechnet.
Die Bitfehlerrate ist für jeden Versuchsaufbau wie in Tabelle 5.2 dargestellt gemessen worden. Diese Bitfehlerrate ergibt sich aus den Dämpfungsgliedern von der Tabelle 5.1 der letzten Seite, deren Messungen aus Tabelle A.3, A.4, A.5 auf Seite 77 und folgender und der Formel 5.5 stammen.

	Bitfehlerrate		
D/dB	Messaufbau 1	Messaufbau 2	Messaufbau 3
3	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$
6	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$
9	$< 5,82 \cdot 10^{-12}$	$3,976 \cdot 10^{-5}$	$8,221 \cdot 10^{-8}$
12	$< 5,82 \cdot 10^{-12}$	1	1
15	$< 5,82 \cdot 10^{-12}$	1	1
18	1	1	1

Tabelle 5.2: Bitfehlerrate bei der Dämpfungsmessung

In Bild 5.7 auf der nächsten Seite ist die Bitfehlerrate in Abhängigkeit von der Dämpfung abgetragen.
Dabei stehen Bitfehlerrate und Dämpfung in einem logarithmischen Zusammenhang. Eigentlich wäre beim 9 dB-Dämpfungsglied und bei Messaufbau 3 dieselbe Bitfehlerrate wie bei Messaufbau 2 und demselben Dämpfungsglied zu erwarten gewesen. Wenn man die Bitfehler in Abhängigkeit von den Fehlerpaketen betrachtet, ist zu sehen, dass es bei 9 dB und Messaufbau 2 circa acht Bitfehler pro Fehlerpaket gibt. Bei Messaufbau 3 und 9 dB sind es nur noch circa vier Bitfehler pro Fehlerpaket.
Diese augenscheinliche Unstimmigkeit lässt sich folgendermaßen erklären: Bei Messaufbau 2 sind in einer Messstrecke drei mit SMA-Kabeln[2] verknüpfte Verbindungen entstanden. Eine solche Dämpfung der SMA-Stecker, SMA-Verbinder und der Koaxialleitung fällt natürlich beim Test ins Gewicht. Bei Messaufbau 3 gibt es nur zwei solcher Verbindungen, die mit SMA-Kabeln verknüpft worden sind und die Messung beeinflussen. Die

[1] Bit Error Rate
[2] hochfrequente Koaxialleitungen mit SMA-Steckern an jeder Seite

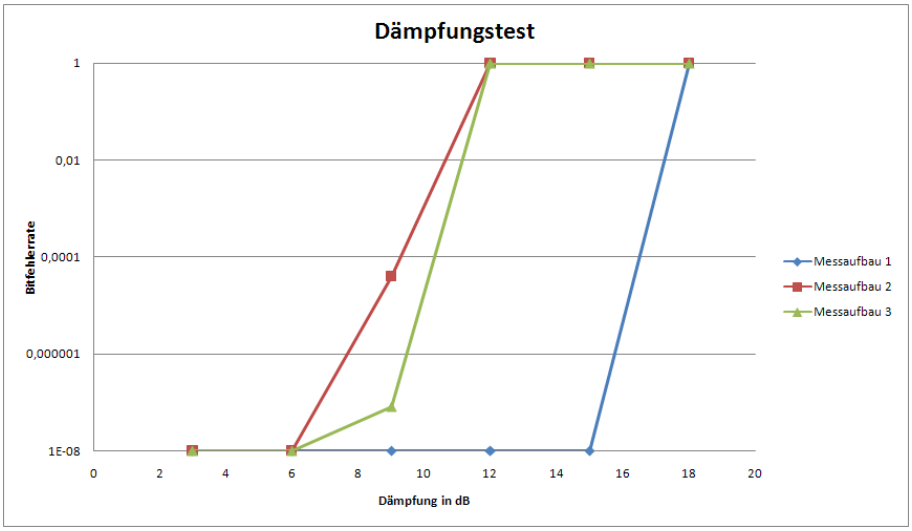

Bild 5.7: Bitfehlerrate in Abhängigkeit von der Dämpfung

Messung wird insofern beeinflusst, dass das IO-Board das Signal empfängt, aufbereitet und dann wieder versendet. Daher fällt die Gesamtdämpfung bei Messaufbau 3 geringer als bei Messaufbau 2 aus und demzufolge werden beim dritten Messaufbau geringere Bitfehlerraten erreicht.
Nun ist es möglich, mit Hilfe dieses Tests eine ungefähre maximale Kabellänge anzugeben. Bis zu einer Dämpfung von weniger als 15 dB besteht eine Bitfehlerrate unter $5,82 \cdot 10^{-12}$. Bei einer Kabeldämpfung von circa $0,7\,^{\mathrm{dB}}/_{\mathrm{m}}$, die durch die Messwerte aus Messaufbau 1 und 2 verifiziert worden ist, ergibt sich eine maximale Kabellänge von etwas mehr als 20 m.

5.3 Tiefpasstest

Dieser Test dient zur Untersuchung des Verhaltens des Phys bei der Störung durch einen Tiefpassfilter. Zur gleichmäßigen Belastung der Übertragungsstrecke benötigt auch dieses Filter einen symmetrischen Aufbau - dies wird mittels eines doppelt eingesetzten Widerstandes $R1$ realisiert. In Bild 5.8 auf der nächsten Seite ist der Aufbau eines symmetrischen Tiefpassfilters in der Übertragungsstrecke dargestellt. Die Grenzfrequenz eines solchen Filters wird mit Formel 5.6 wie folgt beschrieben:

$$fg = \frac{1}{2 \cdot \pi \cdot Rg \cdot C} \tag{5.6}$$

Wenn die gestrichelten Leitungen in Bild 5.8 auf der nächsten Seite kurzgeschlossen werden, erkennt der Kondensator den Widerstand Rg, der sich zwischen den Punkten A und B befindet, als Gesamtwiderstand. Dieser Gesamtwiderstand lässt sich über die folgende

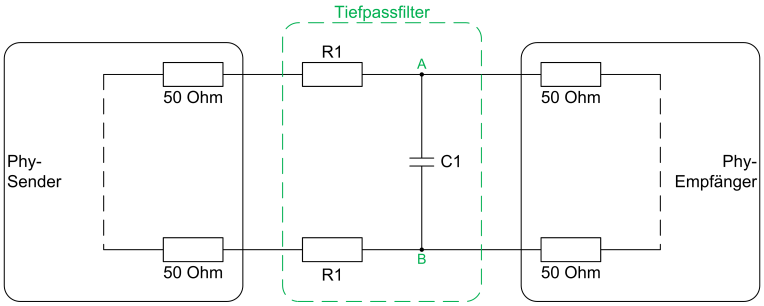

Bild 5.8: Symmetrisches Tiefpass-Filterglied

Formel berechnen:

$$\begin{aligned} Rg &= (2 \cdot R1 + 100\,\Omega) \,||\, 100\,\Omega \\ &= \frac{(2 \cdot R1 + 100\,\Omega) \cdot 100\,\Omega}{200\,\Omega + 2 \cdot R1} \end{aligned} \tag{5.7}$$

Zur Bestimmung des Filters wird Formel 5.7 nach $R1$ umgestellt:

$$\begin{aligned} Rg &= \frac{(2 \cdot R1 + 100\,\Omega) \cdot 100\,\Omega}{200\,\Omega + 2 \cdot R1} \\ (100\,\Omega + 2 \cdot R1) \cdot 100\,\Omega &= Rg \cdot (200\,\Omega + 2 \cdot R1) \\ 100^2\,\Omega^2 + 200\,\Omega \cdot R1 &= 200\,\Omega \cdot Rg + 2 \cdot R1 \cdot Rg \\ 200\,\Omega \cdot R1 - 2 \cdot R1 \cdot Rg &= 200\,\Omega \cdot Rg - 100^2\,\Omega^2 \\ R1\,(200\,\Omega - 2 \cdot Rg) &= 200\,\Omega \cdot Rg - 100^2\,\Omega^2 \\ R1 &= \frac{200\,\Omega \cdot Rg - 100^2\,\Omega^2}{200\,\Omega - 2 \cdot Rg} \end{aligned} \tag{5.8}$$

Das zu übertragene Signal besitzt im Idealfall eine unendliche Flankensteilheit und somit eine Rechteckcharakteristik. Bei jedem Senden einer Null-Eins-Folge werden maximal 3,125 $^{\mathrm{GBit}}/_{\mathrm{s}}$ übertragen; dies entspricht einer maximalen Übertragungsfrequenz von $f = 1,6\,\mathrm{GHz}$.
Zur Erkennung eines Rechtecksignals benötigt man nach der Fouriertransformation mindestens vier Sinusquellen mit den Frequenzen f, $3f$, $5f$ und $7f$. Um das Verhalten des Kondensators bei den Tiefpassfiltern auch mit einer Frequenz von $9f$ zu untersuchen[3], ist dieses Filter aufgebaut und getestet worden. Aus diesem Grund wird die Übertragungsstrecke mit Tiefpassfiltern, deren Eckfrequenzen $f = 1,6\,\mathrm{GHz}$, $3f = 4,8\,\mathrm{GHz}$, $5f = 8,0\,\mathrm{GHz}$, $7f = 11,2\,\mathrm{GHz}$ bzw. $9f = 14,4\,\mathrm{GHz}$ betragen, gestört.
Mit Formel 5.8 und den Grenzfrequenzen lassen sich die nötigen Filter bestimmen. In Tabelle 5.3 auf der nächsten Seite sind die verwendeten Widerstands- und Kondensatorwerte der erstellten Filter aufgelistet.

[3] vgl. Abschnitt 1.3 auf Seite 5

fg/GHz	$C1/pF$ gewählt	Rg/Ω berechnet	$R1/\Omega$ berechnet	$R1/\Omega$ gewählt
1,6	1,5	66,31	48,43	47
4,8	0,5	66,31	48,43	47
8,0	0,33	60,29	25,90	27
11,2	0,2	71,05	72,72	68
14,4	0,2	55,26	11,76	12

Tabelle 5.3: Berechnete Werte der Filterglieder

Zur Kontrolle der berechneten Werte ist das Filter mit PSpice simuliert worden. In Bild 5.9 wird der Schaltungsaufbau der Simulation abgebildet.
Die Spannungsquelle und die Widerstände $R1$ und $R2$ fungieren wie beim Dämpfungs-

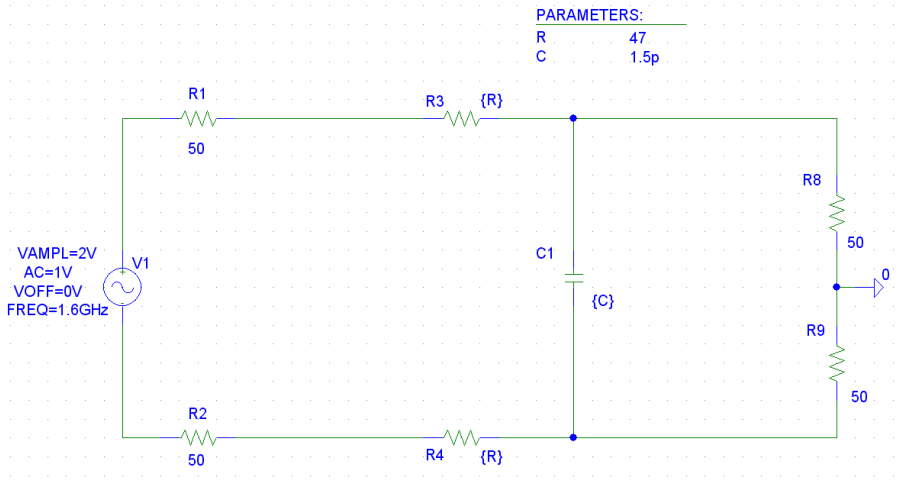

Bild 5.9: Schaltungsaufbau des 1,6 GHz-Filters

glied als Sender und die Widerstände $R8$ und $R9$ stellen den Empfänger der Schaltung dar. R3 und R4 bilden zusammen mit dem Kondensator $C1$ einen Tiefpassfilter mit einer Grenzfrequenz von 1,6 GHz. In Bild 5.10 auf der nächsten Seite ist die Simulation des Filters zu sehen.
Die Grenzfrequenz des Filters beträgt - wie schon oben ausgerechnet - 1,6 GHz. Alle anderen Filter sind mit den Parametern R und C simuliert worden und haben die berechneten Werte verifiziert.
Wie schon beim Dämpfungstest gezeigt, kann Formel 5.5 auf Seite 64 für die Berechnung der Bitfehlerrate verwendet werden. Außerdem ist auch hier die kleinste Bitfehlerrate nicht Null, sondern kleiner als $5,82 \cdot 10^{-12}$.
Die Bitfehlerrate ist für jeden Versuchsaufbau wie in Tabelle 5.4 auf der nächsten Seite dargestellt gemessen worden. Diese Bitfehlerrate ergibt sich aus den Filtergliedern aus Tabelle 5.3, deren Messungen aus den Tabellen A.6, A.7, A.8 von Seite 80 und folgende

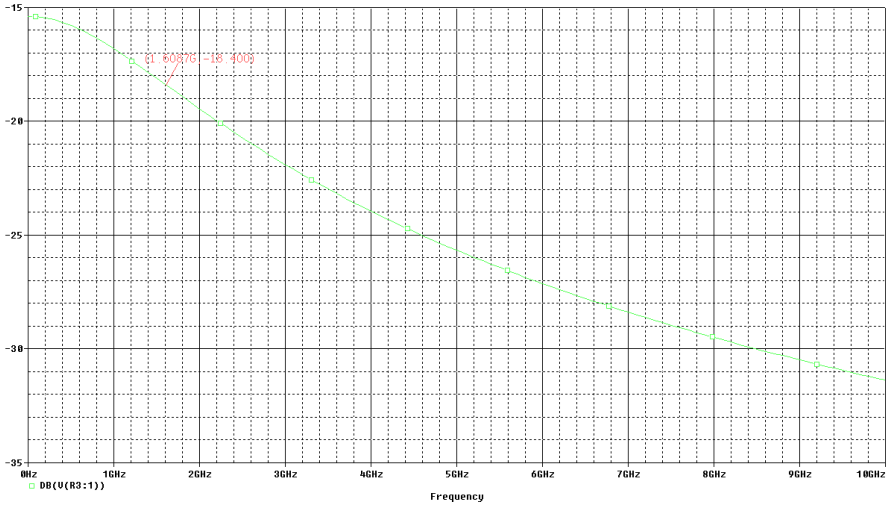

Bild 5.10: Simulation des 1,6 GHz-Filters

und der Formel 5.5 auf Seite 64.

	Bitfehlerrate		
D/dB	Messaufbau 1	Messaufbau 2	Messaufbau 3
1,6	$< 5,82 \cdot 10^{-12}$	$3,32 \cdot 10^{-4}$	$5,36 \cdot 10^{-6}$
4,8	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$
8,0	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$
11,2	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$
14,4	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$	$< 5,82 \cdot 10^{-12}$

Tabelle 5.4: Bitfehlerrate bei der Tiefpassmessung

In Bild 5.11 auf der nächsten Seite ist die Bitfehlerrate in Abhängigkeit von der Grenzfrequenz abgetragen.
Wie schon beim Dämpfungstest liegt auch hier die Bitfehlerrate bei Messaufbau 3 unterhalb der von Messaufbau 2. Hier gilt ebenfalls, dass die Bitfehler pro Fehlerpacket bei Messaufbau 2 größer als bei Messaufbau 3 sind. Die Erklärung hierfür ist analog zu der für den Dämpfungstest: Messaufbau 2 weist wieder mehr SMA-Kabel-Verbindungen als Messaufbau 3 auf, dessen Signal der Repeater aufbereitet hat. Jede SMA-Verbindung dämpft auch hier das Signal, was zusätzliche Bitfehler hervorruft.
Insofern hat dieser Test gezeigt, dass das Übertragungssignal nicht unbedingt eine Leitung mit einer Übertragungsbandbreite von 14,4 GHz benötigt. Schon die Verwendung einer Leitung mit einer Übertragungsbandbreite von mindestens 4,8 GHz ist völlig ausreichend, um eine Bitfehlerrate von weniger als $5,82 \cdot 10^{-12}$ zu erreichen.

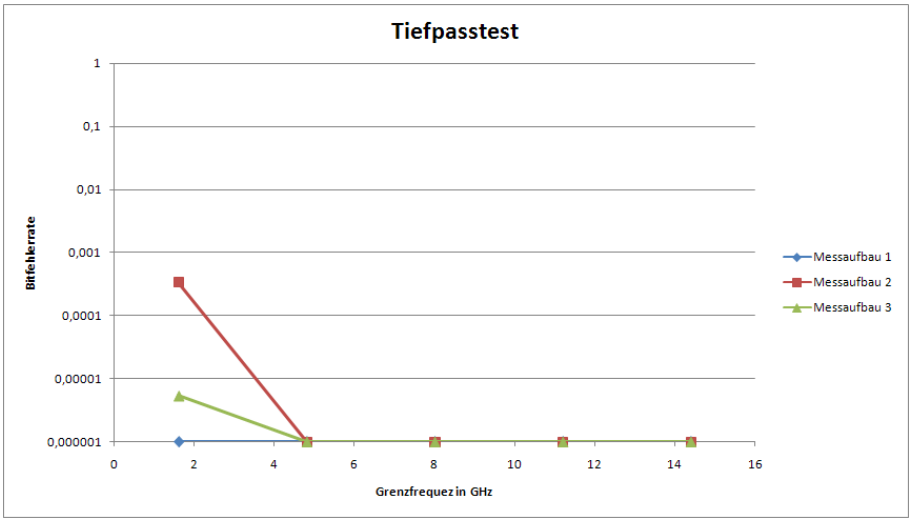

Bild 5.11: Bitfehlerrate in Abhängigkeit von der Grenzfrequenz

5.4 Preemphasistest

Wie schon im Abschnitt 3.1 auf Seite 18 ausgeführt erzielt die Preemphasis ein optimaleres Signal-Rausch-Verhältnis und wirkt der Tiefpasscharakteristik der Leitung entgegen. Bei diesem Test wird die Preemphasis abgeschaltet und das Verhalten des Phys bei einem Dämpfungsglied mit 9 dB-Dämpfung und einem Tiefpassfilter mit einer Grenzfrequenz von 1,6 GHz untersucht.
Aus den Messergebnissen der Tabelle A.9 auf Seite 83 und den zuvor gemessenen Werten hat die Bitfehlerrate und die Bitfehler pro Fehlerpacket - wie in Tabelle 5.5 und Tabelle 5.6 auf der nächsten Seite abgebildet - ermittelt werden können.

		Bitfehlerrate		
Messung	Preemphasis	Messaufbau 1	Messaufbau 2	Messaufbau 3
$D = 9\,\mathrm{dB}$	An	$< 5,82 \cdot 10^{-12}$	$3,976 \cdot 10^{-5}$	$8,221 \cdot 10^{-8}$
	Aus	$< 5,82 \cdot 10^{-12}$	$7,609 \cdot 10^{-4}$	$1,318 \cdot 10^{-7}$
$fg = 1,6\,\mathrm{GHz}$	An	$< 5,82 \cdot 10^{-12}$	$3,324 \cdot 10^{-4}$	$5,358 \cdot 10^{-6}$
	Aus	$< 5,82 \cdot 10^{-12}$	$1,769 \cdot 10^{-3}$	$1,302 \cdot 10^{-5}$

Tabelle 5.5: Bitfehlerrate beim Preemphasistest

	Bitfehler pro Fehlerpaket			
Messung	Preemphasis	Messaufbau 1	Messaufbau 2	Messaufbau 3
$D = 9\,\mathrm{dB}$	An	$0,000$	$7,950$	$4,664$
	Aus	$0,000$	$8,065$	$4,533$
$fg = 1,6\,\mathrm{GHz}$	An	$0,000$	5.966	$3,113$
	Aus	$0,000$	$18,603$	$3,013$

Tabelle 5.6: Bitfehler pro Fehlerpaket beim Preemphasistest

In Tabelle 5.5 von der letzten Seite ist zu sehen, dass bei abgeschalteter Preemphasis die Bitfehlerrate im Vergleich zu den vorher gemessenen Werten zunimmt. Auf Grund der Erzielung eines besseren Signal-Rausch-Verhältnisses durch den Einsatz der Preemphasis ist dies zu erwarten gewesen.
Tabelle 5.6 zeigt auf, dass die Bitfehler pro Fehlerpacket nur bei Messaufbau 2 und beim Tiefpassfilter angestiegen sind. Bei allen anderen Messungen ist die Anzahl der Bitfehler pro Fehlerpaket konstant geblieben.
Da die Preemphasis dem Entgegenwirken gegen die Tiefpasscharakteristik der Leitung dient, kann man ein solches Verhalten auch erwarten. Durch das Abschalten der Preemphasis lässt sich dieser Tiefpass nicht mehr verbessern. Außerdem können angesichts der Tatsache, dass keine Umprogrammierung des Phys auf dem IO-Board stattgefunden hat und die Preemphasis nicht abgeschaltet worden ist, bei Messaufbau 3 und beim Tiefpassfilter keine Änderungen der Bitfehler pro Fehlerpaket entstehen.
Infolgedessen sollte man nach Möglichkeit die Preemphasis immer eingeschalten, um bessere Bitfehlerraten zu erzielen.

Zusammenfassung und Ausblick

Wie die Tests aus dem fünften Kapitel bestätigen, lässt sich mit dem Phy unter den gegebenen Bedingungen eine Datenübertragungsrate von 3,125 $^{\text{GBit}}/_{\text{sec}}$ erzielen. Die in der Einleitung angeführten möglichen Schwierigkeiten, I und II, haben folgendermaßen gelöst werden können:

zu I Durch eine geschickte Wahl der verwendeten Bauelemente (Widerstände und Kondensatoren) sind deren nichtlineare Eigenschaften gar nicht erst eingetreten. So ist in 1.3 gezeigt worden, dass die nichtlinearen Eigenschaften des Kondensators vernachlässigt werden können. Bei der Wahl der Widerstände sind merkliche Abweichungen vom Idealwert vermieden worden.
Durch die Verwendung des Programms HyperLynx ist die Bestimmung des Lagenaufbaus der Platine möglich gewesen. Dieser Lagenaufbau der Platine hat bei der Erstellung des IO-Boards verwendet werden können. Die Bestimmung der Leitungslänge für die Signalausbreitungsgeschwindigkeit ist ebenfalls erfolgreich ermittelt und angewendet worden. Dabei hat sich ergeben, dass die Länge der Signalleitungen um 3 mm differieren darf, um Laufzeitunterschiede zu verhindern.

zu II Die Kodierung des Signals durch den Phy (siehe 3.1) hat bewirkt, dass die Bitfehlerrate bei der Signalübertragung äußerst klein geworden ist. So hat sich beispielsweise beim Dämpfungsglied zeigen lassen, dass die Bitfehlerrate bei einer Gesamtleitungslänge von 20 m kleiner als $5,82 \cdot 10^{-12}$ gewesen ist. Der Vorteil dabei besteht darin, dass sich der PC zum Empfangen und Auswerten der Kameradaten nicht unmittelbar in der Nähe der Kamera befinden muss.

Zum Testen dieses Phys mit einer A400-Kamera wären zwei Optionen denkbar:
Erstens könnte die notwendige Referenzspannung der IO-Bank durch eine Neuerstellung des Processing-Boards richtig angeschlossen werden.
Zweitens ließe sich durch eine Neuerstellung des IO-Boards ein zusätzliches FPGA anschließen. Dadurch könnten die Daten in den SSTL_2-Standard umgesetzt werden.
Beide dieser Verbesserungsvorschläge wären vorstellbar. Dabei ist allerdings zu berücksichtigen, dass der Hersteller bereits einen neuen Phy angekündigt hat. Dieser neue Phy besitzt die gleiche Datenübertragungsgeschwindigkeit wie der hier verwendete. Zudem ist sowohl die Leistungsaufnahme als auch die Wärmeabstrahlung des angekündigten Phys deutlich geringer.

Alles in allem sind die erhaltenen Testergebnisse des Phys für den neunen Übertragungsstandard Fast CameraLink auf Grund der geringen Bitfehlerraten und der hohen Datenübertragungsrate äußerst vielversprechend.

Literaturverzeichnis

[1] *16COM / 40SEG DRIVER & CONTROLLER FOR DOT MATRIX LCD.* CD:/Datenblätter/LCD/L1671.pdf.

[2] *4X / 12X LaneLink Infiniband Cable Assembly Test Report.* CD:/Datenblätter/Kabel/LaneLink Kabeltest.pdf.

[3] *INFINIBAND 4X Plug to Plug.* CD:/Datenblätter/Kabel/745063107_sd.pdf.

[4] *1.5A, 500kHz Step-Down Switching Regulators.* CD:/Datenblätter/Schaltregler/LT1376HVC.pdf, 1995.

[5] *TPS54350.* CD:/Datenblätter/Schaltregler/tps54350.pdf, 2003.

[6] *Chip EMIFIL LC Combined Type.* CD:/Datenblätter/Allgemeine Bauteile/Murata_NFE31PT222Z1E9 (C53).pdf, 2005.

[7] *Chip EMIFIL LC Combined Type.* CD:/Datenblätter/Allgemeine Bauteile/JFET, BF545C, n-Chan , 25mA30V, SOT23 (Q6).pdf, 2005.

[8] *Stratix II Device Handbook.* CD:/Datenblätter/FPGA/Stratix/EP2S30F672C5N, FPGA, FBGA-672.pdf, 2006.

[9] Peter J. Ashenden. *The Designers's Guide to VHDL*, volume 2. Academic Press, San Diego, 2002.

[10] Camera Link Group, CD:/Datenblätter/CamerLink/CameraLink v1_1(B).pdf. *Camer Link - Specigication of the Camera Link Interface Standard for Digital Cameras and Frame Grabbers*, 2004.

[11] Dr. Zellmer GmbH, http://www.firewire-infos.de/ (15.08.2007). *Firewire-Infos*, 2007.

[12] Dr. Zellmer GmbH, http://www.usb-infos.de/(15.08.2007). *USB-Infos*, 2007.

[13] Günter Käs and Peter Pauli. *Mikrowellentechnik*, volume 1. Franzis, 8000 München, Januar 1991.

[14] MachineVision Online, http://www.machinevisiononline.org/public/articles/ index.cfm?cat=167 (15.08.2007). *GigE Vision*, 2007.

[15] Mentor Graphics, http://www.mentor.com (02.08.2007). *HyperLynx*, 2003.

[16] OrCAD, http://www.cadence.com (08.08.2007). *PSpice Student*, 1999. version 9.1.

[17] TDK, http://www.tdk.co.jp/ccv/index.asp (26.07.2007). *Component Characteristic Viewer*, August 2007.

[18] Texas Instrument, CD:/Datenblätter/Phy/tlk3114sc.pdf. *TLK3114SC - Data Manual*, November 2006.

[19] Ulrich Tietze and Christoph Schenk. *Halbleiter-Schaltungstechnik*, volume 12. Springer Verlag, Berlin Heidelberg, Januar 2002.

[20] Wikipedia, http://en.wikipedia.org/wiki/8B10B (02.08.2007). *8B10B encoding*, 2007.

[21] Wikipedia, http://de.wikipedia.org/wiki/OSI-Modell (02.08.2007). *OSI-Modell*, 2007.

[22] Yageo, CD:/Datenblätter/Allgemeine Bauteile/Widerstände.pdf. *DATA SHEET CHIP RESISTORS*, Dezember 2004.

Anhang

A Tabellen

A.1 8B10B-Kodierung

Die folgenden zwei Tabellen und deren Anordnung ist [20] entnommen.

fünf Bit-Wort	sechs Bit-Wort		fünf Bit-Wort	sechs Bit-Wort	
	RD = -1	RD = +1		RD = -1	RD = +1
EDCBA	abcdei		EDCBA	abcdei	
00000	100111	011000	10000	011011	100100
00001	011101	100010	10001	100011	
00010	101101	010010	10010	010011	
00011	110001		10011	110010	
00100	110101	001010	10100	001011	
00101	101001		10101	101010	
00110	011001		10110	011010	
00111	111000	000111	10111	111010	000101
01000	111001	000110	11000	110011	001100
01001	100101		11001	100110	
01010	010101		11010	010110	
01011	110100		11011	110110	001001
01100	001101		11100	001110	
01101	101100		11101	101110	010001
01110	011100		11110	011110	100001
01111	010111	101000	11111	101011	010100

Tabelle A.1: Fünf zu sechs Bit-Kodierung

drei Bit-Wort	vier Bit-Wort	
	RD = -1	RD = +1
HGF	fghj	
000	1011	0100
001	1001	
010	0101	
011	1100	0011
100	1101	0010
101	1010	
110	0110	
111	1110	0001

Tabelle A.2: Drei zu vier Bit-Kodierung

A und a stehen für das LSB.

A.2 Messergebnisse

Dämpfung	Messung	Messaufbau 1 Bitfehler	Fehlerpakete
3 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
6 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
9 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
12 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
15 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
18 dB	1	4031	20971519
	2	4049	20971519
	3	4012	20971519
	4	4049	20971519
	5	4012	20971519

Tabelle A.3: Messergebnisse der Dämpfungsmessung bei Messaufbau 1

Dämpfung	Messung	Messaufbau 2 Bitfehler	Fehlerpakete
3 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
6 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
9 dB	1	6558430	825252
	2	6408183	805434
	3	7061535	887748
	4	6914030	870486
	5	7214850	907838
12 dB	1	1787	20971519
	2	1485	20971519
	3	2042	20971519
	4	2042	20971519
	5	1787	20971519
15 dB	1	0	20971519
	2	0	20971519
	3	0	20971519
	4	0	20971519
	5	0	20971519
18 dB	1	0	20971519
	2	0	20971519
	3	0	20971519
	4	0	20971519
	5	0	20971519

Tabelle A.4: Messergebnisse der Dämpfungsmessung bei Messaufbau 2

Dämpfung	Messung	Messaufbau 3 Bitfehler	Fehlerpakete
3 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
6 dB	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
9 dB	1	12738	2746
	2	12658	2747
	3	15630	3352
	4	14956	3164
	5	14638	3124
12 dB	1	2731	20971519
	2	2731	20971519
	3	4063	20971519
	4	4012	20971519
	5	4063	20971519
15 dB	1	0	20971519
	2	0	20971519
	3	0	20971519
	4	0	20971519
	5	0	20971519
18 dB	1	0	20971519
	2	0	20971519
	3	0	20971519
	4	0	20971519
	5	0	20971519

Tabelle A.5: Messergebnisse der Dämpfungsmessung bei Messaufbau 3

Grenzfrequenz	Messung	Messaufbau 1 Bitfehler	Fehlerpakete
1,6 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
4,8 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
8,0 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
11,2 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
14,4 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0

Tabelle A.6: Messergebnisse der Tiefpassmessung bei Messaufbau 1

Grenzfrequenz	Messung	Messaufbau 2 Bitfehler	Fehlerpakete
1,6 GHz	1	58798546	12620769
	2	57753184	9094033
	3	57994961	9149818
	4	56153612	8961352
	5	54866050	8829171
4,8 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
8,0 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
11,2 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
14,4 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0

Tabelle A.7: Messergebnisse der Tiefpassmessung bei Messaufbau 2

Grenzfrequenz	Messung	Messaufbau 3 Bitfehler	Fehlerpakete
1,6 GHz	1	845406	265782
	2	789719	254416
	3	892591	288634
	4	1015001	327884
	5	1059564	342670
4,8 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
8,0 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
11,2 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
14,4 GHz	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0

Tabelle A.8: Messergebnisse der Tiefpassmessung bei Messaufbau 3

	Messaufbau 1		
	Messung	Bitfehler	Fehlerpakete
$fg = 1,6\,\mathrm{GHz}$	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
$D = 9\,\mathrm{dB}$	1	0	0
	2	0	0
	3	0	0
	4	0	0
	5	0	0
	Messaufbau 2		
	Messung	Bitfehler	Fehlerpakete
$fg = 1,6\,\mathrm{GHz}$	1	311494521	18506046
	2	275201759	15837144
	3	296851603	15880020
	4	308636921	15833564
	5	327302039	15872289
$D = 9\,\mathrm{dB}$	1	132310633	16285610
	2	132771517	16281870
	3	132983388	16314917
	4	127021868	15987003
	5	128561466	16168409
	Messaufbau 3		
	Messung	Bitfehler	Fehlerpakete
$fg = 1,6\,\mathrm{GHz}$	1	2333117	742979
	2	2200655	743365
	3	2275552	745048
	4	2188195	739357
	5	2183608	739778
$D = 9\,\mathrm{dB}$	1	23320	5123
	2	26311	5913
	3	19845	4387
	4	22982	5049
	5	20792	4533

Tabelle A.9: Messergebnisse der Preemphasismessung

B Create ROM

Mit diesem Programm ist die Erstellung eines Datenpackets für die in VHDL geschriebene Komponente **ROM_Control** durchführbar. Für die Zufallszahlen wird die bereitgestellte Funktion RAND verwendet. In dem von dem Programm erstellten Datenpacket sind außerdem sowohl die Größe der Adresse als auch die Anzahl der Bytes vorhanden. Darüber hinaus werden im Datenpacket einige Konstanten definiert, die für das Versenden der Daten erforderlich sind. Der Quellcode des Programms befindet sich auf der beigelegten CD im Ordner *CD:/Quellcode/CS/*. Außerdem enthält die CD ein startfähiges Programm, das das DotNET-Framework 2.0 voraussetzt.

C Inhalt der beigelegten CD

- Diplomarbeit im PDF-Format
  CD:/
- Schalplan des IO-Boards im PDF-Format
  CD:/Schaltplan/
- Datenblätter der verwendeten Bauteile
  CD:/Datenblätter/
- VDHL-Quellcode
  CD:/Quellcode/VHDL/
- Create ROM-Projekt
  CD:/Quellcode/CS/
- PSpice-Simulationsdateien
  CD:/Simulation/

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §25(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtliche oder dem Sinn nach aus anderen Werken entnommmen Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Ort, Datum, Unterschrift