Fachhochschule Westküste
Hochschule für Wirtschaft & Technik

HAW HAMBURG

Masterarbeit

Hagen Steinbach

Entwicklung eines integrierten, digitalen Empfängers für echtzeitfähige, leitungsungebundene, optische Übertragungen

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Computer Science and Engineering
Department Information and Electrical Engineering

Hagen Steinbach

Entwicklung eines integrierten, digitalen Empfängers für echtzeitfähige, leitungsungebundene, optische Übertragungen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Fachbereich Technik
der Fachhochschule Westküste
und
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Robert Fitz
Zweitgutachter: Prof. Dr.-Ing. Alfred Ebberg
Externer Betreuer: Dr.-Ing Michael Faulwaßer (Fraunhofer IPMS)

Eingereicht am: 28.02.2020

Hagen Steinbach

Thema der Arbeit

Entwicklung eines integrierten, digitalen Empfängers für echtzeitfähige, leitungsungebundene, optische Übertragungen

Stichworte

Takt- und Datenrückgewinnung, CDR, optische Datenübertagung, OWC, FPGA

Kurzzusammenfassung

In dieser Arbeit wird eine komplett-digitale Daten- und Taktrückgewinnung auf einem FPGA für echtzeitfähige, kabelungebundene, optische Übertragungssysteme realisiert. Es werden zunächst drei unterschiedliche Lösungskonzepte vorgestellt und deren Vor- und Nachteile in unterschiedlichen Aspekten diskutiert und abgewogen. Das anschließend implementierte Konzept arbeitet mit acht zyklisch phasenverschobenen Taktphasen, welche die Position des Datenauges aus der Übertragung bestimmen. Anschließend wird aus der Position des Datenauges die Taktphase zur Rückgewinnung der Bits gewählt, welche am weitesten entfernt von den Transitionen der Daten ist. Das entworfene System wird in einen bestehenden integrierten Bitfehlerratentester implementiert, wodurch eine Charakterisierung des Systems und von optischen Übertragungsstrecken ermöglicht wird. Zur Realisierung wird das System funktional simuliert und auf der Hardware in seiner maximal möglichen Datenrate untersucht. Des Weiteren wird die allgemeine Jittertoleranzfähigkeit bestimmt und in Vergleich zu einer bekannten Spezifikation gesetzt. Zusätzlich wird das System zur Untersuchung von optischen Transceivern verwendet und deren Charakterisierung anhand der Messergebnisse durchgeführt.

Hagen Steinbach

Title of Thesis

Design of an integrated, digital receiver for real-time, wireless, optical communications

Keywords

Clock-Data-Recovery, CDR, optical transmission, OWC, FPGA

Abstract

In this thesis an all-digital clock-data-recovery unit for real-time, wireless, optical communcations is realized in a FPGA. At first three different concepts for realization are presented and their advantages and disadvantages discussed. The implemented concept uses eight cyclic spaced clockphases, which determine the position of the current data eye. Subsequently, the clockphase, which is farthest away from the edges of the data eye, is chosen for the actual data recovery. The designed system will be implemented into an existing integrated bit-error-tester, which is then used for characterization purposes of optical transmissions. In order to realize the system, it will be simulated, tested on hardware to determine the maximum possible datarate und its measured jitter tolerance is put into a comparison to a known standard. At last the system will be used to examine optical transceivers and the gained results are used for their characterization.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

IrDA	Infrared Data Association
ADPLL	All-Digital-Phase-Locked-Loop
APP	Average-Phase-Picking
ASIC	applikationsspezifischer Schaltkreis
BER	Bit-Error-Ratio
BO-DRU	Datenrückgewinnungseinheit mit blinder Überabtastung (engl. blind-oversampling data-recovery-unit)
CDR	Clock-Data-Recovery
CLB	konfigurierbarer Logikblock (engl. configurable-logic-block)
CMT	Taktmanager (engl. clock-management-tile)
CPP	Taktphasenzeiger (engl. clock-phase-pointer)
DCM	digitaler Taktmanager (engl. digital-clock-manager)
DCO	digital kontrollierter Ozsillator (engl. digital controlled oscillator)
DPP	Direct-Phase-Picking
DSP	digitaler Signalprozessor
DUT	Device-Under-Test

FPGA	Field-Programmable-Gate-Array
FSM	endlicher Zustandsautomat (engl. finite-state-machine)
HDL	Hardwarebeschreibungssprache (engl. Hardware Description Language)
IBERT	integrierter Bitfehlerratentester (engl. integrated-bit-error-tester)
IC	integrierter Schaltkreis (engl. integrated ciruit)
IPMS	Fraunhofer Institut für Photonische Mikrosysteme
IR	infrarote Strahlung
LC	Logikzelle (engl. logic-cell)
LD	Laserdiode
LED	Licht emittierende Diode
Li-Fi	Light-Fidelity
LSB	Least Significant Bit
LUT	Look-Up-Table
LVDS	low-voltage-differential-signaling
MCRP	Multi-rotierenden-Taktphasen (engl. Multi-Rotating-Clock-Phases)
MSB	Most Significant Bit
PLL	Phasenregelschleife (engl. Phase-Locked-Loop)
PRBS	pseudozufallgenerierter Bitstrom (engl. pseudo-random bit stream)
RAM	Random Access Memory
SIL-5	System-Interface-Level-5
SNR	Signal-Rausch-Abstand (engl. signal-noise-ratio)

UI	Unit-Intervall
VCO	Voltage-Controlled-Oscillator
VHDL	Very High Speed Integrated Circuit Description Language
Wi-Fi	Wireless-Fidelity
µC	Mikrocontroller

1 Einleitung

Im Zuge der voranschreitenden Digitalisierung in allen Lebensbereichen, gewinnt die Vernetzung technischer Geräte immer mehr an Bedeutung. Begriffe wie *Internet of Things* oder *Internet of Everything* werden häufig in einem solchen Zusammenhang genannt. Hierbei werden vor allem unterschiedliche Anforderungen wie allgemeine Störsicherheit, Verfügbarkeit und hohe Bandbreite an die digitalen Übertragungssysteme gestellt. Im Umgang mit sensiblen Daten gewinnt zudem die Abhörsicherheit der Übertragungen an Bedeutung. Des Weiteren steigt der Bedarf nach kabelungebundenen Kommunikationssystemen zur komfortablen Abdeckung und einfacheren Vernetzung vieler Endgeräte.

In optisch-kabelungebundenen Kommunikationssystemen werden als Übertragungsmedien sichtbares Licht oder infrarote Strahlung verwendet, wodurch sehr hohe Bandbreiten serieller Datenübertragungen zu realisieren sind. Des Weiteren bieten die physikalischen Eigenschaften der Übertragungsmedien eine allgemeine Störsicherheit gegen andere elektromagnetische Interferenzen, was die Kommunikationssysteme für den Einsatz in Industrieumgebungen mit starken Interferenzen, durch beispielsweise bestehende Funknetzwerke, interessant macht. Zusätzlich ist natürliche Abhörsicherheit gewährleistet, da die verwendeten Übertragungsmedien von Objekten blockiert werden. Ein Abgreifen der übertragenen Informationen außerhalb des Raumes wird damit unmöglich und ein Angreifer müsste sich im direkten Übertragungspfad befinden. So können entsprechende optische Kommunikationssysteme beispielsweise zur Herstellung einer abhörsicheren, kabellosen Verbindung eines Endgerätes zum Intranet in einem Unternehmen eingesetzt werden. Zudem ermöglicht ein interoperables, optisches System in Verbindung mit Ethernet und anderen Kommunikationstechnologien und - standards die Realisierung echtzeitfähiger Kommunikationssysteme nach bekannten Standards, welche ebenfalls im industriellen Umfeld gefragt sind. Zur Realisierung echtzeitfähiger Kommunikationssysteme werden geringe Latenzen und die Erhaltung einer kontinuierlichen, zuverlässigen Datenübertragungen an die Sender und Empfänger des Kommunikationssystems gestellt.

Um Interoperabilität in einem echtzeitfähigen Übertragungssystem zu gewährleisten, ist die schnelle Umsetzung und Interpretation der übertragenden Informationen durch digitale Schaltungen notwendig. Hierfür eignet sich die Verwendung eines Field-Programmable-Gate-Arrays (FPGAs) zur unkomplizierten Implementation einer Vielzahl unterschiedlicher digitaler Operationen auf einem Chip. Zudem wird durch einen FPGA eine sehr hohe Parallelisierbarkeit unterschiedlicher Operationen ermöglicht. So kann dieser eine komplett integrierte Lösung für den Empfang, Interpretation und die Weiterverarbeitung der Informationen für mehrere Kommunikationskanäle gleichzeitig bieten und damit geringe Latenzen erreichen und Echtzeitbedingungen erfüllen. Um die gewünschten Funktionen auf einem FPGA zu realisieren, wird die allgemeine Beschreibung der Funktionsweise der Operationen in einer Hardwarebeschreibungssprache (engl. Hardware Description Language) (HDL) notwendig, beispielsweise Very High Speed Integrated Circuit Description Language (VHDL). Zudem lässt sich das System durch die Beschreibung in einer HDL einfach in andere Systeme und Technologien übertragen und bietet eine hohe Flexibilität bei der Neukonfiguration und Anpassung.

In digitalen Übertragungssystemen wird häufig auf das zusätzliche Senden der Taktinformation verzichtet, um die zur Verfügung stehende Bandbreite hauptsächlich nur zum Senden der Informationen zu nutzen. Um einen korrekten und fehlerfreien Kommunikationsbetrieb zu gewährleisten, ist es für den Empfänger notwendig aus dem empfangenen Nutzsignal die Taktinformation zu extrahieren, um die Bits bzw. die Informationen korrekt wiederherzustellen. Dieser Prozess wird durch eine im Empfänger realisierte Clock-Data-Recovery (CDR) übernommen, welche zusätzlich zur Rückgewinnung der Daten ebenfalls die Synchronisation der Daten mit dem Systemtakt des Empfängers übernimmt und für die Weiterverarbeitung im System vorbereitet. Ein bekannter Vertreter einer CDR ist die analoge Phasenregelschleife (engl. Phase-Locked-Loop) (PLL), welche aus dem empfangenen Nutzsignal den Sendetakt auf der Seite des Empfängers rekonstruiert und mit diesem dann die Daten korrekt zurückgewinnt. Zusätzlich zur analogen CDR gibt es Konzepte von komplett-digitalen Bausteinen, welche sich schließlich auf einem FPGA realisieren lassen und sich damit die hohe Integrierbarkeit zu Nutzen machen. Des Weiteren bieten digitale CDRs in der Regel ein stabileres Verhalten und kürzere Einrastzeiten auf den Datenstrom, was eine schnelle Rückgewinnung der Informationen ermöglicht.

Durch eine CDR in Verbindung mit einem Bitfehlerratentester ist zudem die Charakterisierung von digitalen Übertragungssystemen möglich. Insbesondere lassen sich äußerliche Einflüsse und die Performance in unterschiedlichen Konfigurationen optisch-

kabelungebundener Kommunikationssysteme bewerten. Möglich zu untersuchende Aspekte sind die Performanceunterschiede bei unterschiedlicher Ausrichtung von Sender und Empfänger oder unterschiedlicher Sendeleistung. Ebenso kann der Einfluss von anderen Lichtquellen, Reflexionen und teilweises blockieren des Sende- und Empfangsbereich auf die Performance des optischen Kommunikationssystems bestimmt werden.

2 Problemstellung

Das Ziel dieser Arbeit ist die Entwicklung einer komplett-digitalen CDR für optische, kabelungebundene und echtzeitfähige Kommunikationssysteme. Es erfolgt die Erarbeitung von drei Lösungskonzepten zur möglichen Realisierung der digitalen CDR. Die erarbeiteten Konzepte werden auf unterschiedliche Aspekte wie maximal erreichbare Datenrate, erwartbare Performance und Komplexität bewertet. Um Portierbarkeit und Flexibilität im Entwicklungsprozess und späteren Einsatz zu gewährleisten, wird die zu entwickelnde CDR in VHDL beschrieben, funktional simuliert und auf einem FPGA implementiert. Anschließend erfolgt die Integration des ausgewählten Konzeptes in einen bestehenden Bitfehlerraten-Tester, um eine Charakterisierung der CDR und des optischen Übertragungssystems nach unterschiedlichen Aspekten durchzuführen.

Diese Arbeit wurde am Fraunhofer Institut für Photonische Mikrosysteme (IPMS) in Dresden angefertigt. Die genutzten Technologien des Fraunhofer IPMS sind in der Arbeit kenntlich gemacht.

3 Stand der Technik

3.1 Field Programmable Gate Array

3.1.1 Allgemein

FPGAs sind digitale, integrierte Schaltkreise (engl. integrated ciruits) (ICs) mit konfigurierbaren Logikblöcken, welche miteinander frei verbunden werden können. Dadurch wird eine große Vielfalt an Einsatzmöglichkeiten eröffnet, da FPGAs nach Belieben von Entwicklern programmiert werden können, um unterschiedliche Aufgaben und Operationen durchzuführen. [1, S. 1]

Die zu implementierenden logischen Funktionen werden mithilfe von HDLs auf Register-Transfer-Level beschrieben. Zu den bekanntesten Vertretern der HDLs zählen VHDL und Verilog. Diese Funktionsbeschreibungen lassen sich dann mithilfe eines Synthesetools in eine für den FPGA lesbare Konfigurationsdatei übersetzen und auf den FPGA laden. Die Funktionen werden im FPGA direkt in Hardware implementiert, was eine hohe Parallelisierbarkeit unterschiedlicher Prozesse und Operationen ermöglicht.

Allgemein können digitale Systeme im Entwurf und ihrer Implementation in mehrere Gruppen eingeordnet werden. So lassen sich digitale Systeme mithilfe von diskreten Elementen wie einzelnen Logikgattern, applikationsspezifische integrierte Schaltkreise (ASICs), FPGAs oder komplexeren Prozessorsystemen, zu denen beispielsweise Mikrocontroller (µC), digitale Signalprozessoren (DSPs) oder konsumorientierte Prozessoren gehören, implementieren. Der Aufbau logischer Funktionen durch diskrete Elemente wurde lange Zeit verfolgt, ist jedoch in seinem physischen Platzbedarf sehr hoch. Zudem kann die Verdrahtung der Elemente untereinander zu Schwierigkeiten führen und das Design ist statisch, nachdem es aufgebaut worden ist. Dagegen besitzen ASICs einen geringen Platzbedarf und bieten bei den Möglichkeiten der Verdrahtung sehr hohe Freiheiten. Sie sind jedoch nicht rekonfigurierbar und werden nur für spezielle Anwendungen entwickelt,

da der Entwicklungsprozess durch viele Tests und Simulationen sehr zeitaufwendig ist. Des Weiteren sind ASICs nur ökonomisch sinnvoll, wenn die gewünschte Funktion durch keine Alternative lösbar ist oder sehr hohe Stückzahlen benötigt werden. FPGAs, DSPs und µC teilen sich gewisse Charakteristiken. So sind sie rekonfigurierbar, kompakt und je nach Anwendung bieten sie eine sehr gute Alternative zu einem ASIC. Der µC und DSP besitzen im Gegensatz zum FPGA einen vorgeschriebenen Befehlssatz, was die Vielfalt der umsetzbaren Aufgaben beschränkt. [2, 1] So lassen sich auf einem FPGA viele Aufgaben parallel ausführen, während der µC und der DSP sein Programm sequentiell durchläuft.

Ein FPGA besitzt eine feste physische Größe und hat frei konfigurierbare, interne Verdrahtungsressourcen, welche die flexible Verbindung von digitaler Logik in einem hohen Maß ermöglicht. Durch die hohe Flexibilität werden viele Einsatzmöglichkeiten eröffnet. So kann ein FPGA gleichzeitig sowohl typische Funktionen eines DSPs als auch eines µCs übernehmen, was ihn zu einer guten Alternative komplexerer Systeme macht, in denen mehrere ICs benötigt werden. So sind heute FPGAs in konsumorientierten Geräten wie Mobiltelefonen, aber auch in der Industrie als physische Schnittstellen zur Datenverarbeitung und -übertragung wiederzufinden. Des Weiteren werden sie im Entwurfsverfahren von ASICs zur einfachen und zeitsparenden Verifikation von Prototypen eingesetzt.

3.1.2 Aufbau und Ressourcen

Logikfunktionen

Ein FPGA besteht aus einer Matrix von mehreren frei konfigurierbaren Logikblöcken (engl. configurable-logic-blocks) (CLBs), welche durch ebenfalls frei konfigurierbare, elektrische Leitungen miteinander verbunden sind. Ein CLB beinhaltet mehrere *Slices* und Logikzellen (engl. logic-cells) (LCs). Die LC ist das Kernelement eines FPGAs und unterscheidet sich im Aufbau je nach Hersteller und Modell des FPGAs. Der prinzipielle Aufbau einer LC ist in der Abbildung 3.1 dargestellt. Die logischen Funktionen werden in der Regel durch Look-Up-Tables (LUTs) gelöst, welche ebenso als Random-Access-Memory (RAM) oder ein Schieberegister konfiguriert werden können. Des Weiteren befindet sich in der LC ein Multiplexer (MUX) und ein Register, welches als Flipflop konfiguriert werden kann, was eine Vielzahl von logischen Operationen innerhalb der LC ermöglicht.

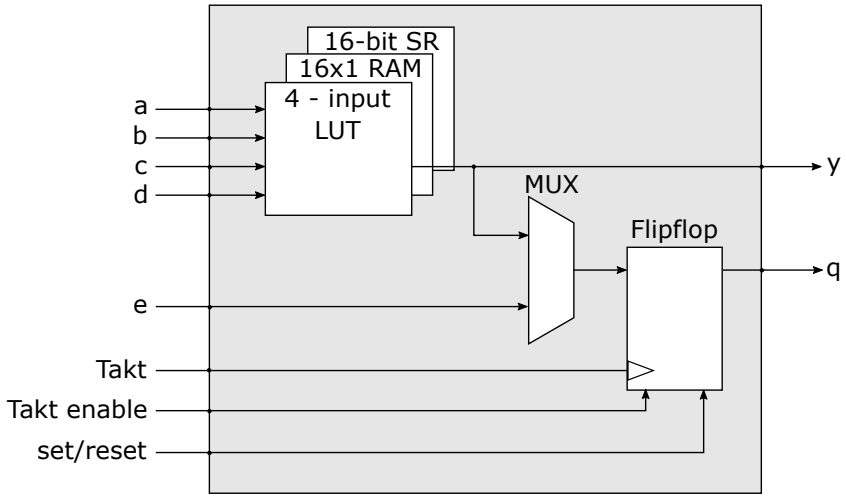


Abbildung 3.1: Vereinfachtes Blockschaltbild einer LC, nach [1, S. 74]

Jedes Slice besteht herkömmlicherweise aus zwei LCs und mehrere Slices formen einen CLB, wobei die Anzahl je nach Modell und Hersteller variieren kann. Die übliche Anzahl ist vier. Innerhalb des CLBs lassen sich die Verbindungen der Slices und LCs frei konfigurieren und es existieren weitere, direkte Verbindungen zwischen benachbarten CLBs, welche sehr geringe Signallaufzeiten zwischen den unterschiedlichen Logikbausteinen ermöglichen. Die Abbildung 3.2 zeigt einen kleinen Ausschnitt aus der Bausteinmatrix eines FPGAs.

Zusätzlich können weitere dedizierte Blöcke unterschiedliche komplexe Funktionen bereitstellen, wie etwa Multiplikationen. Manche Funktionen benötigen sehr viele Ressourcen, sollten diese durch CLBs realisiert werden, und durch den Einsatz dedizierter Blöcke lässt sich der Platzbedarf optimieren. Des Weiteren werden zum Teil dedizierte Speicher in Form von Random Access Memory (RAM)-Blöcken auf einem FPGA bereitgestellt. Die Abbildung 3.3 zeigt eine beispielhafte Anordnung der unterschiedlichen Funktionsblöcke in einem FPGA. Ebenso können ganze Prozessorkerne und µCs in einem FPGA implementiert werden, welche als direkte Hardwarekerne vorliegen können, sogenannte *Hardcores*, oder mittels der programmierbaren Logikblöcke realisiert werden, sogenannte *Softcores*. Dies ermöglicht die Implementierung von Funktionen in Soft- und Hardware in einem Chip. Hardcores lassen sich grundsätzlich mit einem höheren Takt versorgen und bieten eine höhere Bandbreite als Softcores. [1, S. 57 ff.]

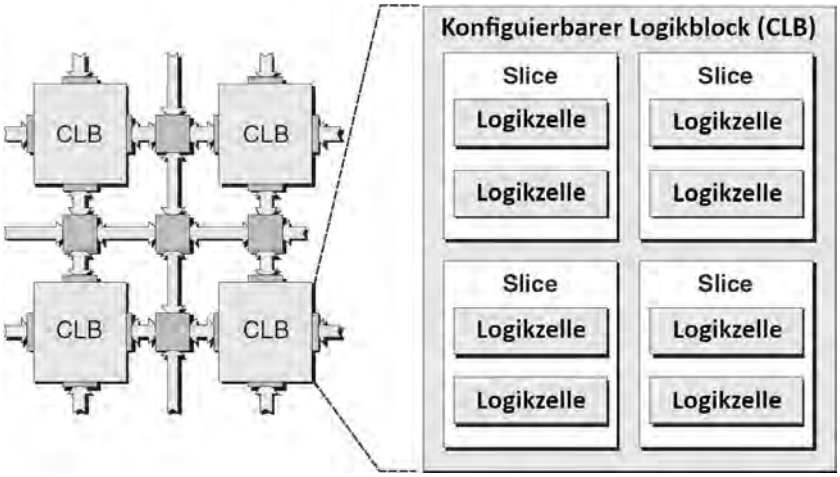


Abbildung 3.2: CLB-Matrix mit Zwischenverbindungen und schematischer Darstellung eines CLBs, modifiziert nach [1, S. 76]

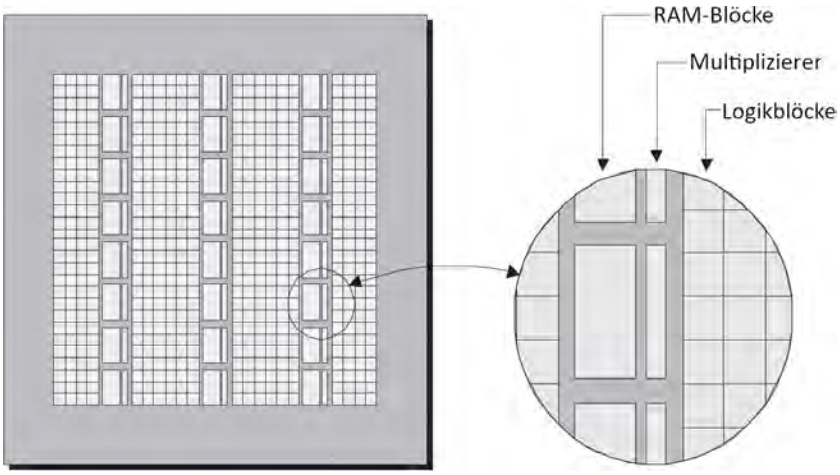


Abbildung 3.3: Schematische Darstellung der verschiedenen Funktionsblöcke in einem FPGA, modifiziert nach [1, S. 79]

Taktverbindungen

Alle synchronen Logikelemente in einem FPGA werden durch ein Taktsignal getrieben, wie z.B. ein Flipflop. Dieses Taktsignal wird häufig über spezielle Pins in den FPGA geführt und dann im Chip über Taktbäume oder Taktnetze verteilt und mit den gewünschten Registern verbunden. Durch die Struktur der Taktbäume und -netze wird gewährleistet, dass an allen Blöcken das Taktsignal zur nahezu gleichen Zeit erscheint und es zu einem ausreichend kleinem Versatz durch unterschiedliche Signallaufzeiten kommt, wodurch ein synchroner Betrieb gewährleistet werden kann. Der Takt wird im FPGA über ein spezielles, separates Netzwerk getrieben und steht in keiner Verbindung zu dem programmierbaren Netzwerk zwischen den CLBs. Die Abbildung 3.4 zeigt eine stark vereinfachte Version eines Taktbaumes in H-Struktur in einem FPGA. Häufig gibt es mehrere parallele Taktbäume oder -netze, die es ermöglichen unterschiedliche Bereiche im FPGA auf unterschiedlichen Taktdomänen zu betreiben.

Über Taktmanager lassen sich auf dem FPGA unterschiedliche Takte über einen Referenztakt erzeugen und über das Verteilungsnetz an die gewünschten Register bringen. Diese Taktmanager sind dedizierte Funktionsblöcke auf dem FPGA in Form von PLLs und digitalen Taktmanagern (engl. digital-clock-manager) (DCMs). Ist der Taktmanager mithilfe eines Regelkreises realisiert, lässt sich zusätzlich der Jitter des Ausgangstaktes weitestgehend minimieren. [1, S. 84 ff.]

Durch den allgemeinen Aufbau der FPGAs ist auf logische Schnittstellen zwischen den Taktbäumen zu verzichten, da dies zu langen Signallaufzeiten führt und damit die allgemeine Performance des Systems senkt.

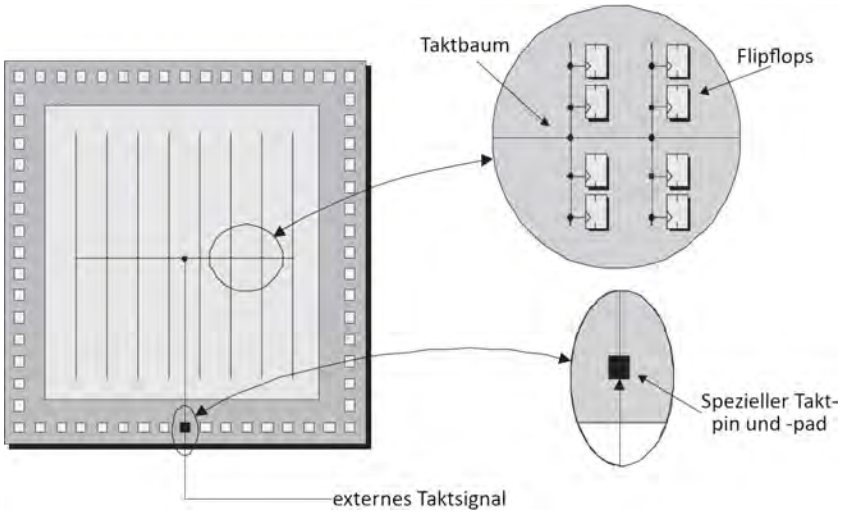


Abbildung 3.4: Taktbaum in einem FPGA, modifiziert nach [1, S. 84]

3.1.3 Entwicklung der FPGAs

Der erste kommerziell erhältliche FPGA XC2064 wurde von der Firma Xilinx Inc. im Jahre 1985 vorgestellt [1, S. 50] und beinhaltet 64 CLBs, die einem Äquivalent von bis zu 1000 Logikgattern entsprechen, und 58 Ein- und Ausgangspins. Ein CLB des XC2064 besteht aus einer LC, welche eine LUT mit vier Eingängen und ein Flipflop besitzt. Der Chip kann je nach Modell der FPGA-Familie von 33 MHz bis zu 100 MHz getaktet werden [3]. Die Spartan-6 Familie von Xilinx Inc. wurde im Jahr 2009 vorgestellt und beinhaltet je nach Modell 3480 bis zu 147443 LCs. Jeder CLB besteht aus zwei Slices mit jeweils vier 6-Eingangs-LUTs und acht Flipflops [4]. Die CLBs lassen sich mit einer Frequenz von bis zu 400 MHz betreiben [5]. Dagegen besitzen die neuesten Serie-7-FPGAs von Xilinx mindestens 6000 LCs, können aber bis zu 1,955 Millionen LCs aufweisen [6]. Die Komplexität der FPGAs hat durch die Verbesserung der Halbleiterverarbeitung im Laufe der Zeit stark zugenommen, was höhere Flexibiliät und höhere Integration von Systemen mit sich bringt, wie beispielsweise die Implementation von Softcores und Hardwarefunktionen auf einem Chip.

3.2 Jitter

3.2.1 Taktjitter

Mikroprozessoren und viele digitale Schaltungen benötigen Taktsignale zur zeitlichen Koordinierung von Operationen. Da Taktsignale durch physikalische Komponenten erzeugt werden gibt es eine Ungenauigkeit und Zufälligkeit in ihren Prozessen, welche sich in zeitlichen Schwankungen im Takt und ihren Taktflanken äußern. Diese Schwankungen werden als Jitter bezeichnet und können in taktsynchronen Systemen zu Störungen in der Operationsweise oder bis zum gesamten Systemausfall führen. Der Effekt von Jitter ist in der Regel bei geringen Taktfrequenzen unbedeutend, da dieser im Vergleich zur Taktperiode sehr gering ist. Mit zunehmender Taktfrequenz steigt jedoch der Einfluss des Jitters auf die Periode und damit erhöht sich die Gefahr von Störungen und einem fehlerhaften Betrieb. [7, S. 2-3]

3.2.2 Datenjitter

Datenjitter tritt an Flipflops auf, wird durch den allgemeinen Taktjitter hervorgerufen und ist nur an den Übergängen zwischen den Datenpegeln zu erkennen. In Abbildung 3.5 ist die Einsynchronisierung von asynchronen Eingangsdaten zum Systemtakt dargestellt. Es wird ein D-Flipflop betrachtet mit einem Dateneingang D, dem Takteingang CK und dem Ausgang Q. D wird bei steigender Taktflanke von CK abgetastet und Q so mit dem Takt synchronisiert. Der Jitter des Taktes CK nimmt dann direkten Einfluss auf den zeitlichen Verlauf des Ausgangssignales Q. In der Takt- und Datenrückgewinnung wird die Dauer, die ein Datenbit anliegt, als Unit-Intervall (UI) beschrieben. Bei der Betrachtung eines idealen Taktes ist der Datenstrom ebenfalls ideal und besitzt ein konstantes UI, welches der Periode des Taktes entspricht. Bei einem nicht idealen Takt weicht das UI des Datenstromes leicht vom nominalen, idealen UI ab. [7, S. 5]

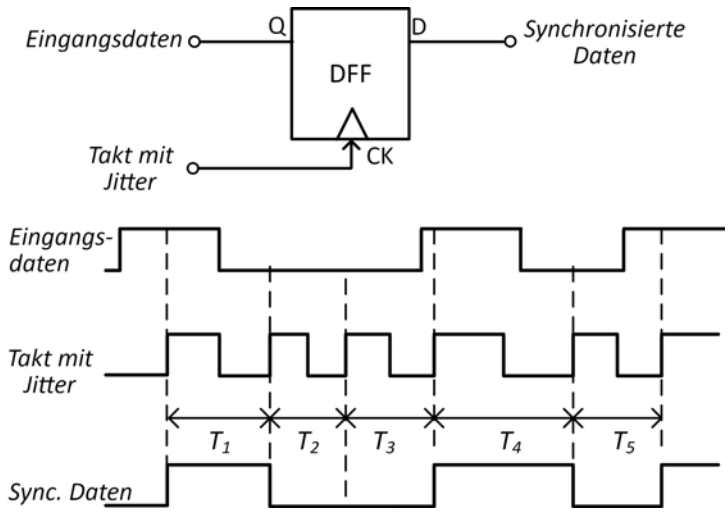


Abbildung 3.5: Eingangsdaten werden mit jitterbehafteten Takt einsynchronisiert, modifiziert nach [7, S. 6]

Augendiagramm

Durch die Aufnahme eines Augendiagramms wird Jitter und die allgemeine Qualität eines Datensignals in Datenübertragungen mess- und charakterisierbar. Hierzu wird ein jitterbehafteter, zufälliger Datenstrom betrachtet, welcher in gleichmäßigen Abständen abgetastet wird. Durch Übereinanderlegen der unterschiedlichen Daten-UIs lässt sich dann ein Augendiagramm erzeugen. Die Abbildung 3.6 zeigt ein Augendiagramm einer optischen Datenübertragung mit den wichtigsten Hauptparametern. Das Augendiagramm gibt Aufschluss über die Zeitpunkte an denen die mögliche Datenpegel „0“ und „1“ am Weitesten voneinander entfernt sind, was durch die maximale vertikale Öffnung des Auges gezeigt, während auftretender Jitter das Auge in horizontaler Richtung schließt. Die vertikale und horizontale Öffnung des Auges gibt Aufschluss über den optimalen Zeitpunkt einer ungestörten Abtastung des Datenbits, welcher in der Regel in der Mitte des Auges liegt. Sollte Rauschen, Interferenz, Dämpfung, Jitter oder andere Störeinflüsse in einem System bedeutend auftreten, kann sich das Auge komplett schließen. Dann sind die Signalpegel nicht mehr eindeutig zu unterscheiden und eine fehlerfreie Detektion ist nicht mehr gewährleistet. [7, 8]

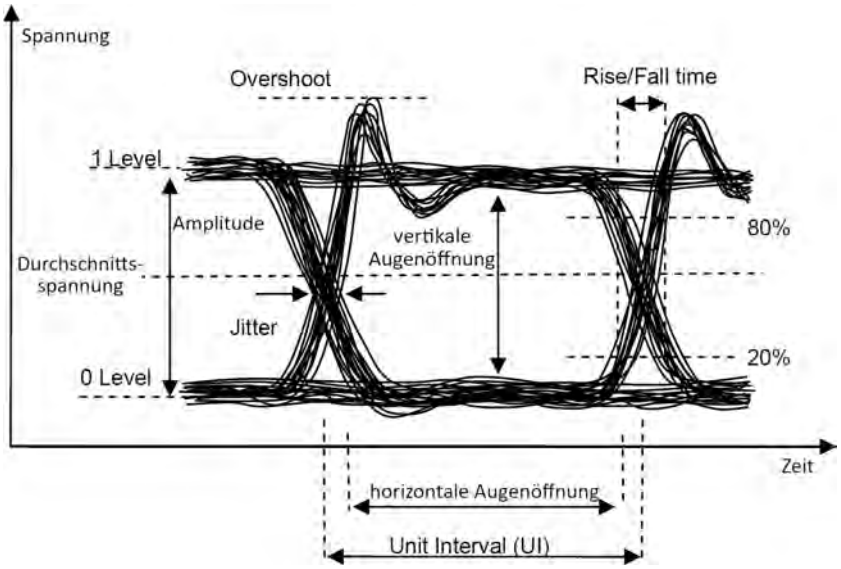


Abbildung 3.6: Darstellung eines Augendiagramms, modifiziert nach [8, S. 81]

3.2.3 Kategorisierung von Jitter

Absoluter Jitter

Der absolute Jitter (engl. total jitter), der in einem System auftritt, setzt sich aus einem zufälligen Anteil und einem deterministischen Teil zusammen. In den folgenden Abschnitten werden diese beiden Teilkomponenten als zufälliger Jitter und deterministischer Jitter näher beschrieben. [7, S. 36]

Zufälliger Jitter

Der zufällige Jitter fasst alle Jitterkomponenten zusammen, deren Histogramm und Dichtefunktion unbeschränkt ist. In elektronischen Systemen wird zufälliger Jitter durch beispielsweise thermisches Rauschen und durch nicht exakt verlaufende Schaltvorgänge von Gattern hervorgerufen. Im Jitterhistogramm stellt sich zufälliger Jitter als eine Normalverteilung dar. Die Abbildung 3.7 zeigt ein erzeugtes Augendiagramm und das dazugehörige Jitterhistogramm eines Datenstrom mit zufälligem Jitter. [7, S. 35]

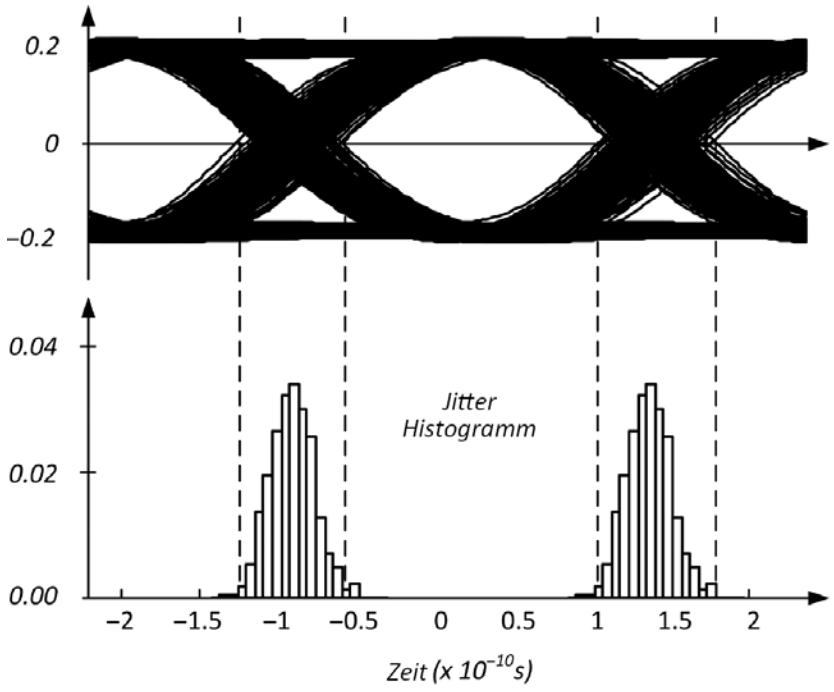


Abbildung 3.7: Darstellung eines Augendiagramms mit zugehörigem Jitterhistogramm bei einer Datenübertragung mit zufälligem Jitter, modifiziert nach [7, S. 8]

Deterministischer Jitter

Der deterministische Jitter umfasst alle Jitterkomponenten, deren Histogramm und Dichtefunktion beschränkt ist. In der Abbildung 3.8 ist das Augendiagramm eines Datenstromes mit deterministischen Jitter und dem dazugehörigen Histogramm dargestellt. Die gewöhnlichen Ursachen für deterministischen Jitter sind die Modulationen des Taktes durch Rauschen der Versorgungsspannung, Übersprechen anderer Signale, Bandbreitenlimitation des Kanals und weitere. Abhängig von seinem Ursprungsmechanismus können weitere Unterarten des Jitters definiert werden, wie beispielsweise der Duty-Cycle-Verzerrung Jitter. Dieser entsteht durch die Asymmetrie des Taktes, wenn in einem System die steigende und fallende Taktflanke für Operationen genutzt werden. Ebenso kann deterministischer Jitter durch die gewählte Form der Datenübertragung und Kodierung der einzelnen Bits und deren Übertragungsverhalten hervorgerufen werden. [7, S. 36]

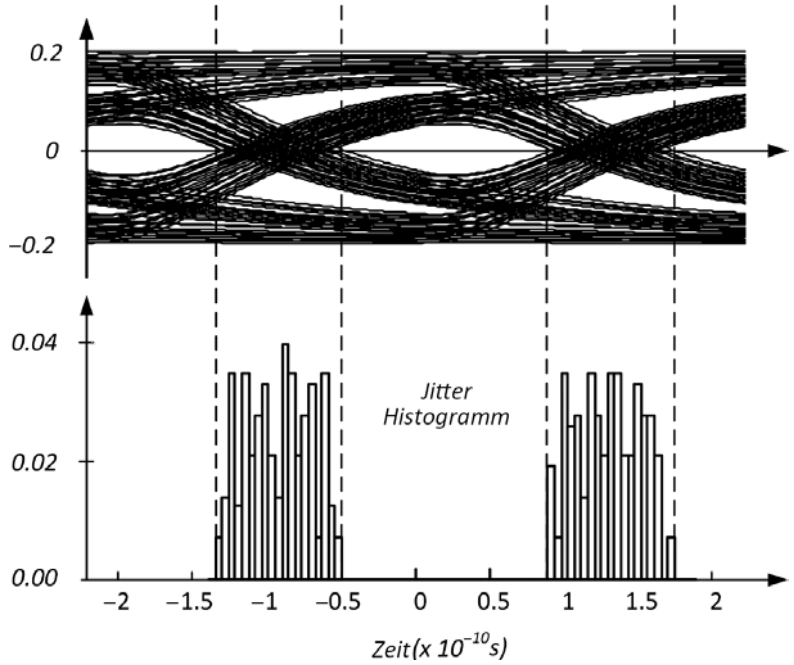


Abbildung 3.8: Darstellung eines Augendiagramms mit zugehörigem Jitterhistogramm bei einer Datenübertragung mit deterministischem Jitter, modifiziert nach [7, S. 8]

3.2.4 Bedeutung von Jitter in digitalen Übertragungssystemen

Ein digitales Kommunikationssystem, bestehend aus Sender, Kanal und Empfänger, ist in Abbildung 3.9 gezeigt. Alle drei Komponenten sind für Jitter in einem Kommunikationssystem verantwortlich und fügen in der Übertragung selbst Jitter hinzu. Durch den realen, jitterbehafteten Takt des Senders besitzt die Datenübertragung direkt nach der Übermittlung an den Kanal Datenjitter. Des Weiteren bewirkt die physikalische Tiefpass- und Speichercharakteristik des Kanals, dass der Signalpuls des übertragenden Bits nicht sofort nach dem Senden verschwindet und die Signalpegel des nächsten und nachfolgenden Bits beeinflusst. Dieser Effekt wird als Intersymbolinterferenz bezeichnet und wirkt sich als deterministischer Jitter auf die Übertragung aus. Ein Entzerrer am vorderen Ende des Empfängers kann den Empfangsjitter reduzieren, welcher durch den Kanal und den Sender hervorgerufen wurde. Anschließend werden die entzerrten Daten der Clock-Data-Recovery (CDR) übermittelt, welche die Frequenz und Phase eines internen Taktes zur Rückgewinnung und Synchronisation der Daten mit dem Systemtakt des Empfängers kontrolliert. Dabei ist die CDR ebenfalls von Jitter betroffen, besitzt jedoch eine begrenzte Jitterfilterungscharakteristik. [7, S. 13-14]

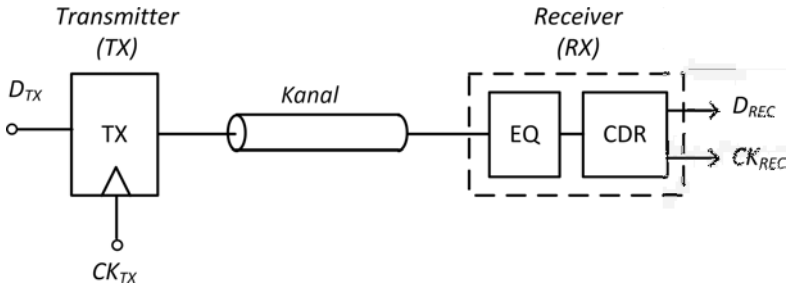


Abbildung 3.9: Blockdiagramm eines Kommunikationssystems, modifiziert nach [7, S. 13]

Jitter in synchronen digitalen Systemen

In modernen integrierten Schaltungen ist der Großteil der digitalen Komponenten zur synchronen Operation taktflankengesteuert, womit auftretender Jitter direkten Einfluss auf die Funktionsweise des Systems hat. Das zentrale Element dieser Systeme ist das Flipflop, welches als Blockschaltbild mit seinem Signalverhalten in Abbildung 3.10 gezeigt ist. Das ideale Flipflop schaltet sofort bei Erscheinen der Taktflanke CK das anliegende Eingangssignal X auf den Ausgang Y. Das Verhalten des realen Flipflops weicht hiervon ab. So muss das Eingangssignal zunächst für eine bestimme Zeitdauer τ_{su} (*setup-time*) vor der Taktflanke stabil anliegen und darf sich nach dem Erscheinen der Taktflanke nicht unmittelbar ändern für die Zeitdauer τ_{ho} (*hold-time*). Zusätzlich wird durch den internen Schaltungsprozess des Flipflops eine weitere Verzögerung hinzugefügt, welche als *Clock-to-Q* τ_{cq} bezeichnet wird. [7, S. 111]

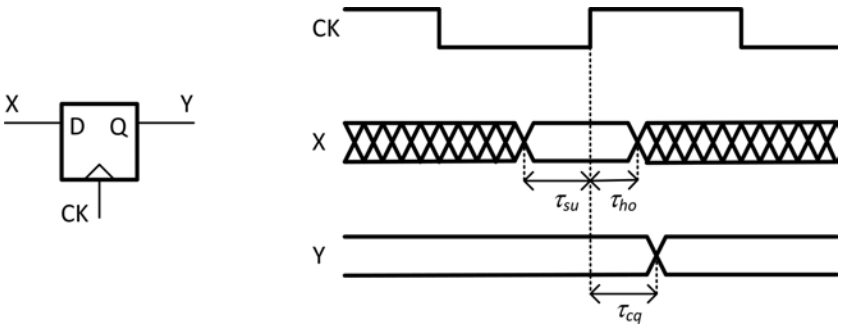


Abbildung 3.10: Blockschaltbild und Signalverhalten eines D-Flipflops [7, S. 112]

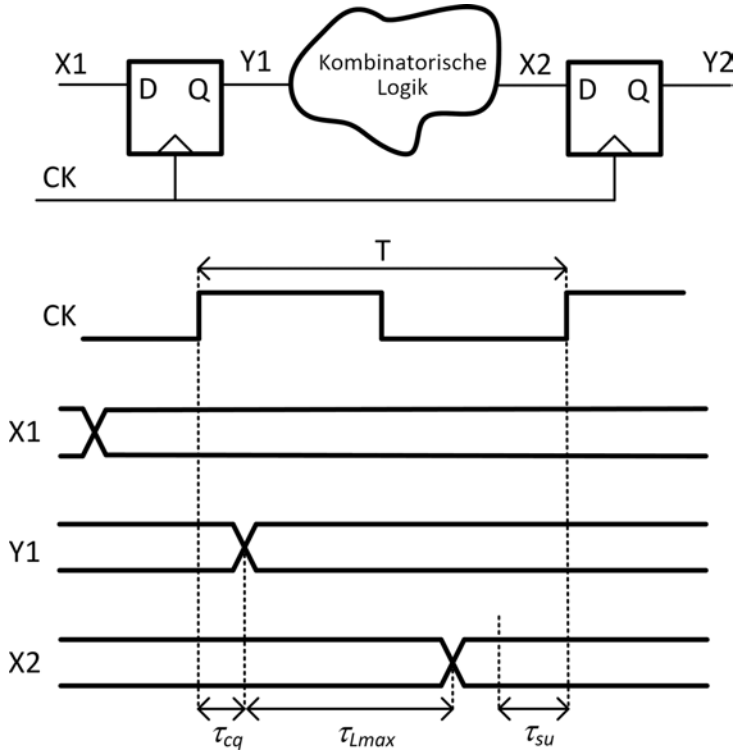


Abbildung 3.11: Blockschaltbild eines taktsynchronen Systems mit Signalverlauf [7, S. 112]

Die Abbildung 3.11 zeigt eine typische Konfiguration eines digitalen, taktsynchronen, flankengesteuerten Designs. Ein Flipflop erfasst das Eingangssignal X1 und gibt dieses bei steigender Taktflanke nach der Zeit τ_{cq} an Y1 aus. Anschließend wird Y1 durch die kombinatorische Logik verarbeitet und das Ergebnis X2 erzeugt und durch die Gatter- und Signallaufzeiten in der kombinatorischen Logik werden weitere Verzögerungen τ_{L} addiert. Das Ergebnis X2 muss mindestens τ_{su} lang stabil anliegen, damit es mit der nächsten steigenden Taktflanke durch das zweite Flipflop erfasst wird. Der *Worst-Case* tritt auf, wenn die Verzögerung durch die kombinatorische Logik maximal wird τ_{Lmax}. Aus den Gegebenheiten ergibt sich für die Taktperiode T des Systems folgender Zusammenhang:

$$T > \tau_{\mathrm{cq}} + \tau_{\mathrm{Lmax}} + \tau_{\mathrm{su}} \tag{3.1}$$

Damit weiterhin die *hold-time* τ_{ho} am zweiten Flipflop nicht verletzt wird, ist die minimale Verzögerung durch die kombinatorische Logik τ_{Lmin} ausschlaggebend. Es ergibt sich folgender Zusammenhang:

$$\tau_{\mathrm{Lmin}} > \tau_{\mathrm{ho}} - \tau_{\mathrm{cq}} \tag{3.2}$$

Zusätzlich erreichen durch auftretenden Taktversatz die Taktflanken in einem realen System nicht exakt zum gleichen Zeitpunkt alle synchronen Komponenten. Dies tritt vor allem in großen digitalen Kernen auf, in denen die Taktbäume und -netze sehr groß und komplex sind. Die Abbildung 3.12 zeigt auftretenden Taktversatz zwischen zwei Flipflops. Der Taktversatz τ_{sk} zwischen beiden Flipflops ergibt sich aus der Differenz der Zeitpunkte, an denen die Taktflanken die Flipflops erreichen.

$$\tau_{\mathrm{sk}} = \tau_{\mathrm{d2}} - \tau_{\mathrm{d1}} \tag{3.3}$$

Des Weiteren beeinflusst Taktjitter die maximal mögliche Frequenz des digitalen Systems und kann zur Verletzung der *setup-time* τ_{su} führen. Da Taktjitter sich nicht als Taktversatz äußert und die synchronen Komponenten gleichermaßen betrifft, ist die Betrachtung des kombinatorischen Pfades und der *hold-time* τ_{ho} irrelevant. Unter Betrachtung des Taktversatzes τ_{sk} und des Taktjitters τ_{jitter} ändern sich die Gleichung 3.1 und 3.2 wie folgt:

$$T > \tau_{\mathrm{cq}} + \tau_{\mathrm{Lmax}} + \tau_{\mathrm{su}} - \tau_{\mathrm{sk}} + \tau_{\mathrm{jitter}} \tag{3.4}$$

$$\tau_{\mathrm{Lmin}} > \tau_{\mathrm{ho}} - \tau_{\mathrm{cq}} + \tau_{\mathrm{sk}} \tag{3.5}$$

Kommt es zur Verletzung einer dieser Zeitbeschränkungen ist die korrekte Funktionsweise des Flipflops nicht mehr gewährleistet und das Ausgangssignal kann dann nicht vorhergesagt werden. In seltenen Fällen kann das Flipflop in einen metastabilen Zustand versetzt werden. Anschließend nimmt nach der Erholungszeit τ_{recover} das Ausgangssignal einen festen Zustand an und das Flipflop kann wieder normal beschaltet werden. [7, S. 112-114]

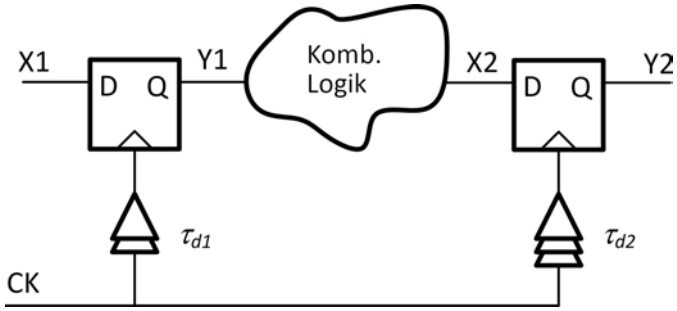


Abbildung 3.12: Taktversatz in einem synchronen Design [7, S. 113]

3.3 Digitale Übertragungssysteme

3.3.1 Allgemein

Übertragungssysteme in der digitalen Kommunikationstechnik besitzen prinzipiell den gleichen Aufbau und die gleiche Funktionsweise. So werden analoge Sprachsignale, Bildsignale, Dateien oder andere Formen von Informationen für die Übertragung als binäre Sequenzen betrachtet. Nachdem die zu sendende Information in eine binäre Sequenz überführt ist, muss diese für den gewählten physischen Übertragungskanal aufbereitet werden. Beim physischen Übertragungskanal kann es sich um ein einfaches Kabel, verzweigte Kabelpaare, optische Glasfaser, elektromagnetische oder kabelungebundene optische Übertragungen durch den Raum handeln. Demnach sind digitale Kommunikationssysteme Systeme, die eine digitale, binäre Sequenz als Schnittstelle zwischen einem Sender, der Datenquelle, und einem Empfänger, der Datensenke, besitzen. [9, S. 2]

Die Abbildung 3.13 zeigt das Modell eines digitalen Datenübertragungssystems. Der Quellkodierer digitalisiert die zu sendende Information und überführt diese in eine binäre Sequenz, welche vom Kanalkodierer für die Übertragung über den Kanal aufbereitet wird. Nach der Übertragung über den Kanal erreicht die Nachricht die Empfängerseite und der Kanaldekodierer gewinnt die ursprüngliche digitale Sequenz zurück. Anschließend gewinnt der Quelldekodierer die eigentliche Information aus dem digitalen Signal zurück. [9, S. 2]

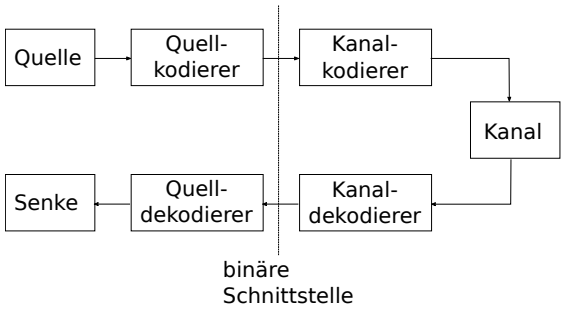


Abbildung 3.13: Blockschaltbild eines digitalen Übertragungskanals, nach [9, S. 2]

Es gibt unterschiedliche Gründe für die Verwendung von digitaler Übertragung im Vergleich zur analogen Übertragung. So ist digitale Hardware zuverlässiger, stark miniaturisierbar und besser integrierbar. Weiterhin vereinfachen standardisierte binäre Schnittstellen zwischen Kanalkodierung und Quellkodierung die Implementation von komplexeren

Systemen. [9, S. 3] Im folgenden Abschnitt wird auf gewisse Eigenschaften von digitalen Übertragungssystemen spezieller eingegangen.

3.3.2 Nachrichtenquellen

Die Nachrichten- bzw. Datenquelle kann Informationen in Form von diskreten Symbolen erzeugen, wie zum Beispiel Buchstaben eines Alphabetes oder binäre Symbole einer Datei. Alternativ kann sie ebenso ein analoges Signal erzeugen, beispielsweise ein Sprachsignal eines Mikrofons oder einen Ausgangspegel eines Sensors. Generell wird der Ausgang der Quelle als zufälliger Prozess modelliert. Das Ziel ist es den Ausgang der Datenquelle in ein Signal zu formen, welches sich für die Übertragung eignet. Im Allgemeinen werden Kommunikationssysteme so entworfen, dass sie mit einer Vielzahl möglicher Ausgänge der Quelle funktionieren. [9, S. 5]

3.3.3 Kanal

Allgemein

Der Kanal eines Kommunikationssystems ist das Verbindungselement zwischen der Datenquelle und Datensenke. Dieser ist normalerweise als gegeben definiert und der Entwickler hat wenig Einfluss auf den Kanal. Notwendige Verstärker, Antennen, Laser und andere werden für die Erzeugung der gewünschten Wellenformen als Teil des Kanals betrachtet. [9, S. 7-8]

Die digitale Sequenz der Quelle wird im Kanalkodierer zur Übertragung auf dem Kanal in entsprechende Signalformen in Form von elektrischen Pulsen, Wellen oder optischen Signalen gebracht. Zum Erzeugen der reinen Signalform gibt es unterschiedliche Herangehensweisen. So ist im binären Fall die einfachste Kodierung das An- und Ausschalten des Signals, wobei eine „1“ bei aktivem Puls übertragen wird und eine „0“ bei ausgeschaltetem Puls. Die gewählte Pulsform der Informationsübertragung wird als Leitungscode bezeichnet. Jeder Leitungscode lässt sich nach unterschiedlichen Kriterien beurteilen und weist unterschiedliche Vor- und Nachteile auf. So sollte er eine möglichst geringe Bandbreite belegen, leistungseffizient sein und kann es ermöglichen, Fehler zu erkennen und zu korrigieren. Weiterhin ist es von Vorteil, wenn das übertragene Signal keinen Gleichanteil aufweist und die Taktinformation zur Rückgewinnung der Daten direkt aus dem Signal ermittelbar ist. [10, S. 327-329]

Die Kanalkodierung und -dekodierung wird normalerweise als Modulation und Demodulation bezeichnet. Dieser Begriff stammt ursprünglich aus der analogen Kommunikation, jedoch beschreibt dieser in der digitalen Kommunikation einen ähnlichen Vorgang. [9, S. 7]

Rauschen

Ein Kanal wird in der Regel über seine möglichen Ein- und Ausgänge definiert. Die Abbildung 3.14 zeigt das Modell eines realen Kanals mit Rauschen. So liegt am Eingang des Kanals das Signal $X(t)$ an, zu dem das Rauschsignal $Z(t)$ bei der Übertragung aufaddiert wird. Dies führt zur Bildung des Ausgangssignales $Y(t) = X(t) + Z(t)$. Dabei ist zu beachten, dass das Rauschen unabhängig vom Eingangsignals ist, womit die Gleichung $Z(t) = Y(t) - X(t)$ ungültig ist. [9, S. 8]

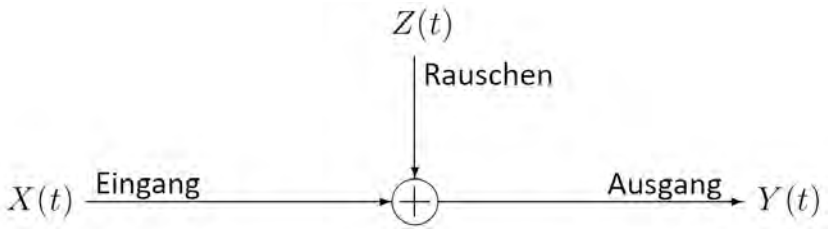


Abbildung 3.14: Modell eines verrauschten Kanals, modifiziert nach [9, S. 8]

3.3.4 Charakterisierung von digitalen Übertragungssystemen

Um eine möglichst fehlerfreie Kommunikation zu erreichen, werden in der analogen Kommunikationstechnik Signale mit einem hohen Signal-Rausch-Abstand (engl. signal-noise-ratio) (SNR) übertragen, damit das Signal am Empfänger möglichst exakt reproduziert werden kann. Das SNR gibt hierbei Aufschluss über die allgemeine Qualität des Systems. Dagegen ist in der digitalen Kommunikation das Ziel des Empfängers das eigentliche Bit, die Information zu erkennen und nicht die übertragende Signalform zu reproduzieren. Somit beruht der Empfang der Information auf einem Entscheidungsprozess und die Wahrscheinlichkeit, dass bei diesem Entscheidungsvorgang ein Fehler auftritt, liefert eine gute Aussage über die Gesamtqualität des Systems und wird als Bit-Error-Ratio (BER) bezeichnet. [10, S. 506]

Die Messung des BERs lässt sich durch die Übertragung einer vorher bekannten Bitfolge bestimmen, welche auf der Empfängerseite wiederhergestellt wird. Das BER bildet sich dann über das Verhältnis der fehlerhaft empfangenen Bits und der insgesamt übertragenen Bits. Jitter, Signalstärke, gewählter Leitungscode und anderweitige Umwelteinflüsse und Störungen können das BER maßgeblich beeinflussen und können bei der Bestimmung des BERs bewusst modifizert werden, um die Gesamtqualität des Systems umfassender zu bewerten.

3.4 Daten- und Taktrückgewinnung

3.4.1 Herausforderungen der Datenrückgewinnung im Empfänger

In der digitalen Übertragung ist zwischen asynchronem und synchronem Betrieb zu unterscheiden. In der asynchronen digitalen Übertragung ist der Empfangstakt des Empfängers asynchron zum Sendetakt des Senders. So können nur kleine Pakete (engl. *burst-mode*) übertragen werden, die häufig mit einem Startbit beginnen und einem Stoppbit enden, auf die sich der Empfänger zeitweise synchronisiert. Bekannte Systeme und Protokolle sind zum Beispiel das UART-Protokoll und der CAN-Bus. Synchrone System bieten hingegen die Möglichkeit einer kontinuierliche Übertragung von großen Datenpaketen, wie beispielsweise bei USB. Um eine möglichst fehlerfreie Übertragung zu gewährleisten, muss das empfangene Signal vom Empfänger zu den richtigen Zeitpunkten abgetastet werden. Dazu wird ein Takt benötigt, der synchron mit dem Takt des Datenstromes verläuft und der in seiner Phasenbeziehung die Signallaufzeit zwischen Sender und Empfänger weitestgehend kompensiert [10, S. 363]. Ist die Datenrate und die damit benötigte Taktrate zur Rückgewinnung bekannt, ist noch keine fehlerfreie Detektion möglich, da beispielsweise durch Drift und Temperaturabhängigkeiten die Taktraten von Sender und Empfänger sich minimal unterscheiden können, was zu Fehlern in der Übertragung führen würde [9, S. 11].

Es gibt drei Methoden, um die Synchronisation des Empfängers mit dem Sender zu erreichen. So können Sender und Empfänger mit einem Takt von einer gemeinsamen Taktquelle betrieben werden, was passend für große Datenmengen mit sehr hoher Bandbreite ist. Jedoch gestaltet sich die technische Umsetzung bei einer großen räumlichen Trennung schwierig. Des Weiteren ist es möglich über den Kanal das Taktsignal separat zu übertragen, was freie Kapazitäten im Kanal und freie Kapazitäten an die Sendeleistung stellt. Diese Methode wird beispielsweise bei SPI und I^2C angewandt. Eine dritte Möglichkeit ist es, das Taktsignal direkt aus dem Datensignal zu gewinnen. Dabei nimmt die Komplexität des Empfängers zu, jedoch ist diese Methode sehr energieeffizient und benötigt keine zusätzliche Bandbreite im Kanal. [10, S. 363] Dieses Verfahren wird in der Regel bei kabelungebundenen Kommunikationssyteme wie etwa Wireless-Fidelity (Wi-Fi), aber ebenfalls bei USB angewandt.

Der folgende Abschnitt zeigt einige grundlegende Ansätze zur Takt- und Datenrückgewinnung.

3.4.2 Ansätze zur Realisierung

Die Clock-Data-Recovery (CDR), bildet im Empfänger eines Kommunikationssystems den Funktionsblock, welcher für die Daten- und Taktrückgewinnung und der Einsynchronisation der gewonnenen Daten mit dem System verantwortlich ist. So gewinnt sie aus dem verrauschtem Empfangssignal die Daten- und Taktinformationen zurück und bereitet diese für die Weiterverarbeitung im Empfänger auf. Kernelement bildet in der Regel ein taktflankengesteuertes D-Flipflop, welches die Datenbits zum Systemtakt des Empfängers synchronisiert. [11] Allgemein können CDRs ausschließlich aus analogen oder digitalen Komponenten bestehen, aber auch die Verwendung von analogen und digitalen Komponenten in einem System ist möglich. Der Aufbau von CDRs lässt sich in unterschiedliche Kategorien aufgliedern. So bilden Topologien, die eine Regelschleife zur Justierung des erzeugten Taktes besitzen eine Kategorie. Beispiele sind hier die PLL, ein Delay-Locked-Loop (DLL) und der Phaseninterpolator. Eine weitere Kategorie bilden Topologien, die auf Überabtastung beruhen, jedoch keine Regelschleife zur Phasenjustierung aufweisen. [11, S. 46-47]

Einzelne Vertreter beider Kategorien werden nachfolgend kurz näher beschrieben.

Phase-Locked-Loop

Die auf einer PLL basierten CDRs bieten die Möglichkeit einer relativ freien einstellbaren Datenrate. Die Abbildung 3.15 zeigt das Blockschaltbild einer PLL-Architektur ohne Referenztakt. Eine frequenzverfolgende Regelschleife liefert durch einen Frequenzdetektor (FD) einen Vergleichswert zwischen den Eingangsdaten und dem intern erzeugten Takt zur Abtastung und stellt die Frequenz des internen Taktes ein. Der interne Takt wird von einem Voltage-Controlled-Oscillator (VCO) geliefert und benötigt keinen externen Referenztakt. Eine phasenverfolgende Regelschleife bringt mithilfe des Ausgangssignales eines Phasendetektors (PD) den internen Takt in die gewünschte Phase zu den Eingangsdaten, um eine korrekte Abtastung zu gewährleisten.

Während des Startens oder bei Verlust der Phasenfixierung ist der Frequenzdetektor aktiv und kontrolliert die Eingangsspannung des VCOs mithilfe einer Ladungspumpe (engl. charge pump) (CP) und eines Schleifenfilters (engl. loop-filter) (LF), welcher zur Eliminierung von hochfrequenten Störungen dient. Dadurch wird die Frequenz des VCOs der Frequenz der Datenrate angepasst. Fällt die Frequenzdifferenz in den Fangbereich der

phasenverfolgenden Regelschleife, passt diese über einen eigenen Regelkreis, bestehend aus CP und LF, die Phase des VCOs gewünscht an. Die Phasenbeziehung des Datentaktes und des wiedergewonnen Taktes wird durch den Phasendetektor bestimmt. Für eine zuverlässige Detektion des Datenbits, wird dieses in der Regel mit einem Takt abgetastet, der 180° zum Sendetakt phasenverschoben ist und somit in der Mitte des Datenauges liegt. Ist die Phase und die Frequenz fixiert, liefert der Phasendetektor die korrekt abgetasteten Daten. Um Störungen und Interferenzen beider Schleifen untereinander zu unterdrücken muss der Arbeitsbereich beider Regelschleifen angepasst werden. Die phasenverfolgende Regelschleife arbeitet normalerweise mit einer geringeren Bandbreite als der frequenzverfolgende Regelkreis. Die dargestellte Variante ist eine rein analoge PLL. Einzelne Bauelemente können durch digitale Komponenten ausgetauscht werden, bis hin zu einem All-Digital-Phase-Locked-Loop (ADPLL). Durch den Austausch der analogen Komponenten kann die benötigte Fläche auf dem Chip reduziert werden und die Akquirierungszeit der Phase und des Taktes ebenfalls. Des Weiteren gestalten sich Stabilitätsanalysen einfacher und das System verliert an Spannungs- und Temperaturabhängigkeiten. [11, S. 47-48] Ebenso lassen sich auch PLLs nur mit einem Regelkreis implementieren, der sowohl die korrekte Frequenz als auch die korrekte Phasenbeziehung einstellt.

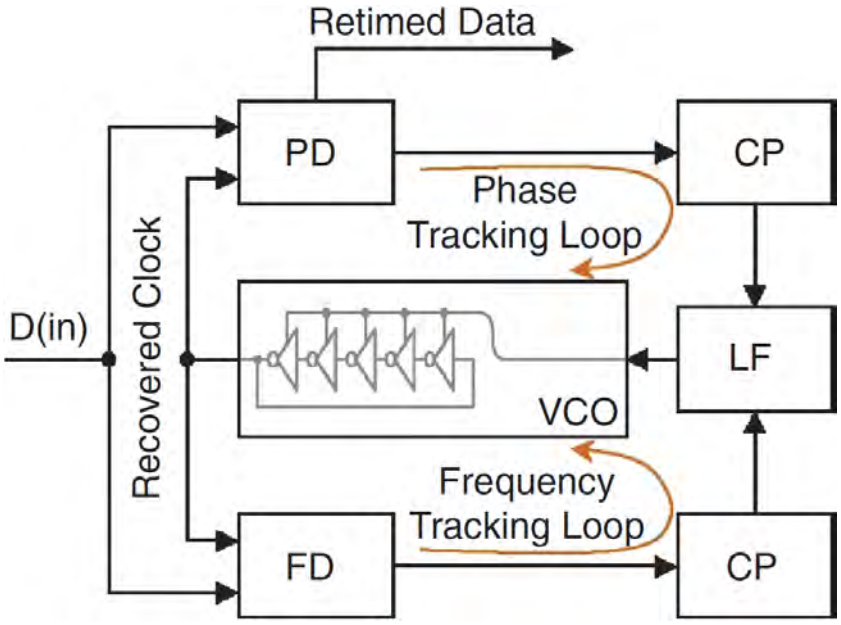


Abbildung 3.15: PLL als CDR ohne externen Referenztakt, modifiziert nach [11, S. 49]

Phaseninterpolator

Eine CDR mit einem Phaseninterpolator hat unterschiedliche Vorteile gegenüber der Verwendung einer CDR mit einer PLL. So läuft das System stabiler, benötigt eine kürzere Akquirierungszeit der Phase und ist weniger anfällig für Jitter. Die Abbildung 3.16 zeigt das Blockschaltbild einer CDR mit Phaseninterpolator. Eine PLL erzeugt aus einem externen Referenztakt mehrere, fest zueinander phasenverschobene Takte für den phasenverfolgenden Regelkreis. Dieser wählt eine Taktphase der erzeugten Takte aus und gewinnt damit die Daten zurück. Über den Phasendetektor (PD) und einen digitalen Schleifenfilter (DLF) wird die Phase bei einer Abweichung erneut gewählt. In der phasenverfolgenden Schleife werden nur digitale Komponenten verwendet. Ein Nachteil ist, dass diese Lösung im Gegensatz zur reinen Verwendung einer PLL eine geringere Bandbreite zur Akquisition der Datenrate bietet. [11, S. 50-52]

Die Datenrate muss vorher bekannt sein, damit die entsprechenden Referenztakte mit nahezu gleicher Frequenz erzeugt werden können. Kleine Taktabweichungen zwischen Sender um Empfänger lassen in der Regel kompensieren.

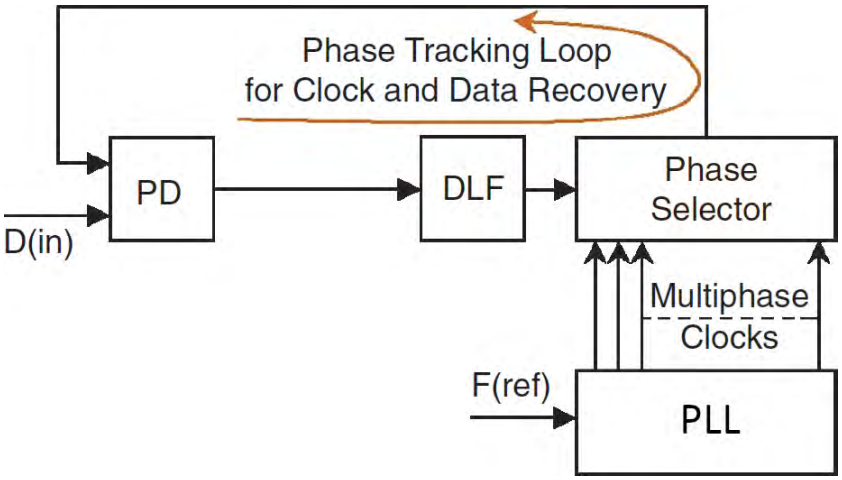


Abbildung 3.16: Phaseninterpolator als CDR, modifiziert nach [11, S. 52]

Überabtastung

Ein CDR-Design, dass auf Überabtastung beruht, ist als Blockschaltbild in Abbildung 3.17 dargestellt. Jedes Bit wird mehrfach abgetastet und über Entscheidungslogik wird die richtige Bitsequenz gewonnen. Es sind mindestens drei Samples pro empfangenen Datenbit erforderlich, welche in der Regel über mehrere, phasenverschobene Takte gewonnen werden, die durch beispielsweise eine PLL bereitgestellt werden. Jedoch ist es auch möglich, die Überabtastung mit einem sehr hochfrequenten Takt durchzuführen und jeder Abtastzeitpunkt bildet eine eigene Abtastphase. Die abgetasteten Werte werden dann an den Datenrückgewinnungsblock übergeben, welcher aus einem Datenregister, Flankendetektor und Datenwähler besteht. Der Flankendetektor lokalisiert die Kanten bzw. Transitionszeitpunkte in den Daten. Anhand des Ausgangs des Flankendetektors wählt der Datenwähler das richtige Bit aus, welches zurückgewonnen wird. [11, S. 53]

Die zwei Basiskonzepte zur Auswahl der richtigen Datenbits beruhen auf Phasenwahl oder der schlechteren Mehrheitswahl. Die Mehrheitswahl nimmt an, dass die Mehrheit aller Samples einer Überabtastungsperiode den Wert des eigentlich zu samplenden Bits haben sollte. Die Phasenwahl beruht auf der Detektierung von Transitionen in der Übertragung, womit sich die Mitte des Datenauges bestimmen lässt und die entsprechende Taktphase zur Ermittlung des korrekten Bits ausgewählt wird. Die Auswahl der Taktphase kann direkt nach dem Auftreten einer Kante neu gewählt werden, bekannt als Direct-Phase-Picking (DPP), oder nach einem festen Intervall von ankommenden Bits, bekannt als Average-Phase-Picking (APP). Das Auftreten einer Mehrheit von Kanten in einer Taktphase lässt darauf schließen, dass diese Phase am weitesten von der Mitte des Datenauges entfernt ist und für die Rückgewinnung nicht genutzt werden sollte. Das DPP besitzt eine schlechtere Performance in Bezug auf das BER, ist jedoch mit weniger Hardwareaufwand und Leistungsbedarf als APP verbunden. Wird der Faktor zur Überabtastung erhöht, lässt sich die Phasenauflösung verbessern und damit der Phasenfehler verringern. Gleichzeitig ist dies mit steigender Komplexität der Schaltung verbunden. [12, S. 74]

Grundlegende Vorteile dieser Topologien sind die schnelle Datengewinnung ohne Verzögerung bzw. Akquirierungszeit, keine Jitterakkumulierung und die allgemeine Stabilität der Systeme. Da keine Rückkopplung vorhanden ist und es sich um ein Feed-Forward System handelt, eignen sich diese Topologien für Systeme mit einer sehr hohen Datenbandbreite. Einsetzbar sind diese Systeme bei kontinuierlichen Datenübertragungen und paketbasiertem burst-mode. Für die übertragenden Daten wird jedoch gefordert,

dass diese sehr viele Transitionen aufweisen, um die korrekten Samples auszuwählen und gegen hochfrequente Jitter möglichst resistent zu sein. Des Weiteren sind große Datenregister erforderlich, um Daten zwischenzuspeichern, welche gerade bei sehr schnellen und asynchronen Datenübertragungen stark zunehmen. [11, S. 53]

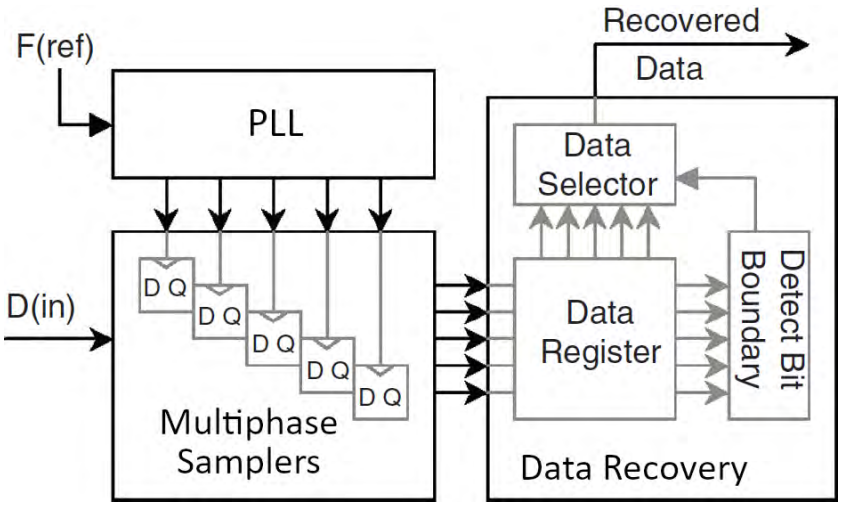


Abbildung 3.17: Blockschaltbild einer CDR mit Überabtastung, modifiziert nach [11, S. 53]

3.5 Optische Übertragung

3.5.1 Allgemein

Die kabelungebundenen Übertragungstechnologien bieten eine hohe Flexibilität und Mobilität und bieten in gewissen Szenarien eine gute Alternative zu anderen bekannten Übertragungssystemen. Heutzutage werden überwiegend Radiowellen aus dem elektromagnetischen Spektrum für die digitale Übertragung verwendet, beispielsweise im Mobilfunk oder Wi-Fi. Jedoch werden zunehmend Wellen aus dem optischen Spektrum, wie etwa infrarote Strahlung (IR), für die digitale Übertragung interessant und verwendet. In unterschiedlichen Einsatzmöglichkeiten ist die optische Übertragung der konventionellen Übertragung mit Radiowellen überlegen. So bietet diese eine deutlich höhere Bandbreite und eine breite Auswahl an Übertragungsfrequenzen, welche zur Zeit dieser Arbeit weltweit im Frequenzbereich unreguliert sind. Die Verwendung von Radiofrequenzen ist dagegen stark reguliert und mit dem Erwerb von Lizenzen verbunden. [13, S. 1]

Ein weiterer Vorteil von optischer Kommunikation ist deren Robustheit gegenüber elektromagnetischen Interferenzen und gewährleistet daher ein alternatives Verfahren zur zuverlässigen Datenübertragung. Zusätzlich werden konventionelle Kommunikationssysteme mit Radiowellen durch IR nicht gestört, was einen parallelen Betrieb beider Technologien ermöglicht. Des Weiteren verhält sich IR wie sichtbares Licht und durchdringt keine Objekte oder Mauern, was das Abhören von übertragenen Informationen außerhalb der eingesetzten Räumen nicht ermöglicht. Aber auch die Unterbrechung der Verbindung durch beispielsweise Personen ist möglich. Dazu bietet die Übertragungstechnologie mit IR die Möglichkeit, ein Netzwerk und flexible temporäre Verbindungen zeitsparend aufzubauen. Weitere Vorteile von IR gegenüber Radiowellen sind die geringe Größe und der begrenzte elektrische Leistungsbedarf der optischen Komponenten durch die zunehmend voranschreitende Entwicklung und Verbesserung optisch-elektronischer Geräte in den vergangenen Jahren. [13, S. 2]

An ihrem Einsatzort können die optischen Systeme durch andere Lichtquellen beeinflusst werden. So enthält herkömmliches Umgebungslicht einige Anteile am infraroten Spektrum, was sich als Hintergrundrauschen im Empfangssignal äußert. Des Weiteren erfährt IR zum Teil große Dämpfungen bei der Übertragung durch die Luft und die Atmosphäre. So können Nebel oder Schnee die Reichweite und Qualität des Signal stark einschränken. Durch eine höhere Leistung lässt sich das Problem der Dämpfung und

des Hintergrundrauschens weitestgehend beseitigen, jedoch wird die gesendete Strahlung potentiell gefährlich für das menschliche Auge und kann irreversible Schädigungen hervorrufen. Generell werden kabelungebundene optische Systeme nicht die konventionellen Systeme basierend auf Radiowellen ersetzen und der Einsatz muss je nach Applikation abgewogen werden. So eignen sich optische Übertragungstechnologien vorrangig für kurze Distanzen, in denen Sicherheit, hohe Datenraten und Robustheit gegenüber anderer elektromagnetischer Strahlung wichtig ist. [13, S. 3]

3.5.2 Systemkonfigurationen

Das Einsatzgebiet optisch-kabelungebundener Systeme ist vielfältig. So können sie inner- oder außerhalb von Gebäuden eingesetzt werden, was zu unterschiedlichen Konfigurationen der Sender- und Empfängeranordnung führt. Im Allgemeinen lassen sich die Systeme anhand des optischen Pfades zwischen Sender und Empfänger und dem jeweiligen Sende- und Empfangswinkel beschreiben. Die Abbildung 3.18 zeigt die möglichen Konfigurationen der optischen Systeme. Das System ist *direkt* (engl. directed), wenn der Sender ein gerichtetes Signal mit kleinem Winkel in Richtung des Empfängers übermittelt und der Empfangsbereich des Empfängers ebenfalls sehr klein und auf den Sender gerichtet ist. Arbeitet der Sender oder Empfänger in einem größeren Sende- oder Empfangswinkel handelt es sich um ein *hybrides* System. Dagegen zeichnet sich ein *nicht-direktes* System durch einen großen Betriebswinkel beider Komponenten aus. Ist kein ungehinderter Sichtkontakt (Non LOS) zwischen Sender und Empfänger gewährleistet muss mit reflektierenden Oberflächen gearbeitet werden, um eine zuverlässige Verbindung aufzubauen. Die direkte Verbindung mit einem direkten optischen Pfad zwischen Sender und Empfänger ist am energieeffizientesten.

Ein spezieller Fall ist die zellulare Konfiguration. Hier versorgen stationäre Transceiver mehrere Bereiche oder Zellen konstant mit optischer Leistung. Dieses System bietet Mobilität und ähnelt einem lokalen Wi-Fi-Netzwerk. Im Allgemeinen sind optische Verbindungen gerichteter als Verbindungen mit Radiowellen, was auf die unterschiedlichen Sendeverfahren zurückzuführen ist. Jedoch bleibt die Anfälligkeit für Unterbrechungen der optischen Verbindung, durch beispielsweise Personen, weiterhin bestehen. [13, S. 4-10]

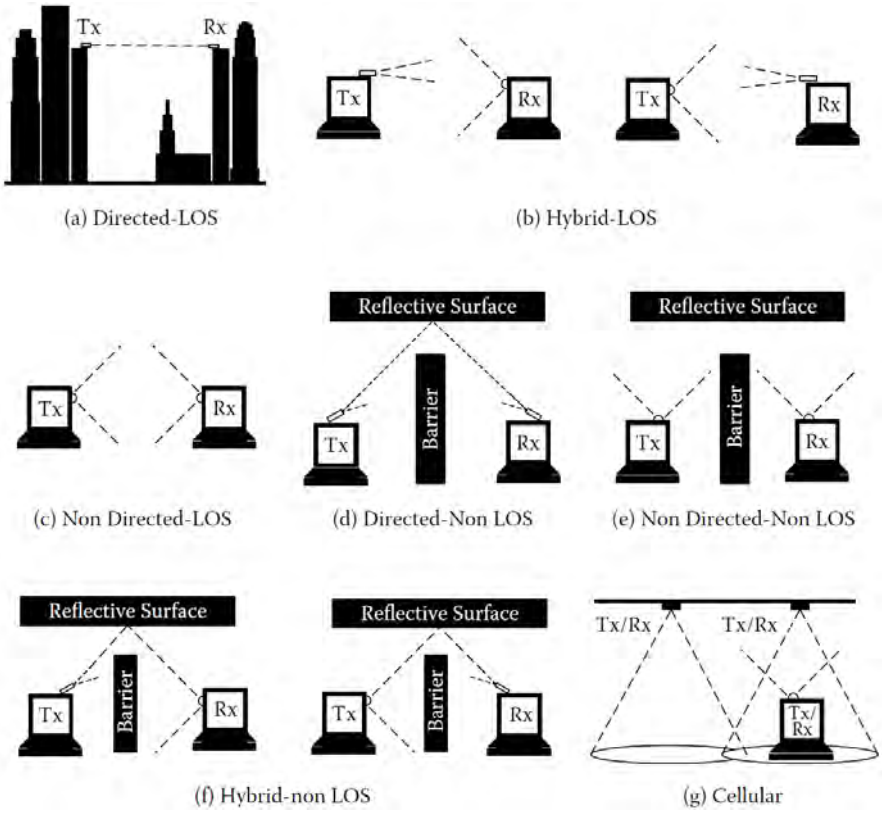


Abbildung 3.18: Konfigurationsmöglichkeiten von kabelungebunden optischen Systemen [13, S. 5]

3.5.3 Technische Beschränkungen

Die Licht emittierenden Dioden (LEDs) und Laserdioden (LDs) bilden heute die Kernelemente in der optischen Datenübertragung. Abhängig vom Einsatzort, gewünschter Bandbreite und gesundheitlichen Aspekten wird eine Komponente für das gewünschte System gewählt. Für Applikation innerhalb von Gebäuden werden vorzugsweise LEDs verwendet, da ihre Strahlung prinzipiell als ungefährlich für das menschliche Auge eingestuft wird. Dagegen müsste die optische Leistung der LDs beschränkt werden oder aufwendigere Optik wäre nötig, um die Augensicherheit zu gewährleisten. LEDs werden in direkten oder hybriden System eingesetzt, da ihr Sendewinkel vergleichsweise hoch ist. Durch eine Anordnung mehrerer LEDs ist es möglich, deutlich größere Bereiche abzudecken und ein nicht-direktes System aufzubauen. Systeme, basierend auf LDs, werden dagegen durch den sehr kleinen Sendewinkel hauptsächlich als direkte Systeme konfiguriert. Des Weiteren kann die Modulationsbandbreite von LDs einige zehnfache Gigahertz

erreichen und im Vergleich dazu ist die Modulationsbandbreite von LEDs geringer und erreicht maximal einige hundert Megahertz. Demzufolge wird für Applikationen mit sehr hohem Datenaufkommen die LD bevorzugt. [13, S. 66-67]

3.5.4 Technische Umsetzungen und Standards

Im Jahre 1993 wurde die internationale Organisation Infrared Data Association (IrDA) gegründet, um interoperable Standards für Soft- und Hardware der IR Kommunikation zu setzten. Zu den Mitgliedern gehören mehr als 160 Unternehmen aus der Telekommunikations- und Automobilbranche, sowie große Produzenten und Entwickler für Soft- und Hardware, von Kommunikationssystemen und Peripheriegeräten. Das allgemeine Ziel besteht darin einfache, kosteneffektive und leistungseffiziente Transceiver für die kabelungebundene optische Kommunikation zu entwickeln. [13, S. 249]

Die IrDA-Transceiver bestehen auf Empfängerseite aus einem Photodetektor und einem Decoder. Der Photodetektor wandelt die optischen Pulse in elektronische Pulse um, aus denen der Decoder Datenbits formt. Außerdem kontrolliert der Empfänger die Verstärkung der elektrischen Pulse, da durch unterschiedliche Abstände der Transceiver die optische Leistung und die Pegel der elektrischen Pulse schwanken. Damit eine zuverlässige Übertragung erhalten bleibt, muss der Empfänger die Pegel der elektrischen Pulse zum Abstand anpassen. Für die Senderseite werden LEDs verwendet, die eine Wellenlänge von 880 nm aussenden. Des Weiteren muss eine kontinuierliche Datenübertragung über 1 m aufrechterhalten werden, mit einem BER $< 10^{-8}$. Mögliche Datenraten reichen von 9,6 kbps bis 100 Mbps. [13, S. 249 ff.]

Im Jahr 2011 wurde von Professor Harold Hass von der Universität Edinburgh eine Idee über eine neue Form der kabelungebundenen Netzwerktechnologie vorgeschlagen. Dazu nutzte er optische Glasfaser und LEDs, um die Daten zu übertragen. Lichtmodulation ist keine neue Technologie, doch der neue Ansatz besteht darin, die Konnektivität durch einfache, herkömmliche LEDs in Lampen zu erreichen. Light-Fidelity (Li-Fi) bildet hierbei eine Technologie, zur Vernetzung verschiedenster Geräte, in einer ähnlichen Funktion wie die herkömmliche Wi-Fi-Technologie. [14] Mit dem Aufkommen des Internet of Things werden zuverlässige Kommunikationskanäle immer wichtiger. Li-Fi nutzt sichtbares Licht von LEDs als Medium zur Datenübertragung, wodurch alle Vorteile der optischen Kommunikation ausgenutzt werden, wie etwa die Robustheit gegenüber anderer elektromagnetischer Strahlung, sehr hohe Bandbreite und Abhörsicherheit. Die Tabelle 3.1 vergleicht

die beiden Technologien Wi-Fi und Li-Fi in unterschiedlichen Aspekten unter realen Aspekten. Im Labor sind jedoch deutlich höhere Reichweiten und Datenraten mit beiden Technologien möglich. [14]

Li-Fi biete eine gute Alternative zu Wi-Fi bei der zuverlässigen Vernetzung unterschiedlicher Geräte innerhalb von hindernisarmen Räumen, die die optische Strahlung blockieren könnten. Jedoch ist die Vernetzung mit Wi-Fi außerhalb von Räumen oder über mehrere Räume hinweg vorzuziehen. Zudem ist durch die erreichbaren Datenraten, die Verbindung von Li-Fi mit Ethernet für die Industrie interessant.

Tabelle 3.1: Vergleich zwischen Wi-Fi und Li-Fi, nach [14, S. 4]

	Li-Fi	Wi-Fi
Datenraten	$< 10\,\text{Gbps}$	$54 - 250\,\text{Mbps}$
Reichweite	$10\,\text{m}$	$20 - 100\,\text{m}$
IEEE Standard	802.15.7	802.11ac
Netzwerktechnologie	Punkt-zu-Muli-Punkt	Punkt-zu-Multi-Punkt
Übertragungsmedium	Licht, IR	Radiowellen
Frequenzband	mehrere hundert THz	$5\,\text{GHz}$

4 Methoden

4.1 Plattform

Die zu realisierende CDR soll in ein bestehendes optisches, kabelungebundenes Übertragungssystem des Fraunhofer IPMS implementiert werden, welches für Datenraten von 125 Mbps ausgelegt ist. In Abbildung 4.1 ist ein Blockschaltbild dargestellt, welches den prinzipiellen Aufbau der optischen Frontends beschreibt. Zum Einsatz in den optischen Tranceivern kommt eine Photodiode zum Empfangen der optischen Signale, welche durch einen Transimpedanzverstärker (TIA) und Limitierungsverstärker (LA) aufbereitet und als low-voltage-differential-signaling (LVDS)-Signal zur Verfügung gestellt werden. Die gewünschte Sendebandbreite wird mit LEDs erreicht. Die Bits werden im fertigen System mit einem NRZ-Signal als 8b10b Leitungscode übertragen und ein verhältnismäßig ressourcenarmer FPGA soll anschließend die Rückgewinnung und Weiterverarbeitung der Daten übernehmen und ebenfalls den Sendebetrieb gewährleisten. Für Testzwecke und erste Implementationen wird das Entwicklungsboard XCM-111-45T [15] von HuMANDATA mit einem Spartan-6-45T [4] verwendet. Hierbei handelt es sich um einen FPGA, der für die Lösung des Problems mehr als ausreichende Ressourcen zur Verfügung stellt und damit den Entwicklungsprozess der CDR vereinfacht. Ein weiteres Breakoutboard, das als Schnittstelle zwischen des optischen Frontends und dem HuMANDATA-Entwicklungsboard dient, wird ebenfalls zur Verfügung gestellt und besitzt GPIO-Pins und LEDs für komfortables Debuggen während der ersten Hardwaretests. Die optischen Frontends sind auf bidirektionale Übertragungen ausgelegt und ermöglicht vollen duplex Betrieb. Zusätzlich soll das fertige System echtzeitfähige Kommunikation nach Echtzeitstandards für Ethernet wie beispielsweise Profinet[1] ermöglichen. Allgemein bedeutet dies, dass die zu entwickelnde CDR für eine kontinuierliche Datenübertragung funktionieren muss und mit einer möglichst geringen Latenz die Daten zurückgewinnt und dem Gesamtsystem für die Weiterverarbeitung zur Verfügung stellt.

[1] https://www.profibus.com/

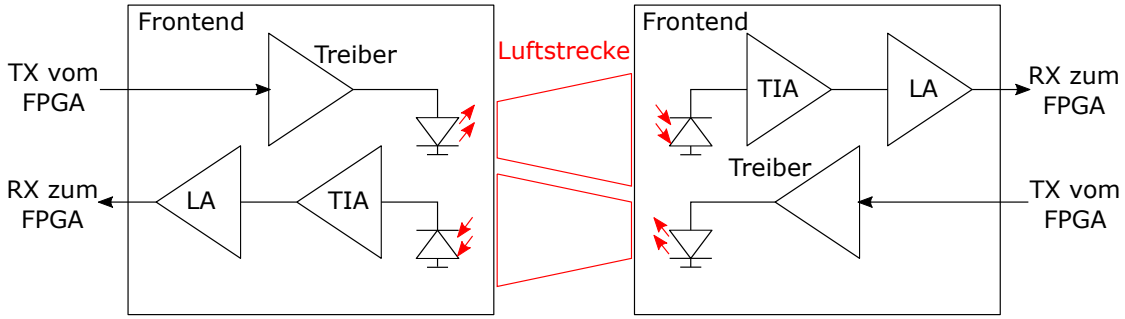


Abbildung 4.1: Blockschaltbild der optischen Übertragungsstrecke und der optischen Frontends

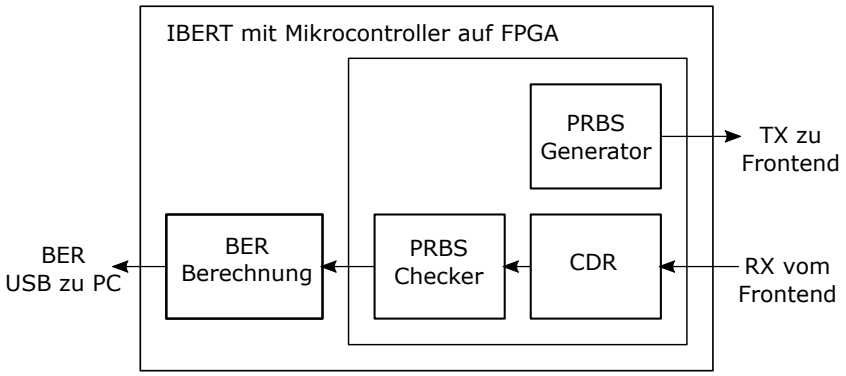


Abbildung 4.2: Blockschaltbild des IBERT-Systems

Die entwickelte CDR wird in einen bestehenden integrierten Bitfehlerratentester (engl. integrated-bit-error-tester) (IBERTs) des Fraunhofer IPMS zur Charakterisierung auf dem FPGA implementiert. In Abbildung 4.2 ist der allgemeine Aufbau des Gesamtsystem im FPGA rudimentär gezeigt. Vor der direkten Implementierung auf dem FPGA werden die einzelnen CDR-Komponenten simuliert und teilweise synthetisiert, um die Korrektheit der Beschreibung im Entwicklungsprozess zu überprüfen. Als Simulationssoftware wird der Xcelium Simulator von Cadence[2] und als Synthesetool die ISE-Design Suite 14.7 von Xilinx[3] verwendet. Teil des IBERTs ist ein vom Fraunhofer IPMS entwickelter Softcore, welcher zusätzlich im FPGA implementiert wird. Dieser steuert den Sende- und Empfangsbetrieb und kann das aktuelle BER der Übertragung ermitteln. Über eine USB-Schnittstelle kann der Softcore mit einem PC kommunizieren und das

[2] https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-parallel-simulator.html
[3] https://www.xilinx.com/products/design-tools/ise-design-suite.html

BER lässt sich direkt im Betrieb auslesen und graphisch darstellen. Die GUI zur Darstellung des BERs ist in Abbildung 4.3 dargestellt. Sie zeigt die empfangenen Bits, Anzahl der fehlerhaften Bits und das aktuelle BER an und ist für zwei parallele Sende- und Empfangskanäle im IPMS-IBERT-System ausgelegt. Zusätzlich ist es über die GUI möglich die Übertragung des Sendekanals zu starten oder zu stoppen und einzelne Bitfehler absichtlich im Sendestrom zu erzeugen.

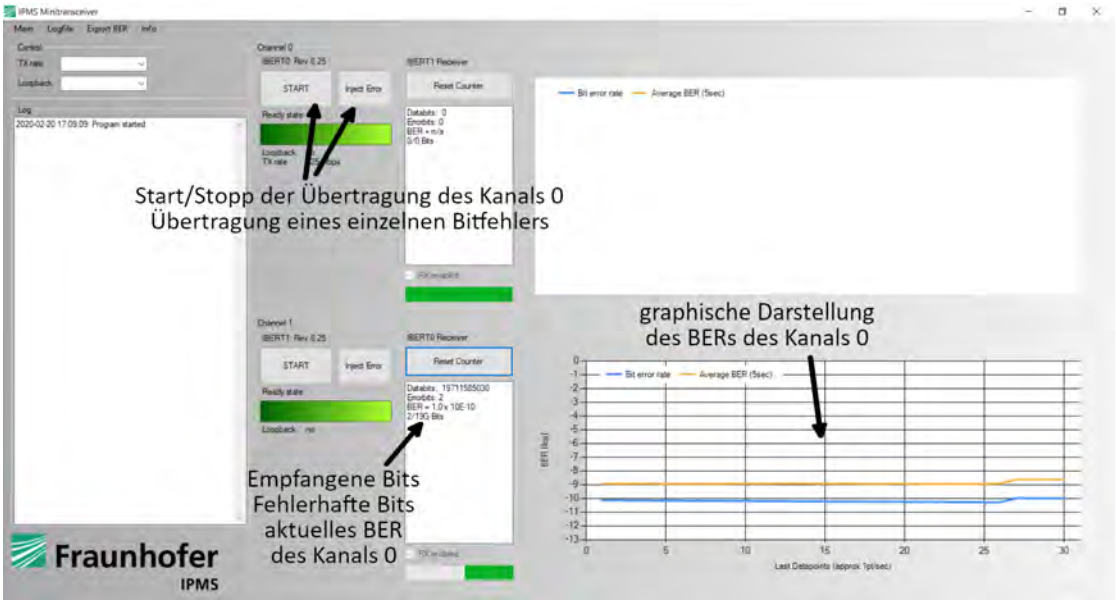


Abbildung 4.3: GUI zur Messung des BERs

4.2 Messkonzepte

4.2.1 Charakterisierung der optischen Transceiver

Zur Charakterisierung der Übertragungsstrecke und der optischen Transceiver wird das BER mit der implementierten CDR gemessen. Hierbei wird die zu sendende Bitfolge als pseudozufallgenerierter Bitstrom (engl. pseudo-random bit stream) (PRBS) generiert und ist somit in ihrem zeitlichen Verlauf bekannt. So lassen sich die zurückgewonnen Daten im Empfänger direkt mit der bekannten Bitfolge vergleichen und das BER ermitteln. Als PRBS-Generator und -Checker wird die von Xilinx in [16] zur Verfügung gestellte Implementation mit einigen Modifikationen für das IPMS-IBERT-System verwendet. Der PRBS-Datenstrom wird über linear rückgekoppelte Schieberegister realisiert. Des Weiteren lässt sich der Einfluss unterschiedlicher Anordnungen der Transceiver auf das BER untersuchen. Durch die Veränderungen der Ausrichtung der optischen Transceiver verändert sich die Empfangsleistung und das SNR. Über die Empfangsleistung, die über den gemessenen Photostrom an der Photodiode ermittelt wird, lässt sich eine allgemeinere Aussage über die Anordnung der Transceiver treffen, bis zu der eine funktionierende Übertragung möglich ist. Hier ist es wichtig einen Zielwert für das BER zu setzen, um die topologischen Grenzen zu ermitteln.

4.2.2 Jittertoleranz

Das Verhalten einer CDR auf unterschiedlichen Ausprägungen von Jitter kann durch eine Jitteroleranzmessung überprüft werden. Die Abbildung 4.4 zeigt den prinzipiellen Messaufbau einer Jittertoleranzmessung. Hierfür wird dem Takt, welcher einen PRBS-Generator betreibt, absichtlich periodischer Jitter durch Modulation hinzugefügt. Zur Modellierung des Jitters wird in der Regel als Modulator ein Sinus verwendet und dann das BER mit dem jitterbehafteten PRBS-Signal ermittelt. Anschließend kann durch das Setzen eines Ziel-BERs eine Jittertoleranzkurve ermittelt werden, welche eine Aussage liefert, wie die CDR in bestimmten Szenarien agiert und ob diese für die gewollte Implementation geeignet ist. Die Amplitude des modulierten Jitters wird langsam erhöht, bis das Ziel-BER nicht mehr erreicht wird und so die Toleranzgrenze der CDR und des selbst gewählten Ziels erreicht ist. Diese Messung wird mit steigenden Modulationsfrequenzen durchgeführt und so die Kurve aufgenommen. In Abbildung 4.5 ist ein beispielhaftes Jittertoleranzdiagramm dargestellt, in dem zusätzlich eine Kurve als Minimalanforderung

definiert ist. Die resultierende Jittertoleranzkurve liegt bei bestandenem Test über der Minimalanforderung. Im niederfrequenten Modulationsbereich werden Taktwanderungen modelliert, in denen die Jitteramplitude in der Regel $>1\,\mathrm{UI_{pp}}$ ist. Diese Wanderungen sollte die CDR erfassen und verfolgen können und weiterhin die Daten korrekt zurückzugewinnen. Im hochfrequenten Modulationsbereich sind die Jitteramplituden $<1\,\mathrm{UI_{pp}}$ und modellieren damit hochfrequenten Jitter, welcher eine Schließung des Datenauges hervorruft. Die Aufgabe der CDR ist es, diesen Jitter zu ignorieren und weiterhin eine korrekte Zurückgewinnung der Daten ermöglichen. [7, 17]

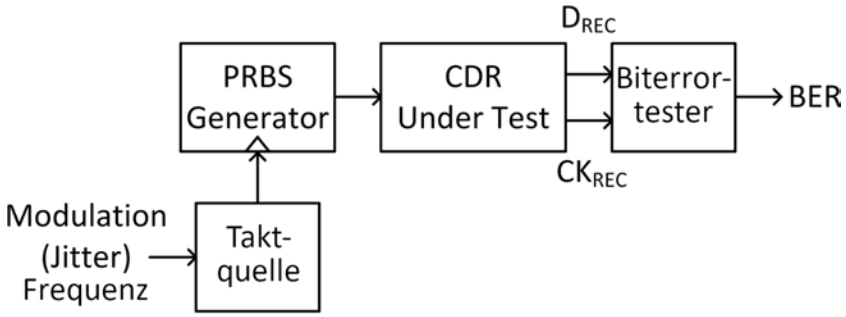


Abbildung 4.4: Prinzipielles Messverfahren zur Aufnahme einer Jittertoleranzkurve, modifiziert nach [7, S. 151]

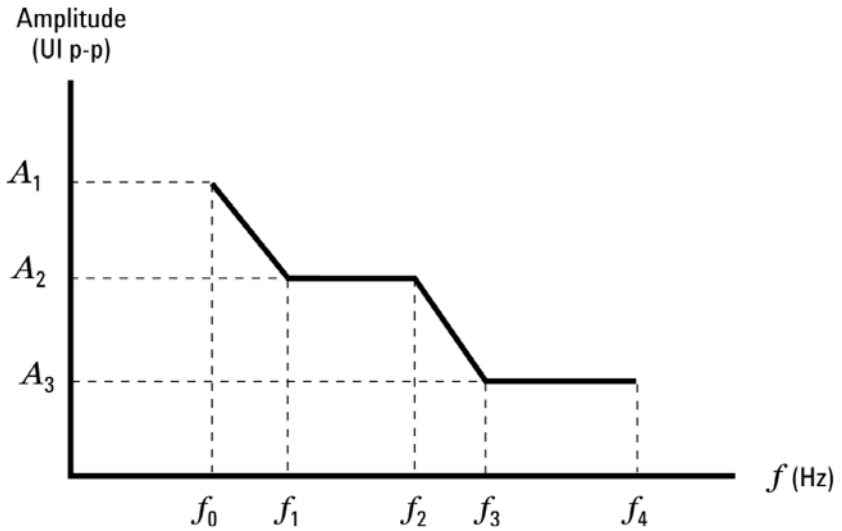


Abbildung 4.5: Beispielhaftes Jittertoleranzdiagramm mit eingezeichneter Spezifikation und Eckfrequenzen [17, S. 3-10]

4.3 Vorstellung der Lösungskonzepte

Im folgenden Abschnitt werden unterschiedliche Lösungskonzepte zur Realisierung einer voll-digitalen CDR unter Zuhilfenahme von Literatur vorgestellt. Es werden unterschiedliche Eigenschaften bewertet, wie etwa der Hardwareaufwand, Realisierbarkeit auf einem FPGA, Komplexität, abschätzbare Datenrate und Zuverlässigkeit.

4.3.1 Entscheidungsschaltung mit geringem Hardwarebedarf und blinder Überabtastung (BO-DRU)

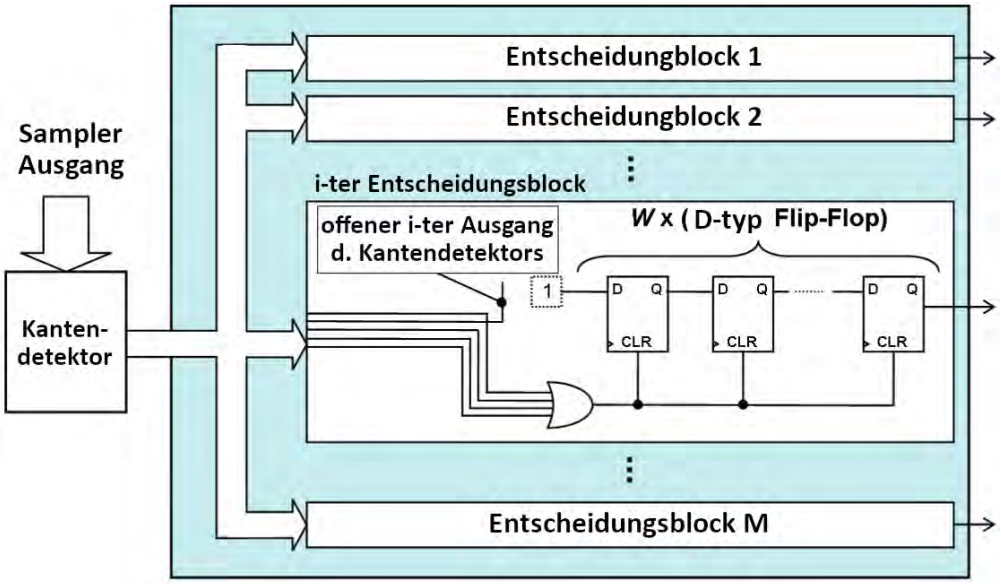
In [12] werden zwei Algorithmen zur Datenrückgewinnung nach dem Prinzip einer Datenrückgewinnungseinheit mit blinder Überabtastung (engl. blind-oversampling data-recovery-unit) (BO-DRU) vorgestellt. Sie unterscheiden sich in der Hardwareimplementation von herkömmlichen, bekannten Implementationen des DPPs und APPs. So verbinden sie den geringen Hardwareaufwand des DPPs mit der Performance des APPs. Beide Algorithmen werden auf einem überabgetasteten Bitstrom angewendet und zeichnen sich vor allem durch ihre geringe Komplexität und geringen Hardwarebedarf aus. Sie sind komplett digital realisierbar, lassen sich einfach auf einem FPGA oder als ASIC implementieren und bieten eine ähnliche Jittertoleranz und Performance zu Ansätzen, die eine PLL verwenden. Der prinzipielle Aufbau und die Funktionsweise von CDRs mit Überabtastung ist in Unterabschnitt 3.4.2 beschrieben.

Die vorgestellten Algorithmen *S2par* und *Ccnt* besitzen einen einstellbaren Parameter W zur Justierung der CDR und den Überabtastungsfaktor M. Bei *S2par* werden in einem fließenden Intervall von W konsekutiven Bits die Positionen der Kanten in einer Taktdomäne bestimmt und treten alle Kanten in einer Taktdomäne auf, wird die Phase zur Rückgewinnung neu gewählt, ähnlich zu APP. Als Kantendetektor lässt sich eine einfache XOR-Verknüpfung benachbarter Taktdomänen implementieren. Die Abbildung 4.6a zeigt die Hardwareimplementation von *S2par*. Der Kantendetektorausgang führt zu M-Entscheidungsblöcken, welche das Auftreten von Kanten in einer Domäne testen. Nach dem Empfangen von W Bits wird der Eingang „1“ aus dem Schieberegister geschoben, wenn alle Kanten nur in der i-ten Domäne auftraten, um anschließend eine neue Taktphase zur Rückgewinnung zu ermitteln. Tritt eine Kante in einer anderen Domäne auf, wird das Schieberegister zurückgesetzt und es wird keine neue Taktphase gewählt. Die optimale Taktphase p_{opt} wird bei ungeradem M nach Gleichung 4.1 gewählt.

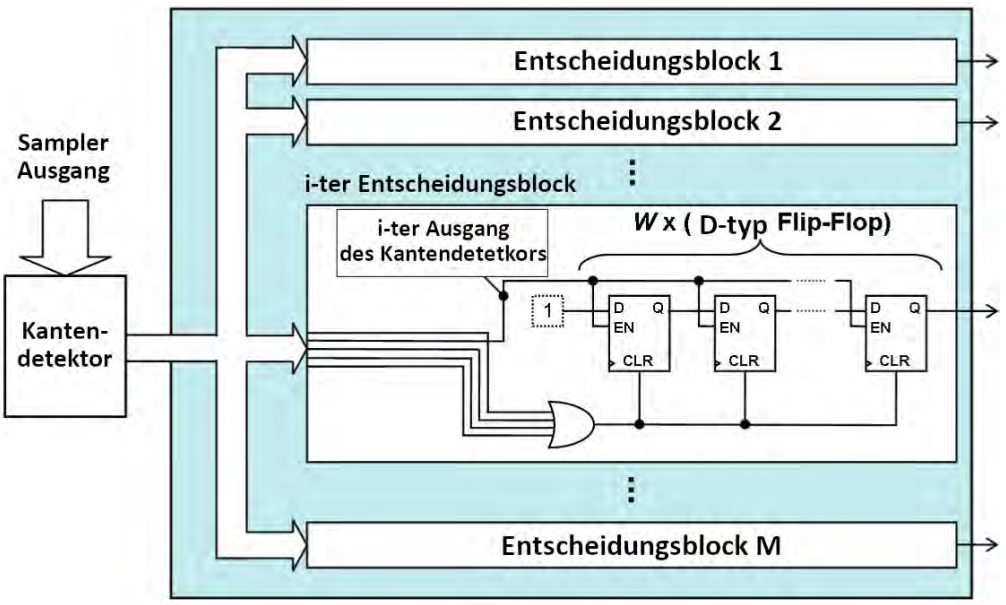
$$p_{opt} = \left(i + \frac{M+1}{2} \right) \cdot \mod M \tag{4.1}$$

Im Gegensatz zu APP werden die auftretenden Kanten nicht gezählt und es kommt auch zu keiner komplexen Mehrheitsentscheidung, welche Taktdomäne mehr Kanten aufweist. Die Wahl der optimalen Taktphase nach Gleichung 4.1 kann auf dem FPGA durch eine einfach LUT gelöst werden und bedarf keiner komplexeren Berechnung. Der *Ccnt*-Algorithmus ist in seiner Hardwareimplementation ähnlich, dargestellt durch Abbildung 4.6b. Dieser wählt eine neue Taktphase zur Rückgewinnung beim Auftreten von W-Kanten in einer Taktdomäne aus. Tritt eine Kante in der entsprechenden Taktdomäne auf, wird die „1“ eine Instanz weiter durch das Schieberegister geschoben. Treten jedoch Kanten in anderen Taktdomänen auf wird das Schieberegister zurückgesetzt. Die Schieberegisterlänge W ist bei *Ccnt* in der Regel kleiner zu wählen als bei einer identischen Übertragung mit implementierten *S2par*-Algorithmus. Jedoch benötigen beide Algorithmen für eine zuverlässige Arbeitsweise viele Transitionen in den übertragenen Daten, um stets die korrekte Taktphase zu wählen.

Das Taktsignal zur Steuerung der Flipflops aller Entscheidungsblöcke sollte eine Taktdomäne sein, welche ebenfalls zur Abtastung genutzt wird. Des Weiteren ist die nominale Frequenz der einzelnen Takte gleich der Datenrate zu wählen und ein Überabtastungsfaktor von mindestens fünf zu erreichen. Da kaum kombinatorische Logik verwendet wird und der Großteil des Systems durch Schieberegister realisierbar ist, eignen sich die vorgestellten Umsetzungen sehr gut für einen FPGA. So sind sehr hohe Datenraten bis an die Grenze der realisierbaren Überabtastung möglich und ein geringer Energiebedarf ist erwartbar. Je höher der Faktor der Überabtastung ist, umso höher ist die Phasenauflösung und das System kann genauer auf Jitter und Verschiebung des Auges reagieren. Bei einer Impulsverkürzung durch einen DC-Offset auf dem Kanal behält das System die Phase bei und sollte weiterhin korrekt funktionieren, da die gewählte Phase weiterhin in der Mitte des Datenauges liegt. Jedoch zeigt sich in den vorgestellten Messergebnissen auch, dass die Systeme nur eine leichte Verbesserung in Bezug auf das BER gegenüber den etablierten Algorithmen APP und DPP bieten. Für optische Systeme, die zum Teil sehr störanfällig sind, ist diese Eigenschaft zu berücksichtigen. Außerdem zeigen die vorgestellten Algorithmen keine Möglichkeit zur Detektion von Taktwanderung und dem damit verbundenen Verlieren von Bits durch Bitslip, wenn zum Beispiel der Sendetakt den Empfangstakt um eine Phase überholt.



(a) Einscheidungsblock des *S2par* Algorithmus, modifiziert nach [12, S. 76]



(b) Einscheidungsblock des *Ccnt* Algorithmus [12, S. 76]

Abbildung 4.6: Aufbau der vorgestellten Entscheidungsblöcke

4.3.2 Phaseninterpolator (MRCP)

Zum allgemeinen Aufbau einer CDR mit Phaseninterpolator, vorgestellt in Unterabschnitt 3.4.2, wird in [18, 19] eine CDR mit Multi-rotierenden-Taktphasen (engl. Multi-Rotating-Clock-Phases) (MCRP) gezeigt, die zusätzlich die Position des Datenauges einer Datenübertragung ermittelt und verfolgt. Über die Phasenbeziehung des Datenauges zu den wählbaren Taktphasen, wird die optimale Taktphase zur Rückgewinnung der Daten ermittelt. Der prinzipielle Aufbau der Schaltung und deren Funktionsblöcke sind in Abbildung 4.7 gezeigt. Diese Schaltung wurde auf einem ASIC getestet und erfüllt die Spezifikationen des SerDes-Framer-Interface 5 [20] und damit die elektrischen Charakteristiken des System-Interface-Level-5 (SIL-5) [21] und erreicht ein BER von 10^{-12} bei einer Übertragung von 2,5 Gbps mit einem hochfrequenten, totalen Jitter von 0,7 $\mathrm{UI_{pp}}$. Die Schaltung besteht aus vier Funktionsblöcken und in der vorgestellten Implementierung werden acht Taktphasen zur optimalen Auswahl des Abtastzeitpunktes zur Verfügung gestellt. Die acht Takte stehen in einer festen, unveränderbaren Phasenbeziehung zueinander, sind jeweils um 45° phasenverschoben und laufen mit der Frequenz der Datenrate. Die Taktphasen werden durch den Taktinterpolator (CI) und -selektor (CS) erzeugt und entsprechend multiplexed.

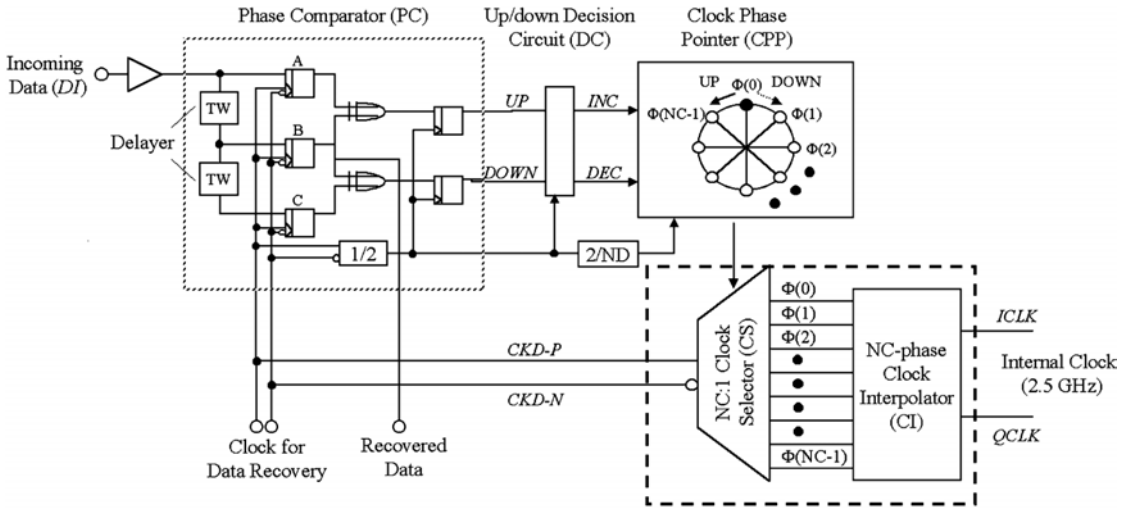
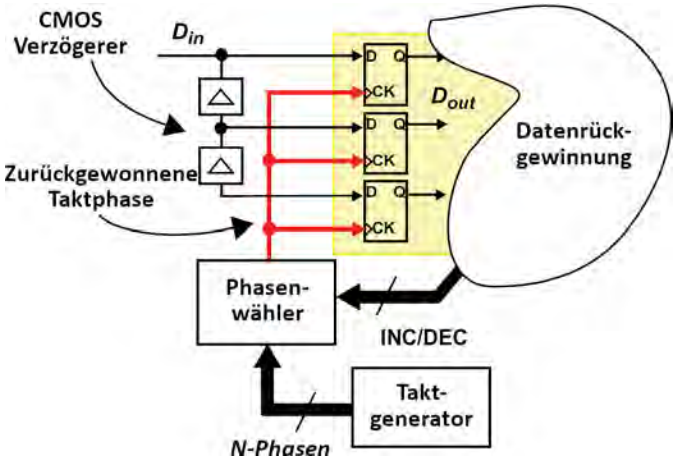


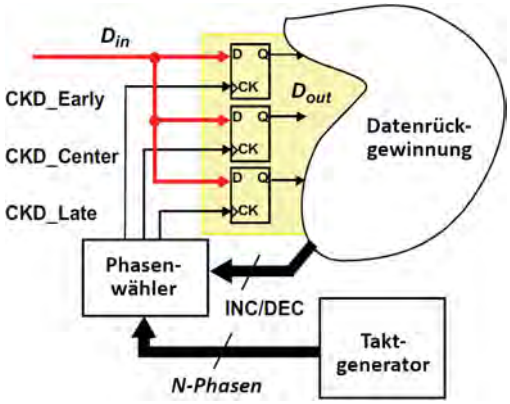
Abbildung 4.7: Funktionsblöcke der Implementation [18, S. 614]

Der Phasenkomparator (PC) gewinnt zum einen das Bit mit der gewählten Taktphase zurück und zum anderen arbeitet dieser als Phasendetektor und treibt die beiden Steuerungssignale UP/DOWN, die zur Neuwahl der Taktphase dienen. Es wird ein Sampler mit dreifacher Überabtastung des empfangenen Signals für die korrekte Funktion des Phasen-

detektors implementiert. Es gibt zwei unterschiedliche Ansätze zum Aufbau des Samplers, welche in Abbildung 4.8 gezeigt sind. Der ankommende Datenstrom D_{in} wird im Ansatz nach [18] auf drei Flipflops geführt, jedoch ist der Pfad zwischen dem ersten zum zweiten und vom zweiten zum dritten Flipflop mit einem CMOS-Verzögerungselement versehen. Die Verzögerungsdauer beträgt $0,25$ UI. Dadurch lässt sich mit einer positiven Taktflanke an allen drei Flipflops eine dreifache Überabtastung realisieren. In [19] werden die CMOS-Verzögerungselmente durch eine Abtastung mit den drei Taktphasen *CKD_Early*, *CKD_Center* und *CKD_Late* ersetzt, die bereits im System zur Verfügung stehen. Die genutzten Taktphasen sind um 90° phasenverschoben, womit das gleiche Ziel der dreifachen Überabtastung in einem Abstand von $0,25$ UI erreicht wird. Hierbei bildet das mittlere Sample das zurückgewonnene Bit und wird durch die gewählte Taktphase *CKD_Center* repräsentiert. [18]



(a) Sampler nach [18]



(b) Sampler nach [19]

Abbildung 4.8: Funktionsprinzip beider Sampler, modifiziert nach [19, S. 112]

Der verwendete Phasendetektor wendet das Prinzip des *Alexander-Hogge*-Phasendetektors an, welches auch als *early-late*-Detektion beschrieben wird. Die Abbildung 4.9 zeigt die *early* und *late* Detektion mit einem Takt CK zur Abtastung des Datenstromes D_{in}. Der Phasendetektor ermittelt, ob eine Datentransition auftritt und ob die gewählte Taktphase in der Mitte des Datenauges liegt. Sind alle Samples gleich wird kein Steuerungssignal getrieben, die gewählte Taktphase liegt in der Mitte des Datenauges. Ist das erste Sample S_1 ungleich den anderen beiden Samples ist die gewählte Taktphase nicht in der Mitte des Datenauges und die Abtastung erfolgt zu früh (*early*). Sind dagegen die ersten beiden Samples S_1 und S_2 gleich und das dritte Sample S_3 unterscheidet sich, erfolgt die Abtastung zu spät (*late*). Unterscheidet sich das zweite bzw. mittlere Sample S_2 vom ersten und dritten Sample ist von einem Einbruch der Datenübertragung, Störungen oder einer nicht erfolgreichen dreifachen Überabtastung auszugehen. Um die entsprechenden Zustände zu ermitteln, wird das erste und zweite Sample über ein XOR-Gatter verknüpft und das dritte mit dem zweiten Sample ebenfalls. Der implementierte Phasendetektor ist in Abbildung 4.10 gezeigt. Ist die gewählte Taktphase zu früh, wird das UP-Signal gesetzt, ist dagegen der Takt zu spät wird das DOWN-Signal gesetzt. Der Phasendetektor kann die Position des Datenauges nur bestimmen, wenn Transitionen in den Daten auftreten. Dementsprechend ist ein Leitungscode zu wählen, der lang anhaltende Pegel über mehrere Bits vermeidet. [22]

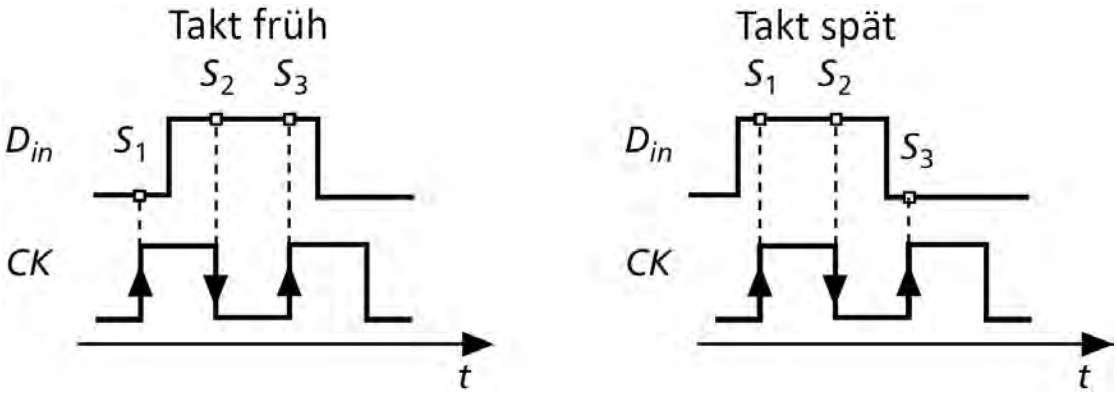


Abbildung 4.9: Darstellung des *early*- und *late*-Zustandes, modifiziert nach [22, S. 96]

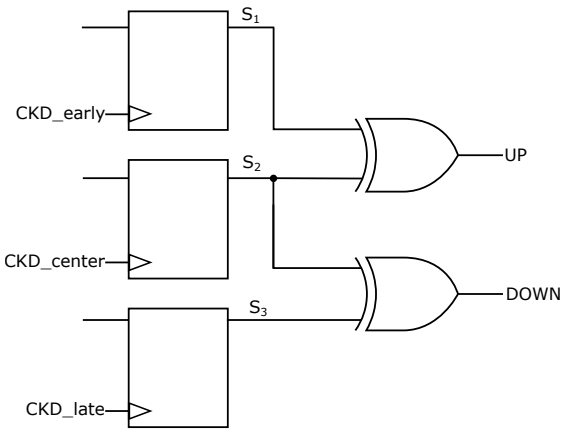


Abbildung 4.10: Implementierter Phasendetektor

Der zweite Block der CDR ist der UP/DOWN Entscheider (DC), der den Schleifenfilter der Schaltung bildet. Das UP- und DOWN-Signal vom Phasenkomparator werden in jeweils separate Schieberegister mit NS-Flipflops geschoben und über folgende bool'sche Gleichungen werden die Ausgangssignale INC und DEC ermittelt:

$$INC = \{UP(0) + ... + UP(NS)\} \cdot \left\{\overline{DOWN(0) + ... + DOWN(NS)}\right\} \tag{4.2}$$

$$DEC = \left\{\overline{UP(0) + ... + UP(NS)}\right\} \cdot \{DOWN(0) + ... + DOWN(NS)\} \tag{4.3}$$

Erfasst der Schleifenfilter ausschließlich UP-Signale wird das Kontrollsignal INC gesetzt und werden hingegen nur DOWN-Signale erfasst, setzt der Entscheider das Kontrollsignal DEC. Treten sowohl UP- als auch DOWN-Signale auf, wird kein Kontrollsignal gesetzt. Die Länge der Schieberegister NS ist frei wählbar und wird in [22] zur Einsparung von Hardwareressourcen mit zwei vier-bitlangen Schieberegistern realisiert, die auf der halben Taktfrequenz arbeiten.

Der dritte Block ist der zyklische Taktphasenzeiger (engl. clock-phase-pointer) (CPP), welcher die Taktphase zur Rückgewinnung der Daten anhand der INC/DEC-Signale bestimmt. Erhält der CPP ein INC-Signal, wird die Taktphase zur Rückgewinnung um einen Schritt verzögert, tritt dagegen ein DEC-Signal auf, wird die Taktphase um einen Schritt vorgezogen, was jeweils die Wahl der benachbarten Taktphase um $\pm 45°$ bedeutet. Durch die zyklische Anordnung ist es möglich eine unendliche Taktwanderung zu verfolgen, da sich die Taktphase unendlich oft anpassen lässt, über ein UI hinaus. Der CPP

arbeitet auf einem um den Faktor ND geteilten Takt. Dies hat den Effekt, dass auf kleinste oder hochfrequente Störungen nicht reagiert wird, sondern nur auf eine kontinuierliche Wanderung der Phase. ND beschreibt die Anzahl der Bits, die nach dem Empfangen eine Neuwahl der Taktphase auslösen und ist in der vorgestellten Implementation ND = 16 gewählt. [18]

Die Abbildung 4.11 zeigt den prinzipiellen Entscheidungsvorgang der CDR mit den acht Taktphasen am Beispiel eines abgetasteten Datenauges. Als gewählte Taktphase zur Rückgewinnung der Daten sei die Taktphase gewählt, welche 135° phasenverschoben ist und bereits sehr nah an den Transitionen im Datenauge positioniert ist. Für den internen Schaltvorgang des Phasendetektors sind bei gewählter Taktphase die rot gekennzeichneten Taktphasen und Samples interessant. Da das Sample der 225°-Taktphase hinter der Transition im Datenauge liegt, ermittelt der Phasendektor, dass die gewählte Phase vergleichsweise spät ist. Es wird eine Verkürzung der Taktphase gefordert, zur Mitte des Datenauges, was in der Neuwahl des 90°-Taktes resultiert. Nun sind die blau gekennzeichneten Taktphasen und Samples für den Phasenkomparator interessant. Da die Taktphase weiterhin zum Teil auf den Transitionen des Auges und dahinter liegt, fordert der Phasenkomparator eine erneute Verkürzung der aktuellen Taktphase, was die Neuwahl vom 45°-Takt zur Rückgewinnung der Daten zur Folge hat.

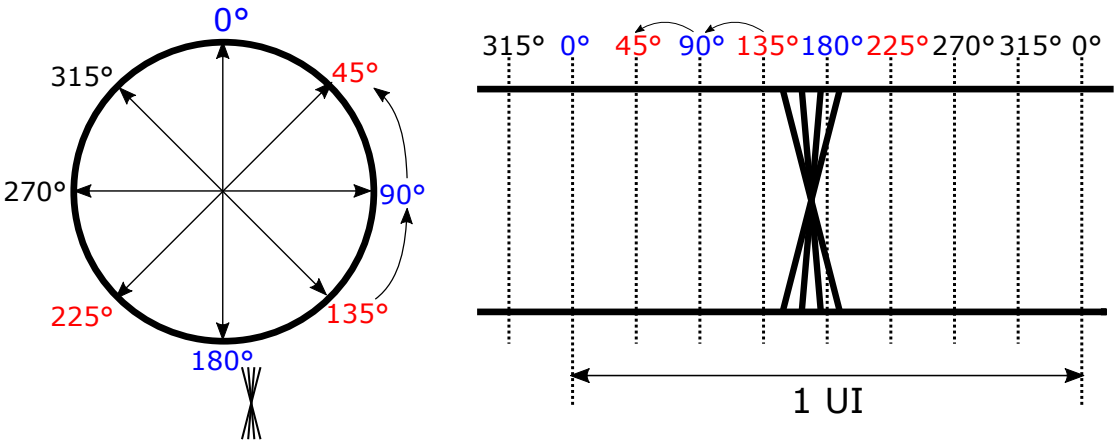


Abbildung 4.11: Betrachtung des Datenauges mit 8 Taktphasen und Auswahl zur Rückgewinnung bei zunächst aktivem 135°-Takt

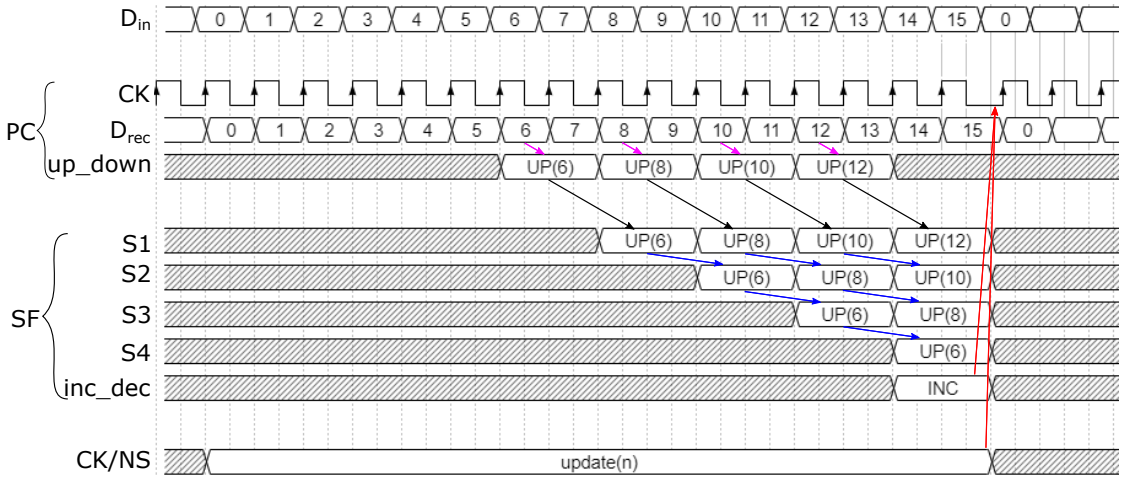


Abbildung 4.12: Zeitlicher Signalverlauf wichtiger Signale

Ein beispielhafter, zeitlicher Signalverlauf der wichtigsten Kontrollsignale ist in Abbildung 4.12 dargestellt. Die Länge des Schieberegisters im Schleifenfilter ist NS = 4 und der CPP wird alle ND = 16 Bits aktualisiert. Das Signal der gewählten Taktphase *CK* spiegelt nur den zeitlichen Verlauf des durch den CPP ausgewählten Takt wieder. Es sind alle acht Taktphasen weiterhin aktiv und eine Phasenverschiebung von *CK* im Diagramm ist im realen System die Wahl einer neuen Taktphase, welche sich modellhaft als eine Phasenverschiebung des Taktes *CK* äußert. Es werden durch die gewählte Taktphase *CK* die ankommenden Daten D_{in} durch den Phasenkomparator abgetastet und zurückgewonnen. Der Phasenkomparator treibt das UP-Signal, da die gewählte Taktphase nahe an den Übergangen der Eingangsdaten ist und somit früh. Das UP-Signal wird nur alle zwei Taktperioden aktualisiert und durch die begrenzte Länge der Schieberegister im Schleifenfilter sind nur die vier erzeugten relevanten UP-Signale vor der Aktualisierung des CPP eingezeichnet. Der Index der UP-Signale gibt an, aus welcher Samplenummer das Signal durch den Phasendetektor erzeugt wurde, gekennzeichnet durch die rosafarbigen Pfade. Die einzelnen Fliflops der Schieberegisters sind als $S1 - S4$ bezeichnet. Die schwarzen Pfade zeigen die Übernahme der UP-Signale des Schleifenfilters vom Phasenkomparator an und durch die blauen Pfade wird das Durchschieben der Signale im Schieberegister verdeutlicht, bis schließlich vier UP-Signale im Schieberegister vorhanden sind und das INC-Signal gesetzt werden kann. Daraufhin wird im nächsten Aktualisierungszyklus die Taktphase zur Rückgewinnung verzögert und in die Mitte des Auges bewegt, gekennzeichnet durch den roten Pfad.

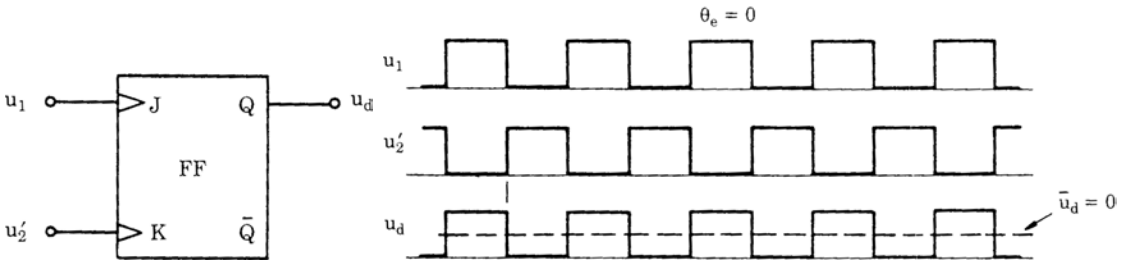
Die maximal mögliche Datenrate wird auf einem FPGA durch die erzeugbaren phasenverschobenen Takte beschränkt. Auf dem Spartan-6 ist es möglich acht Takt mit einer Frequenz von bis zu 200 MHz mit fester Phasenbeziehung zu erzeugen [23, 5], was die maximal mögliche Datenrate auf 200 Mbps beschränkt. Des Weiteren beschränkt sich der Hardwareaufwand auf einige logische Funktionen, Flipflops und LUTs. Es ist jedoch notwendig mehrere Takte zu erzeugen, was einen höheren Energiebedarf bedeutet. Ebenso lassen sich auf einem FPGA mehrere Taktsignale nur umständlich multiplexen und benötigen spezielle Ressourcen, die nicht auf allen FPGAs vorhanden sind. Da das zu entwickelnde System portierbar sein soll, ist auf den Gebrauch spezieller Hardwareressourcen zu verzichten. Werden alle Operationen mit allen Takten immer parallel ausgeführt und nur der gewünschte Ausgang der einzelnen Zweige ausgewählt, kann das Multiplexen von Taktsignalen umgangen werden. Jedoch führt dies zu einem höheren Hardwareaufwand und pro Taktdomäne wird ein vollständiger Pfad benötigt. Des Weiteren hat eine begrenzte Impulsverkürzung in den gesendeten Daten keinen signifikaten Einfluss auf die CDR, da der Schleifenfilter einen ständige Neuwahl der Taktphase verhindert und damit das Heraussprungen der gewählten Taktphase aus der Mitte des Auges.

4.3.3 PLL-basierte CDR (ADPLL)

Eine CDR mit einer PLL lässt sich auf einem FPGA durch das Fehlen analoger Bausteine auf dem Chip nur als All-Digital-Phase-Locked-Loop (ADPLL) realisieren. Dementsprechend werden der Phasendetektor, Schleifenfilter und Oszillator als digitale Bausteine realisiert.

Digitaler Phasendetektor

Ein JK-Flipflop lässt sich als digitaler Phasendetektor verwenden, gezeigt in Abbildung 4.13a. Die empfangenen Daten werden auf den J-Eingang gelegt und mit dem am K-Eingang anliegenden Takt verglichen. Eine positive Flanke am J-Eingang setzt den Ausgang Q auf „1“, eine positive Kante am K-Eingang setzt den Ausgang Q auf „0“. Ein Phasenfehler $\phi_e = 0$ wird dann erzeugt, wenn beide Takte 180° phasenverschoben sind, wodurch sich direkt mit einer Taktflanke des zurückgewonnen Taktes ein Bit zurückgewinnen lässt, da es in der Mitte des Datenauges liegt. Andere Phasendetektoren wie beispielsweise ein XOR-Gattern würden einen Phasenfehler $\phi_e = 0$ erzeugen, wenn beide Eingangssignale um 0° phasenverschoben sind und sich damit nicht für die direkte Abtastung des Datenstromes eignen. Die Abbildung 4.13b zeigt den Signalverlauf der Eingangssignale und des Ausgangssignals bei einer Phasenbeziehung von 180°. [24, S. 17-18]



(a) (b)

Abbildung 4.13: Blockschaltbild und beispielhafter zeitlicher Signalverlauf eines JK-Flipflops [24, S. 17]

Digitaler Schleifenfilter

Als digitaler Schleifenfilter bietet sich in Verbindung mit dem JK-Flipflop als Phasendetektor ein K-Zähler an. Das Blockschaltbild ist in Abbildung 4.14 dargestellt. Der K-Zähler besteht aus zwei unabhängigen Zählern, welche als „UP-Zähler“ und „DOWN-Zähler“ bezeichnet werden. K bildet hierbei den Modulo Wert beider Zähler und ist ein Wert einer Zweierpotenz. Die Frequenz des Operationstaktes *K-Takt* des K-Zählers ist um den Faktor M höher als die Zentralfrequenz f_0 der ADPLL. Durch das $DN/\overline{UP}$ Signal wird die Operation des K-Zählers gesteuert, so ist der UP-Zähler bei einem hohen Signalpegel aktiv und der DOWN-Zähler steht still. Dagegen ist bei einem niedrigen logischen Pegel der DOWN-Zähler aktiv und der UP-Zähler bleibt stehen. Beide Zähler laufen über, wenn ihr Maximalwert erreicht ist und das Most Significant Bit (MSB) beider Zähler bilden entweder den *Carry*-Ausgang (UP-Zähler) oder den *Borrow*-Ausgang (DOWN-Zähler). Dementsprechend wird der Ausgang eines jeden Zählers gesetzt, wenn dieser die Hälfte seines Maximalwertes erreicht hat.

Die Abbildung 4.15 zeigt den Signalverlauf eines K-Zählers in Verbindung mit einem JK-Flipflops als Phasendetektor bei der Zentralfrequenz der ADPLL. Der Phasendetektor treibt das UP/DN Signal und erzeugt bei einem Phasenfehler $\phi_e = 0$ einen Rechteckpuls mit 50 % *duty-cycle*. Der Operationstakt des K-Zählers ist 16-mal höher als die Zentralfrequenz der ADPLL und der Zähler K ist auf den Maximalwert acht gesetzt. Unter gegebenen Bedingungen ist der UP-Zähler acht Takte aktiv und anschließend der DOWN-Zähler für weitere acht Takte. Beide Zähler erzeugen so einen Puls für den Carry- und Burrow-Ausgang. Bei einem Phasenfehler $\phi_e \neq 0$ würden die Pulse eines Ausganges dominieren. [24, S. 212-214]

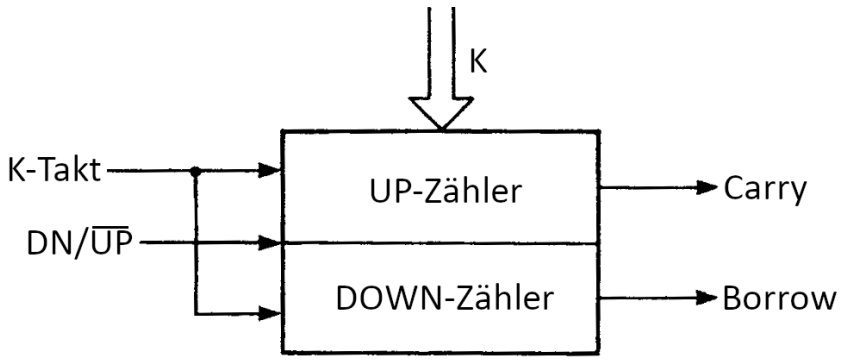


Abbildung 4.14: Blockschaltbild eines K-Zählers, modifiziert nach [24, S. 213]

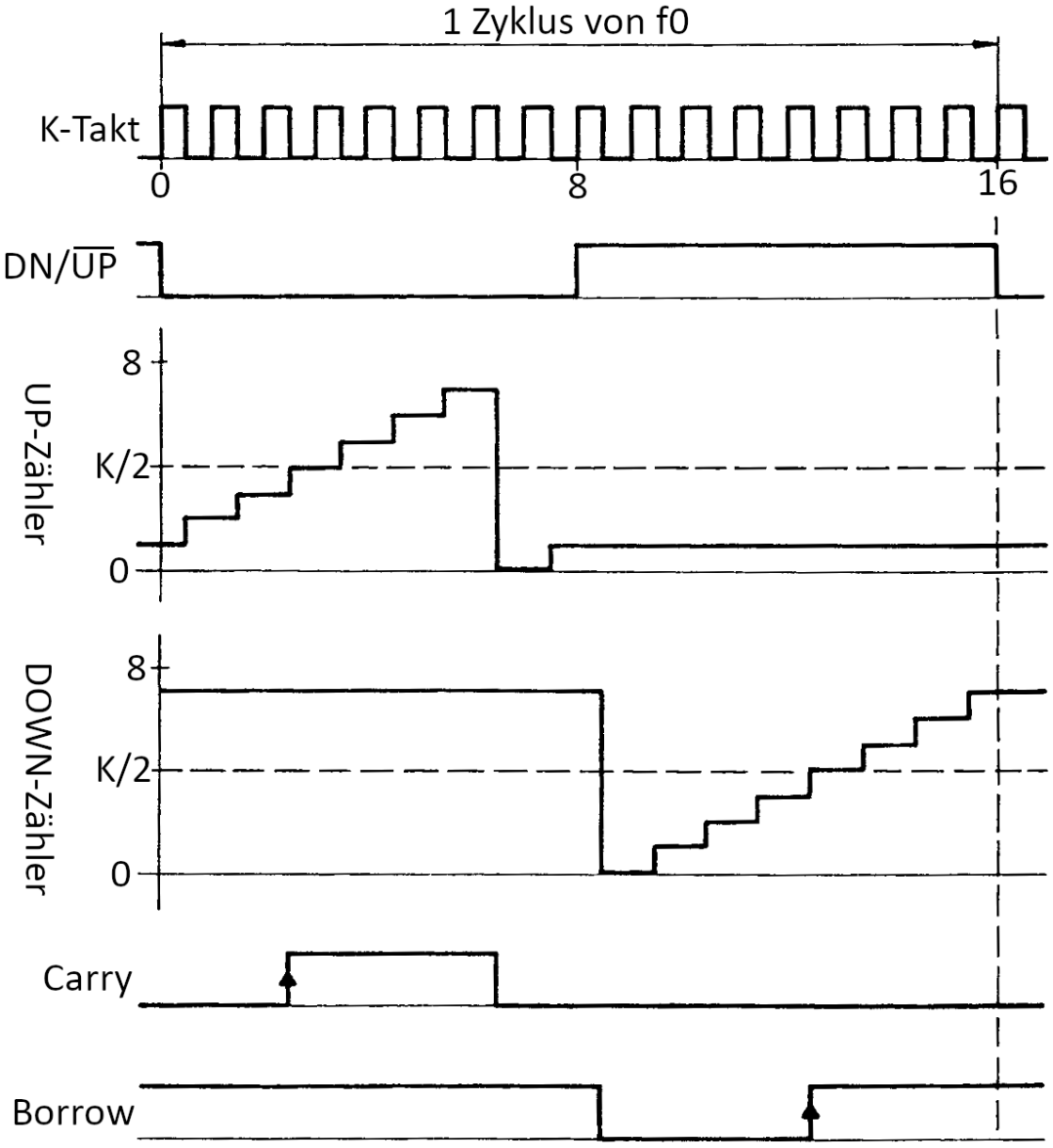


Abbildung 4.15: Beispielhafter Signalverlauf eines K-Zählers, modifiziert nach [24, S. 213]

Digital kontrollierter Oszillator

Der Inkrement-Dekrement-Zähler (ID-Zähler) ist ein digital kontrollierter Ozsillator (engl. digital controlled oscillator) (DCO), dargestellt als Blockschaltbild in der Abbildung 4.16. Aufgrund seiner Eingänge bietet sich dieser für die direkte Anwendung mit einem K-Zähler als Schleifenfilter an. Das Carry-Signal des K-Zählers wird als INC-Signal in den ID-Zähler geführt, das Borrow-Signal treibt den DEC-Eingang und der ID-Takt Eingang wird mit einem Referenztakt getrieben. Der INC- und DEC-Eingang reagiert nur auf steigende Flanken und ist nicht sensitiv für die Dauer der einzelnen Pulse. Treten keine Flanken an den beiden Eingängen auf, gibt der ID-Zähler am Ausgang einen Takt mit halber Frequenz des Referenztaktes aus, dargestellt durch Abbildung 4.17a. Ein internes Toggle-Flipflop verarbeitet dabei die Carry- und Borrow-Pulse und steuert den Taktausgang ID_{out} des DCOs. Das Toggle-Flipflop wird bei jeder positiven Kante des Referenztaktes beschrieben und der Ausgang wird über folgende bool'sche Gleichung bestimmt:

$$ID_{out} = \overline{ID_{Takt}} \cdot \overline{Toggle - Flipflop} \tag{4.4}$$

Ein Carry-Puls wird verarbeitet, wenn das Toggle-Flipflop im Zustand *high* ist und bleibt danach zwei Takte auf *low*, was einen weiteren ID-Out Puls erzeugt und die Frequenz erhöht. Ein beispielhafter Signalverlauf ist durch Abbildung 4.17b gezeigt. Dagegen wird ein Borrow-Puls verarbeitet, wenn das Toggle-Flipflop im Zustand *low* ist und bleibt danach zwei Takte im Zustand *high*. Dies verzögert einen ID-Out Puls und die Frequenz wird verringert. Aus dieser Funktionsweise ergibt sich ein möglicher Frequenzbereich von einem Drittel bis zwei Drittel des Referenztaktes, der von der ADPLL abgedeckt werden kann. [24, S. 216-219]

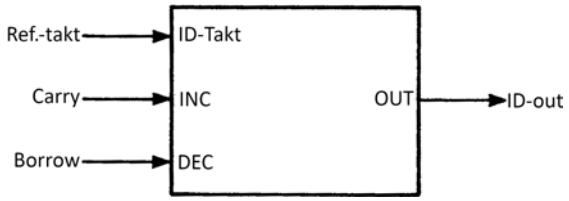
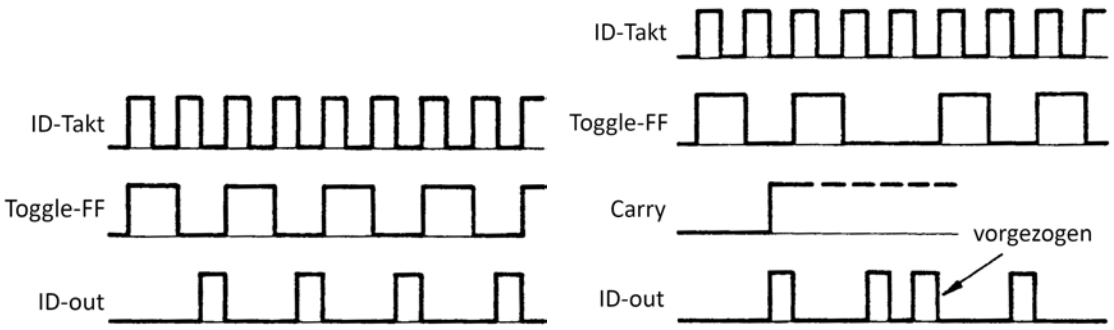


Abbildung 4.16: Blockschaltbild eines ID-Zählers, modifiziert nach [24, S. 217]



(a) Signalverlauf ohne *Carry* oder *Borrow* Pulse

(b) Signalverlauf mit *Carry* Puls

Abbildung 4.17: Signalverläufe eines ID-Zählers in unterschiedlichen Situationen, modifiziert nach [24, S. 217]

Implementation auf einem FPGA

Auf einem FPGA wird der mögliche zu realisierende Frequenzbereich weiter eingeschränkt. Als Referenztakt bietet sich der schnellst mögliche erzeugbare Takt auf dem FPGA an, welcher auf dem gewählten Spartan-6 375 MHz beträgt [5]. Dieser ist in der Regel nur auf die direkte Verbindung synchroner Logik anzuwenden, ohne große kombinatorische Pfade, die durch längere Laufzeiten die Setup-and-Hold-Zeit verletzen können. Das Toggle-Flipflop wird zur richtigen Verarbeitung der Carry- und Borrow- Pulse über kombinatorische Logik modelliert, was bereits zur Verringerung der maximal möglichen Frequenz des Referenztaktes führen kann. Des Weiteren wird der Ausgangstakt direkt über kombinatorische Logik erzeugt, was einen weiteren kombinatorischen Pfad erzeugt und allgemein gilt die Erzeugung von Taktsignalen aus kombinatorischer Logik in einem FPGA als *bad-practice*. Wird das beste Szenario betrachtet, das heißt die Verzögerungen durch die kombinatorische Logik werden vernachlässigt, lässt sich bei einem maximalen Referenztakt $f_{\mathrm{Refmax}} = 375\,\mathrm{MHz}$ folgender maximaler Ausgangstakt f_{out} erzeugen:

$$f_{\mathrm{out}} = \frac{2}{3} \cdot f_{\mathrm{Refmax}} \tag{4.5}$$

$$f_{\mathrm{out}} = \frac{2}{3} \cdot 375\,\mathrm{MHz} \tag{4.6}$$

$$\approx 247\,\mathrm{MHz} \tag{4.7}$$

Ist es möglich die ADPLL mit diesem Referenztakt zu realisieren, kann der Operationstakt des K-Zählers nur doppelt so schnell gewählt werden, wie die Mittenfrequenz der ADPLL. Das schränkt die Schleifenfiltercharakteristik ein und macht die ADPLL empfindlicher für unterschiedliche Störungen wie beispielsweise hochfrequenten Jitter oder eine Impulsverkürzung der Empfangsdaten durch einen aufgeladenen Kanal. Jedoch ermöglicht eine ADPLL eine sehr schnelle Akquirierungszeit des Taktes und damit eine sehr schnelle Rückgewinnung der Informationen. Des Weiteren lässt sich ein größerer Bereich unterschiedlicher Datenraten abdecken, was den Einsatz der ADPLL in unterschiedlichen Kommunikationssystemen ohne Neukonfiguration ermöglicht und das System kann sehr gut auf Taktwanderungen reagieren. Außerdem kann es bei der Übertragung von einem logischen Pegel über mehrere Bits dazuführen, dass der Phasendetektor die Phase durch fehlende Transitionsübergange verliert. Anschließend ist es möglich, dass die PLL den zurückgewonnen Takt komplett verliert und eine Datenrückgewinnung so unmöglich wird. Durch den Einsatz eines Leitungscodes, wie der 8b10b Code im angestrebten Produkt oder eine Modulation, wird dies jedoch verhindert.

4.4 Zusammenfassung und Auswahl

Eine CDR auf Grundlage einer ADPLL ist durch ihre Regelschleifencharakteristik resistent gegen Eingangsjitter und bietet eine hohen Fangbereich. Dadurch ist es möglich, dass ein System ohne Rekonfiguration für unterschiedliche Datenraten genutzt werden kann. Jedoch ist eine Implementierung auf einem FPGA mit einigen Schwierigkeiten verbunden. Allen voran ist es die Erzeugung von Taktsignalen aus kombinatorischer Logik, die durch die internen Verdrahtungsressourcen nur mit vergleichweise hohen Signallaufzeiten möglich ist. Das schränkt die maximal erreichbare Frequenz ein und gilt als *bad practice*. Zwar können ADPLLs auf FPGAs realisiert werden, wie beispielsweise in [25] beschrieben, jedoch nur für eine Übertragung bei 125 Mbps mit zweifacher Überabtastung. Des Weiteren ist diese Implementierung für die Xilinx-FPGA-Familie der Serie-7 ausgelegt, welche performantere FPGAs als der Spartan-6 sind. Unter der Betrachtung, dass die CDR im fertigen Produkt auf einem kleinen, ressourcenarmen FPGA realisiert werden soll, ist diese Methode einer CDR für die gewünschte Anwendung nicht geeignet.

Im Gegensatz dazu ist das vorgestellte Konzept einer CDR mit Überabtastung und einer einfachen feed-forward Datenrückgewinnungseinheit sehr einfach auf einem FPGA zu implementieren. Durch den sehr geringen Hardwareaufwand und der kaum benötigten

kombinatorischen Logik sind zudem sehr hohe Taktraten auf dem FPGA zu erreichen, was ebenfalls sehr hohe Datenraten ermöglichen würde. Jedoch zeigen die Literaturergebnisse keine signifikante Verbesserung gegenüber konventionellen Methoden.

Eine CDR nach dem Vorbild des vorgestellten Phaseninterpolators bietet dahingehend deutlich bessere Ergebnisse. Des Weiteren wurde das Konzept bereits erfolgreich in einem ASIC realisiert und erfüllt die standardisierte Spezifikation SIL-5 [21]. Durch einige Anpassungen ist die CDR ebenfalls auf einem FPGA zu realisieren. Der Ansatz einer CDR mit Phaseninterpolator nach dem Vorbild der MCRP-Architektur wird daher in dieser Arbeit weiter verfolgt und auf einem FPGA implementiert. Die Tabelle 4.1 zeigt eine kurze Übersicht über die vorgestellten Methoden und deren Vor- und Nachteile.

Tabelle 4.1: Zusammenfassung der vorgestellten Algorithmen

Parameter	**BO-DRU**	**MCRP**	**ADPLL**
max. Datenrate	200 Mbps[1]	200 Mbps[1]	245 Mbps[2]
Überabtastung	achtfach	achtfach	einfach
Akquisitionszeit	sofort	nach ND-Bits	nach $5 - 10$ Bits
Impulsverkürzung	resistent	resistent	anfällig
Hardwareaufwand	sehr gering kaum komb. Logik	gering komb. Logik viele Schieberegister	komplex viel komb. Logik
Ergebnisse aus Literatur	zeigt nicht gewünschte Performance	sehr gut SIL-5 [21]	State of the Art
Jitter	resistent gegen Eingangsjitter	resistent gegen Eingangsjitter	resistent gegen Eingangsjitter, interne Jitterfortpflanzung
Taktwanderung	nicht erfassbar	erfassbar	erfassbar

[1] mit interner Erzeugung von acht Taktphasen auf dem Spartan-6

[2] Best-Case Szenario

5 Umsetzung

5.1 Gesamtkonzept für den digitalen Entwurf

5.1.1 Übersicht

Um das System der MCRP-Architektur auf einem FPGA zu implementieren sind einige Modifikationen an dem bestehenden Konzept vorzunehmen, welche im folgenden Abschnitt beschrieben werden. Es wird wie in [18] die CDR mit insgesamt acht Taktphasen realisiert. Das Blockschaltbild in Abbildung 5.1 zeigt die umzusetzende CDR auf VHDL-Level. So besteht die CDR *cdr_top* aus den Teilmodulen des Phasekomparators *phase_comp*, des Schleifenfilters *loopfilter*, des CPPs *clock_phase_pointer* und eines Taktteilers *clockdivide_enable*. Zusätzlich kann die CDR mit den drei generischen Parametern *NS_BITS*, *BITS_OUT_OF_LOCK* und *DIVIDER_BITS_CPP* schnell rekonfiguriert und optimiert werden. Die wichtigsten Signalpfade sind farblich gekennzeichnet und werden in den nachfolgenden Abschnitten näher erläutert. Die Tabelle 5.1 beschreibt die benötigten Taktphasen, deren Phasenbeziehung und Bezeichner für die nachfolgenden Abschnitte. Als umzusetzende Datenrate ist 125 Mbps gewählt, da die optischen Frontends für diese Datenrate ausgelegt sind, womit die acht Taktphasen mit einer Frequenz von 125 MHz erzeugt werden müssen. Signale werden stets vom MSB zum Least Significant Bit (LSB) nummeriert.

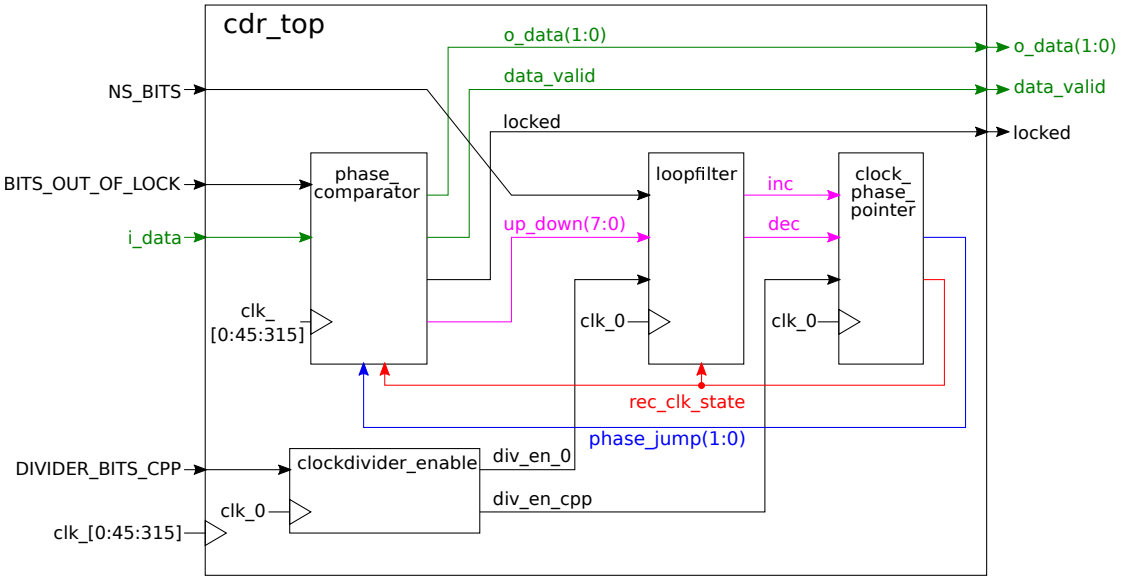


Abbildung 5.1: Blockschaltbild der gesamten CDR mit VHDL-Dateiteilmodulen

Tabelle 5.1: Nummerierung und Phasenbeziehung der acht benötigten Takte

Nummerierung der Taktphase	Phasenwinkel	Bezeichner
0	0°	clk_0
1	45°	clk_45
2	90°	clk_90
3	135°	clk_135
4	180°	clk_180
5	225°	clk_225
6	270°	clk_270
7	315°	clk_315

5.1.2 Clock-Phase-Pointer

Der Taktphasenzeiger (engl. clock-phase-pointer) (CPP) wählt die Taktphase zur Rückgewinnung der Daten, das Interface ist in Abbildung 5.2 dargestellt. Der CPP wird als getakteter, zyklischer, endlicher Zustandsautomat (engl. finite-state-machine) (FSM) modelliert, dessen Zustandsdiagramm in Abbildung 5.3 gezeigt ist. Die acht endlichen Zustände ergeben sich aus den acht möglichen Taktphasen, die zur Rückgewinnung ausgewählt werden können. Die Ausgangssignale des Automaten sind der aktuelle 3-Bit kodierte Zustand *rec_clk_state* der FSM und ein 2-Bit breites Steuerungssignal *phase_jump* zum Anzeigen von kompletten Phasendrehungen. Das Steuerungssignal ist ein Mealy-Ausgang, mithilfe dessen der Phasenkomparator bei der Rückgewinnung Bitslip oder Bitstuff verhindert. Das entsprechende Bit wird jeweils für einen Taktzyklus bei einem Übergang der nullten zur siebten Taktphase und umgekehrt gesetzt, im Zustandsdiagramm durch die roten Pfade verdeutlicht. Die beiden Eingangssignale *inc* und *dec* werden direkt durch den Schleifenfilter getrieben und dienen zur Neuwahl der Taktphase, welche zur Rückgewinnung der Daten genutzt wird. Wird eine *inc*-Anfrage erfasst, wird die Taktphase um 45° verzögert. Bei einer aktiven *dec*-Anfrage wird die Taktphase bei der nächsten Aktualisierung um 45° verkürzt. Beide Eingangssignale sollten nicht gleichzeitig aktiv sein, was durch den Schleifenfilter gewährleistet wird. Sind jedoch beide Signale in einem Störfall gleichzeitig aktiv, wird keine Änderung des Zustandes vorgenommen und die FSM verbleibt im aktiven Zustand. Der CPP wird auf einem geteilten Takt der nullten Taktphase betrieben, um einen ständigen Wechsel zwischen zwei Zuständen zu vermeiden und die allgemeine Sensitivität der CDR auf hochfrequente Störungen zu reduzieren. So wird gewährleistet, dass nur auf niederfrequente Taktwanderungen reagiert wird und nicht auf hochfrequenten Jitter mit geringer Amplitude. Ist das *Enable*-Signal *div_en_cpp* aktiv kann mit der nächsten steigenden Taktflanke eine neue Taktphase gewählt werden.

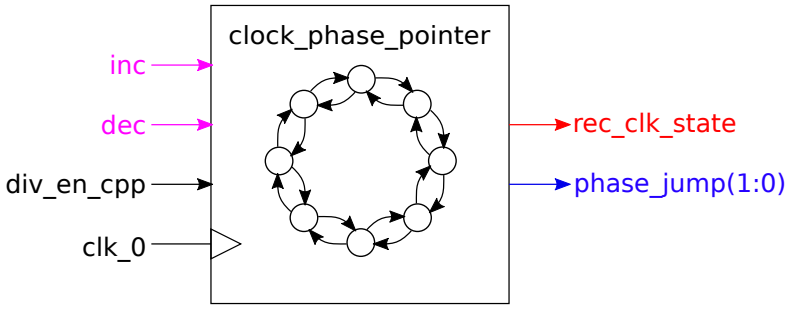


Abbildung 5.2: Interface des CPPs

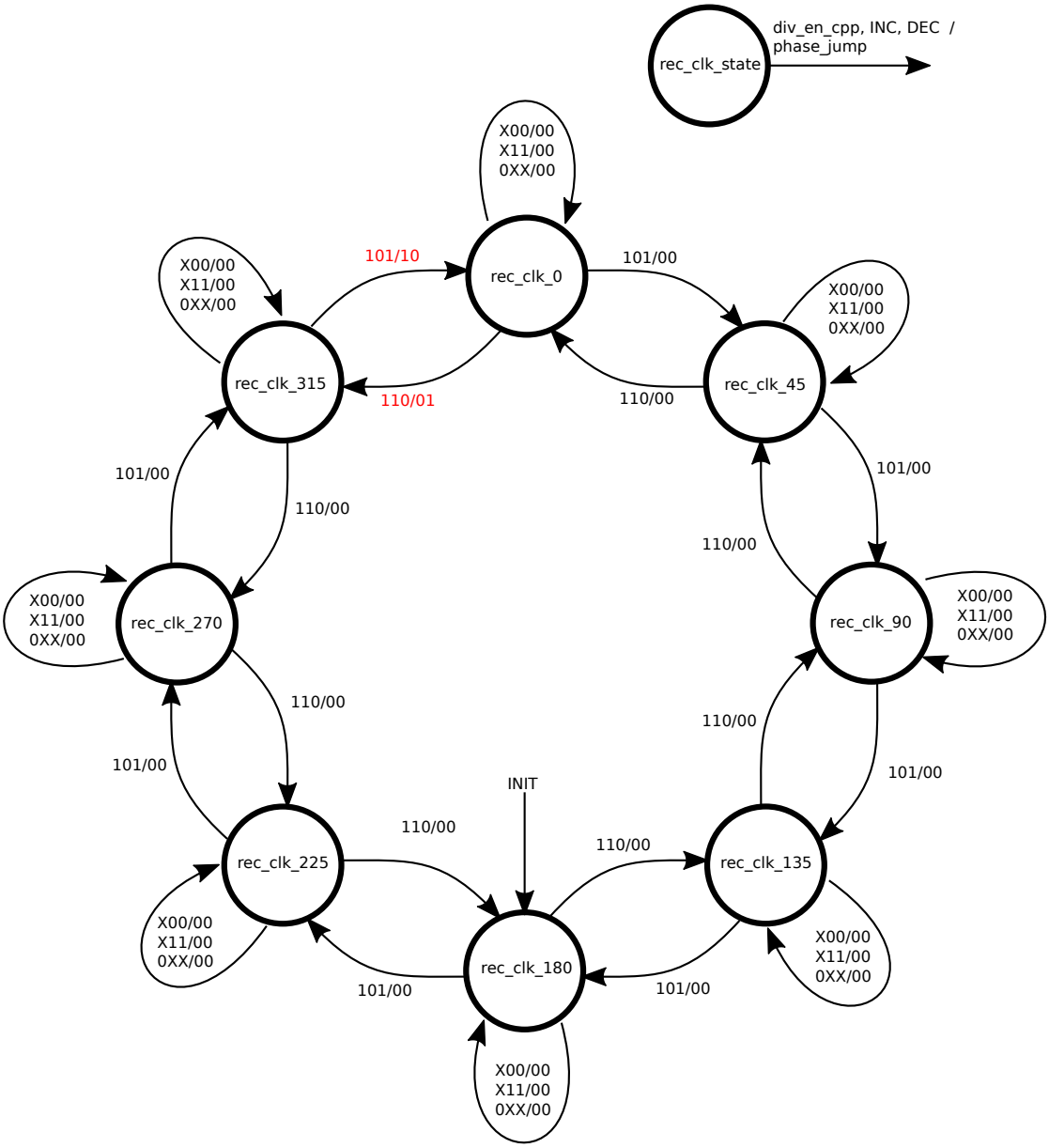


Abbildung 5.3: Zustandsdiagramm des CPP

5.1.3 Phasenkomparator

Der Phasenkomparator ist für die Abtastung des ankommenden seriellen Datensignals und für die Ermittlung der Position des Datenauges zuständig. In Abbildung 5.4 ist das Blockschaltbild des Phasenkomparators gezeigt. Er tastet den seriellen, asynchronen Bitstrom *i_data* mit allen acht verfügbaren Taktphasen ab und gewinnt die Bits *o_data* aus der ausgewählten Taktphase *rec_clk_state* zurück und übergibt diese an das Gesamtsystem. Des Weiteren agiert er als Phasendetektor, gibt mit seinen Ausgangssignal *up_down* Aufschluss über die Position des Datenauges und ob der gewählte Takt zur Rückgewinnung vergleichsweise früh oder spät ist. Ebenso erfasst er, ob eine Datenübertragung stattfindet und setzt dann das *locked*-Signal. Der Phasenkomparator besteht aus unterschiedlichen Teilmodulen, die nachfolgend genauer beschrieben werden.

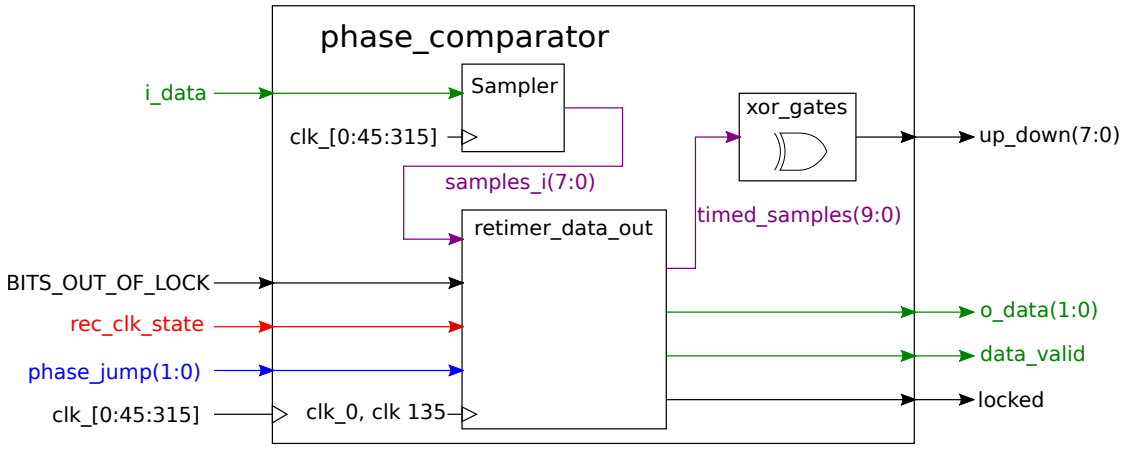


Abbildung 5.4: Blockschaltbild des Phasenkomparators

Sampler

Der in [19] vorgestellte Sampler mit drei Taktphasen wird zur Realisierung auf dem FPGA genutzt, da die benötigten CMOS-Verzögerungselemente auf einem FPGA in der Regel nicht vorhanden sind, für den in [18] gezeigten Ansatz. Um die Setup-and-Hold-Zeiten nicht zu verletzen und eindeutige Signale für die Weiterverarbeitung zu erhalten wird der asynchrone Eingangsdatenstrom zunächst über Synchronisiationsflipflops der jeweiligen Taktdomäne geführt. Das heißt, dass jedes Bit über ein 2-Bit-Schieberegister abgetastet wird. Des Weiteren wird der Sampler mit allen acht Taktphasen realisiert, die stetig den ankommenden Bitstrom abtasten, da das Multiplexen der Taktsignale auf einem

FPGA nicht ohne spezielle Taktbuffer möglich ist. Der zu implementierende Sampler ist in Abbildung 5.5 dargestellt. Dadurch ergibt sich effektiv eine achtfache Überabtastung im Gegensatz zur vorgestellten dreifachen Überabtastung in der Literatur. Jedoch werden weiterhin nur drei Samples zur Ermittlung des Datenauges gewählt, abhängig von der aktuell gewählten Taktphase des CPPs. Die restlichen Samples werden ignoriert, stehen aber bereit, wenn eine neue Taktphase gewählt wird.

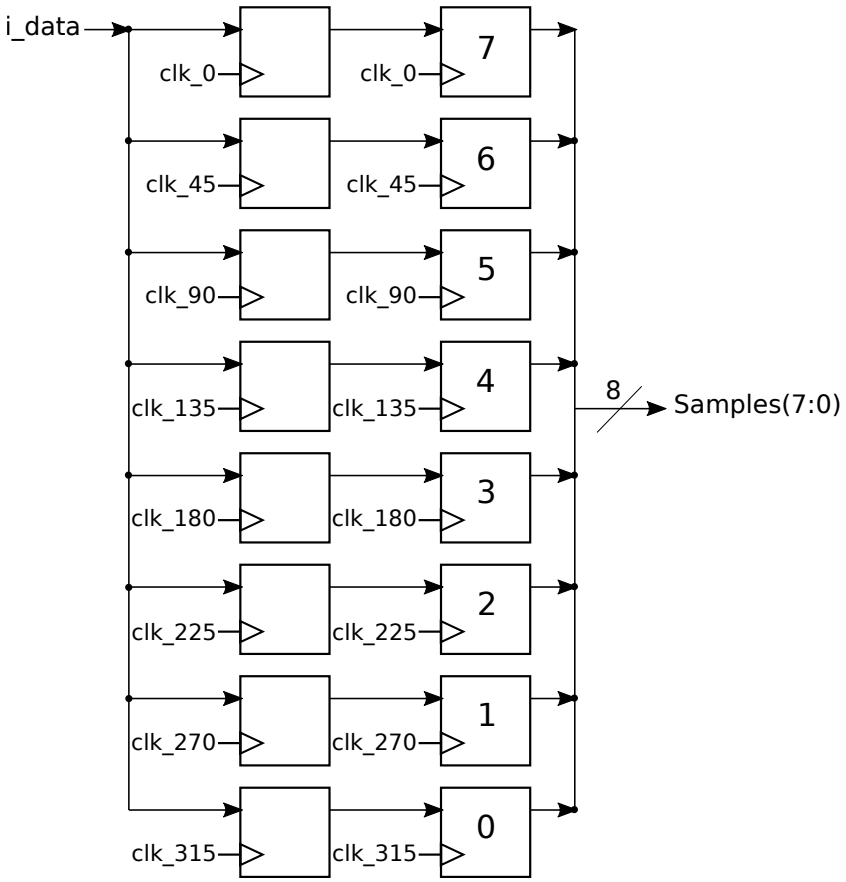


Abbildung 5.5: Sampler mit Synchronisationsflipflops

Synchronisierung der Samples

Damit die restlichen Funktionsblöcke der CDR und des spätere Gesamtsystems auf einer Taktphase betrieben werden können, müssen die gewonnenen Bits mit der selben Taktdomäne synchronisiert werden, vorzugsweise dem Systemtakt. Ebenfalls lässt sich durch die Synchronisierung der Samples die CDR einfacher in andere Systeme integrieren, da die gewonnen Bits bereits zu einem Takt synchron sind und nicht nachträglich im Gesamtsystem synchronisiert werden müssen. Die Synchronisierung der zurückgewonnen Bits übernimmt das Teilmodul *retimer_ data_ out*.

Es wird die nullte-Taktphase des Samplers zur Synchronisierung gewählt, d.h. dass alle Samples der restlichen Taktdomänen nach dem Abtasten mit dieser Taktphase synchronisiert werden. So ist es möglich alle Kontrollsignale ebenfalls synchron zur nullten Taktphase zu erzeugen. Die Synchronisierung der Bits wird durch Schieberegister realisiert, in denen die einzelnen Flipflops auf unterschiedlichen Taktdomänen betrieben werden. Es ist zu beachten, dass alle Samples für die gleiche Dauer verzögert und die Setup-and-Hold Zeiten der Flipflops nicht verletzt werden, um weiterhin eine richtige Funktion der CDR zu gewährleisten. Der gewählte Ansatz ist in Abbildung 5.6 gezeigt. Die Samples der vierten bis siebten Taktphase werden im ersten Schritt mit der dritten Taktphase synchronisiert. Die Samples der ersten bis dritten Taktphase werden mit der nullten Taktphase synchronisiert und das Sample der nullten Taktphase um eine Taktperiode verzögert. Anschließend werden die Samples der vierten und siebten Taktphase auf die nullte-Taktphase synchronisiert und die restlichen Samples um eine weitere Periode verzögert, um den Prozess der Synchronisation abzuschließen. Die Samples der nullten und ersten Taktphase werden nach der ersten Synchronisierungsstufe zusätzlich an den Phasendetektor übergeben, zur korrekten Ermittlung der Kanten in der Übertragung.

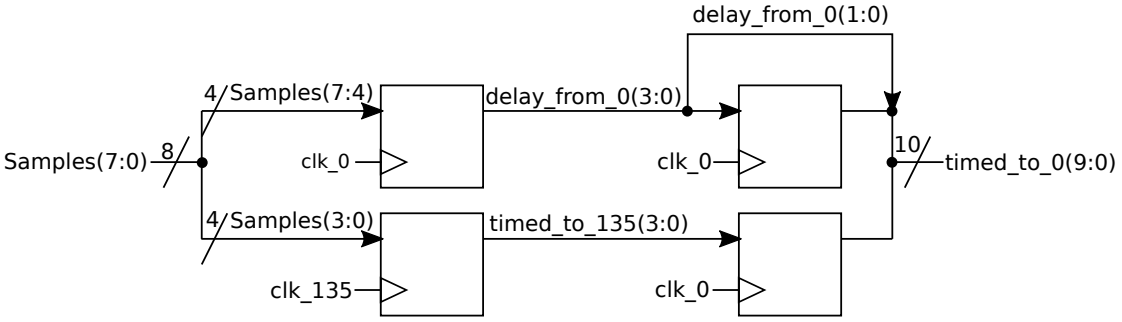


Abbildung 5.6: Synchronisierungsprozess mit Schieberegistern

Ausgabe der Daten

Zusätzlich zur Synchronisation der Samples mit der nullten Taktphase übernimmt das Teilmodul *retimer_data_out* auch die Ausgabe der zurückgewonnen Daten. Da es sich um eine synchrone Datenübertragung unter realen Verhältnissen handelt, kann der System- bzw. Empfangstakt geringfügig schneller oder langsamer sein als der Sendetakt. Dieser Effekt bildet sich als Taktwanderung in der CDR aus und die zurückgewonnene Taktphase wird regelmäßig neugewählt. Bei einem schnelleren Sendetakt wird kontinuierlich die vorangegangene Taktphase gewählt, bei einem langsameren Sendetakt wird die nachfolgende Taktphase gewählt. Des Weiteren bildet sich das Problem, dass der schnellere, serielle Sendedatenstrom auf dem langsameren Systemtakt abgebildet werden muss bzw. der langsamere, serielle Sendedatenstrom auf dem schnelleren Systemtakt. In einem vollduplex Betrieb mit zwei Transceivern, die mit ihrem Systemtakt die Daten über den Kanal senden, ist davon auszugehen, dass in jedem Transceiver jeweils der andere Zustand eintritt, was eine Berücksichtigung beider Zustände erfordert.

Durch die Ausgabe von zwei Bits in Zusammenhang mit einem *valid*-Signal ist es möglich, die beiden Situationen zu berücksichtigen. Im idealen Fall, dass beide Transceiver exakt den gleichen System- und Sendetakt besitzen, wird ein Bit jede Periode zurückgewonnen. Alle zwei Perioden des Systemtaktes werden dann die beiden zurückgewonnen Bits parallel ausgegeben und als gültig gekennzeichnet. Ist der Sendetakt geringfügig schneller als der Systemtakt werden in den meisten Fällen weiterhin alle zwei Taktzyklen die zurückgewonnen Bits als gültig gekennzeichnet. Wechselt die zurückgewonnene Taktphase von der nullten jedoch auf die siebte Taktphase, können zwei Bits innerhalb eines Systemtaktzyklus zurückgewonnen werden, da durch den Synchronisierungsprozess bei diesem Wechsel ein Phasensprung von $-2\,\pi$ durchgeführt wird. Die beiden Bits sind durch die Schieberegister der Synchronisierung bereits erfasst und können direkt ausgegeben werden. Ein Phasensprung von $2\,\pi$ wird bei einem Übergang der siebten Taktphase auf die nullte Taktphase durchgeführt. Dieser Übergang tritt regelmäßig auf, wenn der Sendetakt geringfügig langsamer ist als der Systemtakt. Der Phasensprung kann durch einfaches Abwarten eines Taktzyklus kompensiert werden, da das zurückgewonnene Sample bereits erfasst worden ist und ansonsten doppelt ausgeben werden würde. Die entsprechenden Phasensprünge werden durch den CPP mit dem 2-Bit-Signal *phase_jump* angezeigt.

Die Verzögerung der empfangen Daten bis zur Ausgabe beträgt in dem gewählten Aufbau fünf Taktzyklen. Zunächst werden die Daten über zwei Flipflops in das System einsychronisiert, um anschließend über zwei weitere Flipflops auf die nullte Taktphase

synchronisiert zu werden. Ab diesem Zeitpunkt kann das Sample ausgegeben werden. Im *Worst-Case* kann es jedoch durch die parallele Ausgabe einen weiteren Taktzyklus gespeichert werden, bis das zweite Sample zur Ausgabe bereit steht. Dadurch ergibt sich eine maximal mögliche Verzögerung von fünf Taktzyklen bzw. 40 ns mit einer nominalen Taktperiode von 8 ns bei einer Taktfrequenz von 125 MHz. Diese Verzögerung ist für die späteren echtzeitfähigen Implementierungen zu beachten. Durch einen höheren Sendetakt sinkt auch die absolute, zeitliche Latenz des Systems, da diese sich auf die Anzahl der Taktzyklen bezieht. Bei einem höheren Systemtakt ist zusätzlich die Synchronisation der Samples auf diesen Takt nötig, was ebenfalls zu einer Verringerung der Latenz führen kann. Jedoch ist weiterhin die Einhaltung der Setup-and-Hold Zeit der Flipflops zu beachten und muss je nach gewähltem System zusätzlich betrachtet werden.

Parallel dazu werden die vergangen Taktzyklen gezählt, die seit der letzten Kante in den Daten erfasst wurde, um festzustellen, ob eine Übertragung stattfindet. Dazu werden zwei Samples aus zwei Taktphasen in ein zusätzliches 2-Bit-Schieberegister geschoben und mit dem nachfolgenden Sample über ein XOR-Gatter verknüpft. Wird eine Kante erfasst wird der Zähler zurückgesetzt und das *locked*-Signal gesetzt. Findet keine Datenübertragung statt, werden keine Kanten erfasst und der Zähler erreicht seinen Maximalwert, dann wird das *locked*-Signal wird auf „0“ gesetzt. Das *locked*-Signal dient zur einfachen Überwachung, ob die CDR Daten erhält und arbeitet.

Phasendetektor

Nach der Synchronisation kann durch den Phasendetektor *xor_gates* die Phasenbeziehung der empfangenen Bits zu der gewählten Taktphase ohne Timing-Probleme in den Schaltvorgängen ermittelt werden. Je nach der gewählten Taktphasen zur Rückgewinnung müssen unterschiedliche Samples miteinander verglichen werden.

Der Ansatz der parallelen Pfade wird weiterhin verfolgt und die Samples werden nicht multiplexed, was eine Verknüpfung der Samples mit acht XOR-Gattern notwendig macht, um anschließend den Schleifenfilter mit einem 8-bit breiten *up_down* Eingangssignal zu realisieren. Dieser Ansatz bringt einen vierfach höheren Hardwarebedarf im Schleifenfilter mit sich, jedoch können so direkt die Kontrollsignale für alle Zustände des CPPs berechnet werden und die Sensitivität der CDR auf Taktwanderungen wird erhöht. Die entsprechenden Kontrollsignale sind bei einem Wechsel der zurückgewonnen Phase bereits erzeugt und müssen nicht erst in den Schleifenfilter geschoben und ermittelt werden.

Die logische Verknüpfung der Samples ist in Tabelle 5.2 dargestellt und die Tabelle 5.3 beschreibt den Zusammenhang der zurückgewonnen Taktphase mit den Ausgängen der XOR-Gatter, die schließlich das *up_down*-Signal bilden. Jedes Sample wird mit dem Sample verglichen, dass um $\pm 0{,}5\,\pi$ erfasst wurde. Das hat zur Folge, dass zur Verknüpfung im XOR-Gatter 7 und 0 die synchronisierten Samples, die bereits eine vorherige Instanz auf nullte Taktphase synchronisiert wurden, aus dem Schieberegister verwendet werden. Dies wird durch die Abbildung 5.7 nochmals verdeutlicht. Es ist die Abtastung eines Datenauges zu sehen und die spätere Nummerierung der Samples nach der Synchronisierung auf die nullte Taktphase als *timed_samples*. Im Anhang A.3 ist der implementierte Phasendetektor mit allen acht XOR-Gattern dargestellt.

Tabelle 5.2: Verknüpfung der *timed_samples* zur Phasendetektion

up_down	Verknüpfung *timed_samples*
7	0 xor 8
6	7 xor 5
5	6 xor 4
4	5 xor 3
3	4 xor 2
2	3 xor 1
1	2 xor 0
0	1 xor 9

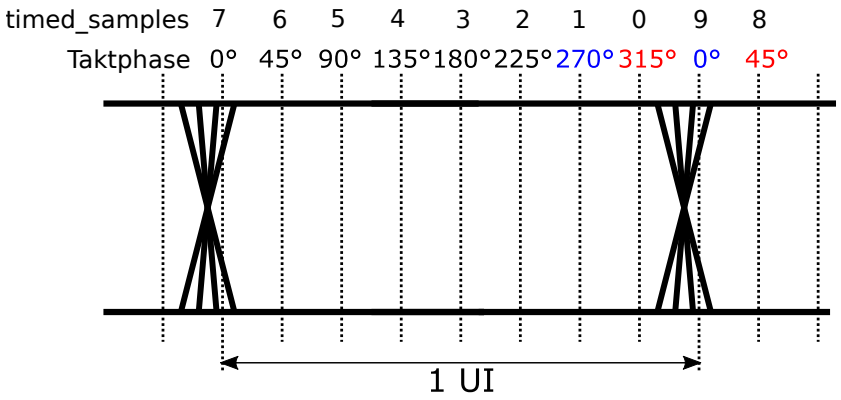


Abbildung 5.7: Abtastung eines Datenauges mit 8 Taktphasen und anschließende Nummerierung nach Synchronisation mit der nullten Taktphase

Tabelle 5.3: Zusammenhang der zurückgewonnen Taktphase mit den Ausgängen des Phasendetektors

rec_clk_phase	XOR-Gatter Zuordnung	
	UP	DOWN
0	0	6
45	7	5
90	6	4
135	5	3
180	4	2
225	3	1
270	2	0
315	1	7

5.1.4 Schleifenfilter

Das Interface des Schleifenfilters *loopfilter* ist in Abbildung 5.8 gezeigt. Der Schleifenfilter besteht aus acht Schieberegistern mit der Länge *NS_BITS*, in die jeweils ein Ausgang eines XOR-Gatter *up_down* des Phasenkomparator geschoben wird. Währenddessen ermittelt kombinatorische Logik den *inc* und *dec* Ausgang nach Gleichung 4.2 und Gleichung 4.3. Es ist zu erwarten, dass der kombinatorische Pfad zur Ermittlung der Ausgangssignale im Schleifenfilter, der kritischste Pfad der gesamten CDR ist. Durch die vielen Eingangssignale der bool'schen Gleichung ist mit mehreren, kaskadierten Logikleveln zur Realisierung im FPGA zu rechnen. Damit die Setup-and-Hold Zeiten der Eingänge des CPPs nicht verletzt werden, ist es durch Pipelining, dem Setzen eines oder mehrerer Flipflops in den kritischen Pfad, möglich diesen Pfad zu verkürzen. Dadurch wird die Errechnung der Ausgangssignale um eine Taktperiode für jedes Pipelinelevel verzögert, was die allgemeine Sensitivität der CDR auf Taktwanderungen und andere Effekte senkt. Ob eine Pipelinestufe notwendig ist, wird bei der statischen Timinganalyse durch die Synthese ersichtlich und kann sich je nach implementiertem Gesamtsystem, Taktrate bzw. Datenrate und gewählter Plattform unterscheiden. Im Schleifenfilter werden alle Schieberegister über die kombinatorische Logik parallel miteinander verbunden, vergleichbar mit der Verknüpfung der parallelen XOR-Gatter im Phasendetektor. Abhängig von der aktuell gewählten Taktphase *rec_clk_state*, wird der Ausgang der kombinatorischen Logik ignoriert oder repräsentiert das *inc* oder *dec* Signal.

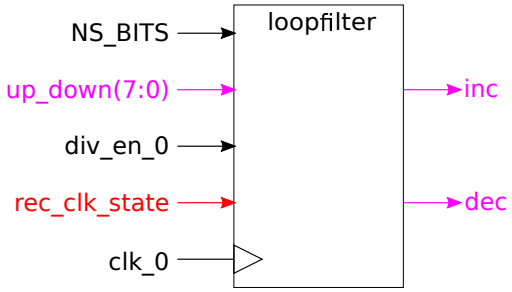


Abbildung 5.8: Interface des Schleifenfilters

5.1.5 Taktteilung und Taktverteilung

Taktteilung

Direkte Taktteilung durch kombinatorische Logik verringert auf einem FPGA die maximal mögliche Taktfrequenz durch den internen Aufbau und deren Verdrahtung. Eine Taktteilung wird auf einem FPGA normalerweise mit einem *enable*-Signal realisiert, welches nach einer gewählten Anzahl an Taktperioden nur für eine Taktperiode aktiv ist und das taktsynchrone Element schalten lässt. Das Teilmodul *clockdivider_enable* erzeugt die entsprechenden enable-Signale für die anderen Teilmodule. Um die gewünschte Taktteilung der einzelnen Komponenten zu realisieren, wird ein Zähler mit der Breite *DIVIDER_BITS_CPP* implementiert, der die Zyklen der nullten Taktphase zählt. Läuft der Zähler über, wird das Signal *div_en_cpp* für einen Takt gesetzt, was das enable-Signal für den CPP ist. Das Signal *div_en_0* ist das enable-Signal für den Schleifenfilter der CDR, welches das LSB des Zählers ist. Es wird jeden zweiten Zyklus der nullten Taktphase aktiv und erreicht damit eine Halbierung der Taktfrequenz.

Taktverteilung

Auf dem verwendeten HuMANDATA Entwicklungsboard ist ein freilaufender Oszillator mit einer Frequenz von 50 MHz mit einem Eingangspin des Spartan-6 verbunden. Dieser Oszillator kann als Referenztaktgeber für unterschiedliche Taktgenerierungsblöcke verwendet werden. Auf dem Spartan-6 sind vier Taktmanager (engl. clock-management-tiles) (CMTs) vorhanden, die zur Generierung von unterschiedlichen Takten auf dem FPGA genutzt werden können. Jedes CMT besitzt eine PLL mit sieben Ausgängen und zwei DCMs mit jeweils vier Ausgängen. Innerhalb eines CMT lassen sich die drei Blöcke

untereinander verbinden und so unterschiedliche Konfigurationen erzielen. So kann eine PLL beide DCMs treiben, und ein DCM kann eine PLL treiben. Ebenso ist es möglich zwei PLLs über zwei CMTs zu kaskadieren, was jedoch die Einstellung bzw. Herstellung einer festen Phasenbeziehung zwischen den Takten beider PLLs nicht ermöglicht. Von Xilinx wird die Verwendung einer PLL empfohlen, wenn unterschiedliche Takte mit unterschiedlichen Frequenzen benötigt werden und die Verwendung von DCMs für eine genaue Phasenverschiebung einzelner Takte. [23]

Da insgesamt acht Phasen einer Taktfrequenz benötigt werden, können diese nicht mit einer einzigen PLL erzeugt werden und eine Kaskadierung zweier PLLs ist nicht möglich, da die acht Takte eine feste Phasenbeziehung zueinander benötigen. Zur Taktgenerierung wird daher eine PLL verwendet, die die beiden DCMs des CMTs treibt. Die PLL erzeugt aus dem Referenztakt zwei Takte mit einer Frequenz von 125 MHz, wobei ein Takt bereits um 45° verschoben ist. Jeder Takt wird in einen DCM geführt und es werden die vier festen Ausgänge des DCMs verwendet, die den Eingangstakt um 0°, 90°, 180° und 270° verschieben, was die Erzeugung aller acht Takte auf einem CMT ermöglicht. Zur Konfiguration der PLL und der beiden DCMs wird der Core-Generator in ISE 14.7 verwendet, der den synthesefähigen VHDL-Code erzeugt.

5.2 Simulationskonzept

Im Entwicklungsprozess wird jede einzelne Komponente des Gesamtsystems durch eine Verhaltenssimulation getestet und auf ihre korrekte Beschreibung überprüft. Abschließend wird eine gesamte Simulation der Verhaltensbeschreibung des VHDL-Topmoduls *cdr_top* vorgenommen. Die Abbildung 5.9 zeigt das allgemeine Simulationskonzept der Topmodultestbench mit den wichtigsten Signalen. In dieser Testbench werden der Taktgenerationsblock *clockwrapper* und das Topmodul der CDR getestet, gekennzeichnet als Device-Under-Test (DUT). Der Taktgenerationsblock *clockwrapper* ist die aus dem ISE-Core-Generator generierte PLL mit den beiden kaskadierten DCMs. Das Modul *clk_gen_pll* erzeugt den festen 50 MHz Referenztakt, der vom *clockwrapper* genutzt wird, um die acht phasenverschobenen Takte zu erzeugen. Die acht erzeugten Takte werden der CDR zugeführt und zusätzlich die nullte Taktphase dem PRBS-Checker. Ein zweiter Referenztakt wird vom Block *clk_gen_prbs* erzeugt und damit der PRBS-Generator *prbs_gen* gespeist, welcher einen seriellen PRBS-7-Bitstrom *prbs_out* erzeugt. Dieser führt direkt auf den Dateneingang der CDR. Die zurückgewonnenen Daten *data_out* der CDR werden mit dem Steuerungssignal *data_valid* an den PRBS-Checker *prbs_check* übergeben, welcher eine Synchronisation mit den erhaltenden Daten durch das Signal *sync* anzeigt, um anschließend auftretende Fehler durch das Signal $data\_out \neq 0$ zu kennzeichnen. Als PRBS-Generator und -Checker wird der von Xilinx in [16] zur Verfügung gestellte Code mit einigen Modifikationen für das IPMS-IBERT-System verwendet. Der Referenztakt *clk_prbs* kann zusätzlich durch eine Funktion zufällig in seiner Periode verändert werden, was Jitter modelliert, um unterschiedliche Testszenarien zu simulieren. Nach der abgeschlossenen Verhaltenssimulation wird eine Timingsimulation des implementierten System mit dem IPMS-IBERT zur Verifikation der allgemeinen Funktion nach der Platzierung und Verdrahtung im FPGA durchgeführt. Hierbei bietet die vorangegangene statische Timinganalyse bereits einen ersten Indikator, ob durch die Implementation keine Setup-and-Hold-Zeiten verletzt werden.

Die abschließende Verhaltenssimulation wird in vier Abschnitte unterteilt, tabellarisch aufgelistet in Tabelle 5.4. Im nullten Abschnitt erfolgt die Datenübertragung in einem synchronen Betrieb, die Datenrate ist ideal mit 125 Mbps. Es sind keine Fehler zu erwarten und zusätzlich keine Änderung der Taktphase zur Rückgewinnung. Im ersten Abschnitt wird die Datenrate mit 125,79 Mbps geringfügig schneller als der Empfangstakt gewählt. Dies sollte in einer stetigen Änderung der Taktphase resultieren und mehrere Phasendrehungen hervorrufen bei einer weiterhin fehlerfreien Rückgewinnung. Nach

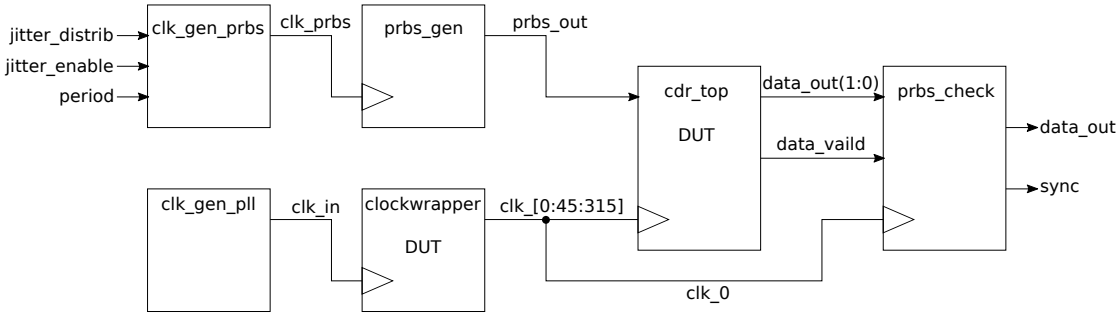


Abbildung 5.9: Aufbau der Topmodultestbench

vier vollständigen Umdrehungen der Taktphasen wird der Simulationsabschnitt gestoppt. Im zweiten Abschnitt beträgt die Datenrate 124,22 Mbps und es ist erneut mit mehreren Phasendrehungen zu rechnen, jedoch in die entgegengesetzte Richtung. Abschließend wird im dritten Abschnitt die eingestellte Datenrate erneut mit 125 Mbps ideal gewählt, jedoch wird durch eine Zufallsfunktion die Sendetaktperiode variiert, was sich als Jitter in der Übertragung äußert. Das System sollte durch Anpassung an den Jitter eine stetige Änderung der Taktphase vornehmen. Fehler können auftreten, sollten sich aber nur auf einzelne Bitslips oder Bitstuffs beschränken.

Tabelle 5.4: Abschnitte der Gesamtsimulation

Abschnitt	Datenrate [Mbps]	Jitter
0	125	nein
1	125, 79	nein
2	124, 22	nein
3	125	ja

Es werden zwei Konfigurationen mit einer unterschiedlichen Taktteilung in der Gesamtsimulation gegenübergestellt, welche durch $DIVIDER\_BITS\_CPP$ erreicht wird. Die Länge der Schieberegister wird auf $NS\_BITS = 8$ festgelegt, da das System mit einem PRBS-7-Signal getestet wird und die längste Folge gleichwertiger Bits acht ist. Dadurch ist gewährleistet, dass stets eine Transition der Daten und das daraus resultierende *up_down*-Signal im Schieberegister gespeichert ist. Zusätzlich wird auf eine Taktteilung im Schleifenfilter verzichtet und dieser mit der nullten Taktphase betrieben, um die allgemeine Sensitivität zu erhöhen. In der ersten Konfiguration wird $DIVIDER\_BITS\_CPP = 2$ gesetzt, was in einem kurzen Aktualisierungszyklus von

vier Taktzyklen resultiert. Es ist zu erwarten, dass dadurch die CDR sehr gut auf Taktwanderungen reagiert, jedoch ebenfalls für hochfrequenten Jitter anfällig wird. In der zweiten Konfiguration wird der Zähler mit einer Breite von $DIVIDER\_BITS\_CPP = 4$ gewählt. Dadurch wählt der CPP nach 16 Taktzyklen eine neue Taktphase. Es ist zu erwarten, dass die CDR eine geringe Toleranz gegenüber Taktwanderungen besitzt, jedoch eine höhere Jittertoleranz im hochfrequenten Bereich. In der Tabelle 5.5 sind die Konfigurationen nochmals zusammenfassend dargestellt.

Tabelle 5.5: Parameterkonfigurationen der getesteten CDRs

Konfiguration	DIVIDER_ BITS_CPP	Taktzyklen von clk_0	NS_BITS	Taktteilung des Schleifenfilters
Nr. 1	2	4	8	nein
Nr. 2	4	16	8	nein

5.3 Messaufbau

5.3.1 Funktionale Tests

Die vollständige Implementation wird vor den eigentlichen Tests mit den optischen Transceivern zunächst auf die allgemeine Funktionalität überprüft. Hierzu werden zwei FPGAs mit dem IPMS-IBERT und der entwickelten CDR programmiert und anschließend über Koaxialkabel direkt verbunden. So werden äußerliche Einflüsse größtenteils eliminiert und die allgemeine Funktionalität der Kommunikationsstrecke und des IBERT-Systems mit der CDR lassen sich verifizieren. Die Abbildung 5.10 zeigt den Aufbau der funktionalen Tests als Blockschaltbild und die Abbildung 5.11 den eigentlichen Messaufbau. Des Weiteren werden mit diesem Messaufbau Tests zur Optimierung der maximal möglichen Datenrate durchgeführt.

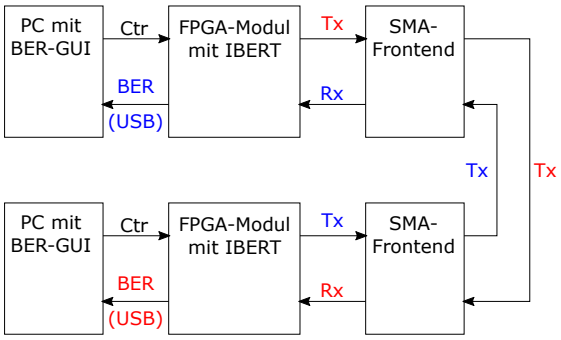


Abbildung 5.10: Blockschaltbild des funktionalen Messaufbaus

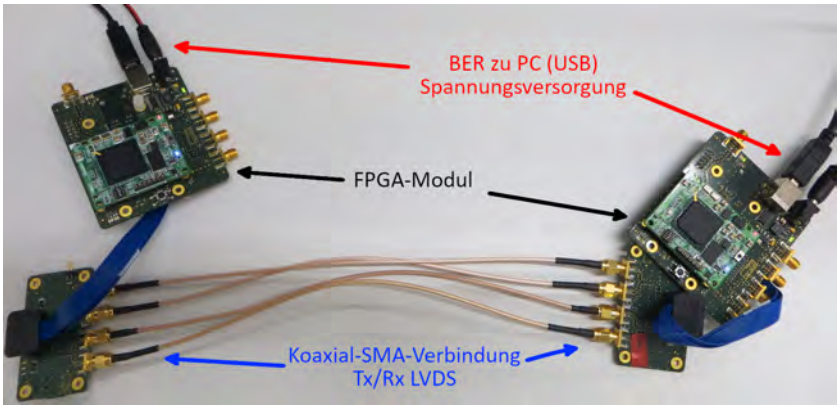


Abbildung 5.11: Messaufbau der funktionalen Tests

5.3.2 Jittertoleranzmessung

Die CDR wird mit den zwei unterschiedlichen Parameterkonfigurationen aus der Simulation zunächst auf ihre Jittertoleranz getestet, um anschließend mit der besseren Konfiguration die Vermessung der optischen Transceiver vorzunehmen. Die verwendeten Geräte sind dem Anhang A.1 zu entnehmen und der Aufbau ist als Blockschaltbild in Abbildung 5.12 dargestellt. Zur Erzeugung des benötigten PRBS-Signales wird der Signalgenerator verwendet werden, welcher die benötigte PRBS-7-Sequenzen direkt erzeugen kann und diese zusätzlich mit einer Frequenzmodulation eines Sinus versehen kann, welche den Jitter modelliert. Der Generator wird so eingestellt, dass beide Ausgänge ein LVDS-Signal erzeugen, welches direkt per Koaxialkabel an das Frontend mit SMA-Anschlüssen an den FPGAs angeschlossen werden kann, um äußerliche Störquellen größtenteils zu vermeiden. Direkt an den SMA-Anschlüssen, so nah wie möglich an den Eingangspins des FPGAs, wird mit dem Oszilloskop das Datenauge der Übertragung ermittelt, um die Übertragung zusätzlich visuell zu bewerten. Ebenso wird das Oszilloskop verwendet, um das Jitterhistogramm aufzunehmen und den eingestellten Jitter am Signalgenerator zu überprüfen. Als Ziel-BER wird 10^{-8} gesetzt und die Jittertoleranz mit den Ergebnissen aus [19] und den Jitterspezifikationen aus [21] verglichen. Der vollständige Messaufbau ist in Abbildung 5.13 gezeigt. In Abbildung 5.13a ist die Anordnung des Oszilloskops, Signalgenerators und FPGA-Moduls gezeigt und in Abbildung 5.13b eine zusätzliche Nahaufnahme des FPGA-Moduls mit SMA-Frontend und angeschlossener Probe des Oszilloskops dargestellt. Zusätzlich kann der Fangbereich der CDR gemessen werden, mit welchen Datenraten eine Übertragung mit einem BER $< 10^{-8}$ gewährleistet werden kann.

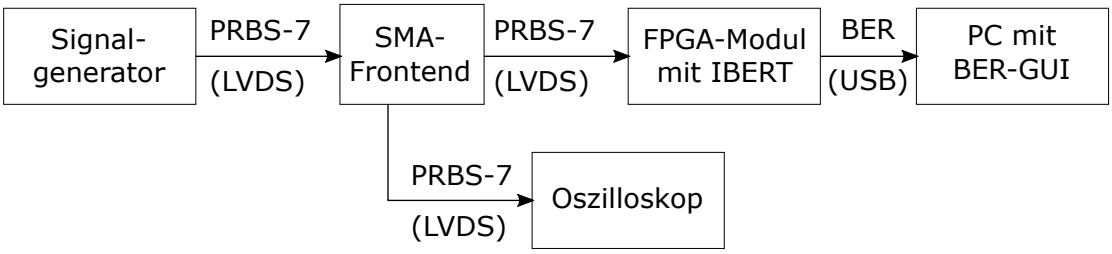
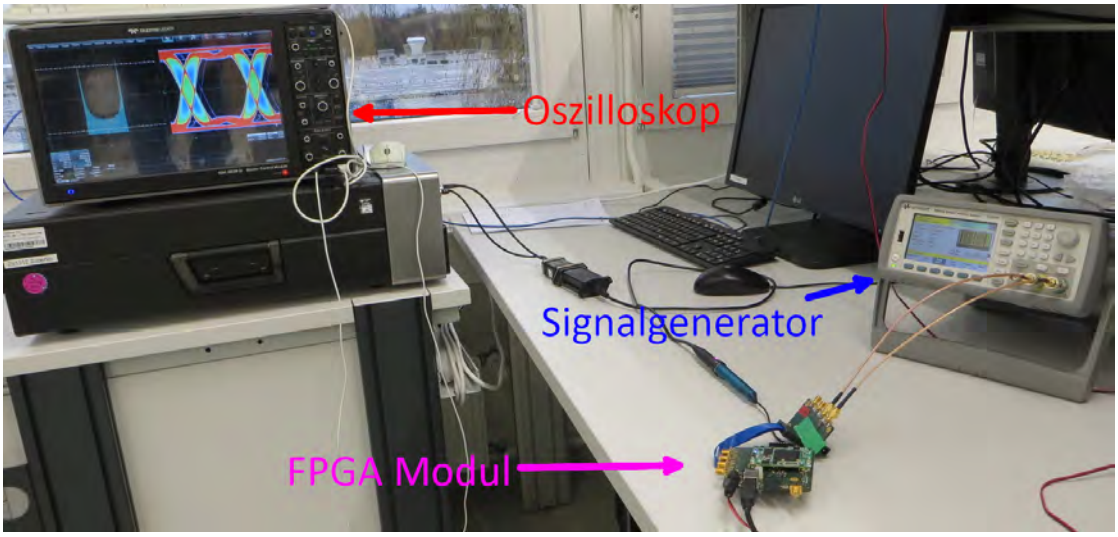
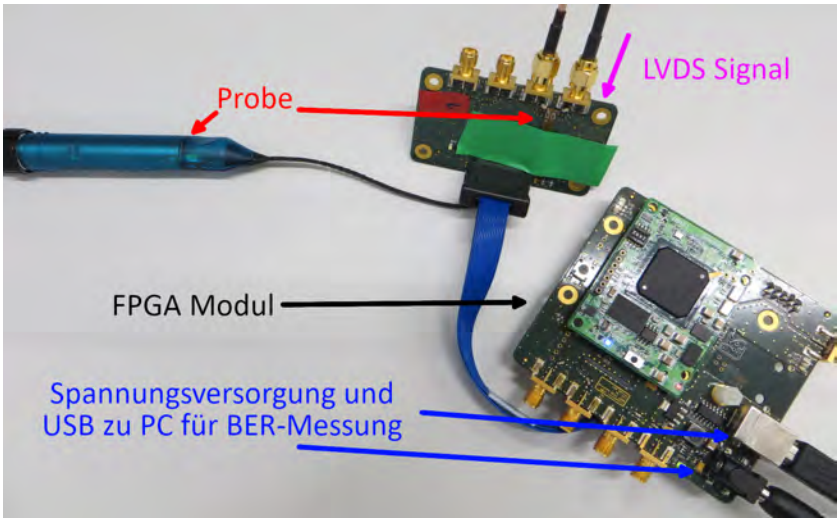


Abbildung 5.12: Blockschaltbild des Messaufbaus zur Messung der Jittertoleranz



(a) Anordnung des Oszilloskops, Signalgenerators und FPGA-Moduls



(b) Nahaufnahme des FPGA-Moduls mit Probe des Oszilloskops

Abbildung 5.13: Aufbau der Jitteroleranzmessung

5.3.3 Charakterisierung der optischen Transceiver

Mit der CDR, die eine bessere Jitteroleranz aufweist, werden zwei Prototypen der optischen Transceiver charakterisiert, welche im vollen duplex Betrieb bei 125 Mbps verwendet werden. Die verwendeten Geräte sind dem Anhang A.2 zu entnehmen und der prinzipielle Messaufbau ist in einem Blockschaltbild in Abbildung 5.14 dargestellt. Zur Aufnahme des Augendiagramms muss das Empfangssignal zunächst über zwei Frontends mit Koaxialkabeln geführt werden, damit die Probe des Oszilloskops platziert werden kann. Um Verfälschungen durch Reflexionen zu minimieren, müssen die Transceiver erhöht positioniert werden und ein möglichst objektfreies Sichtfeld besitzen. Um die optische Empfangsleistung zu variieren, werden die Transceiver entlang einer Achse bewegt und die abfallende Spannung über einen Widerstand hinter der Diode gemessen. Über die abfallende Spannung lässt sich dann die Empfangsleistung rechnerisch ermitteln. Zusätzlich wird der Raum für die Messung abgedunkelt und das Umgebungslicht minimiert, welches durch zeitliche Änderung die Messung der Empfangsleistung verfälschen kann. Ebenfalls wird das Offset der Empfangsleistung bei nicht stattfindender Übertragung gemessen, damit eine eindeutige Aussage über die Empfangsleistung möglich wird. Die Abbildung 5.15 zeigt den vollständigen Messaufbau.

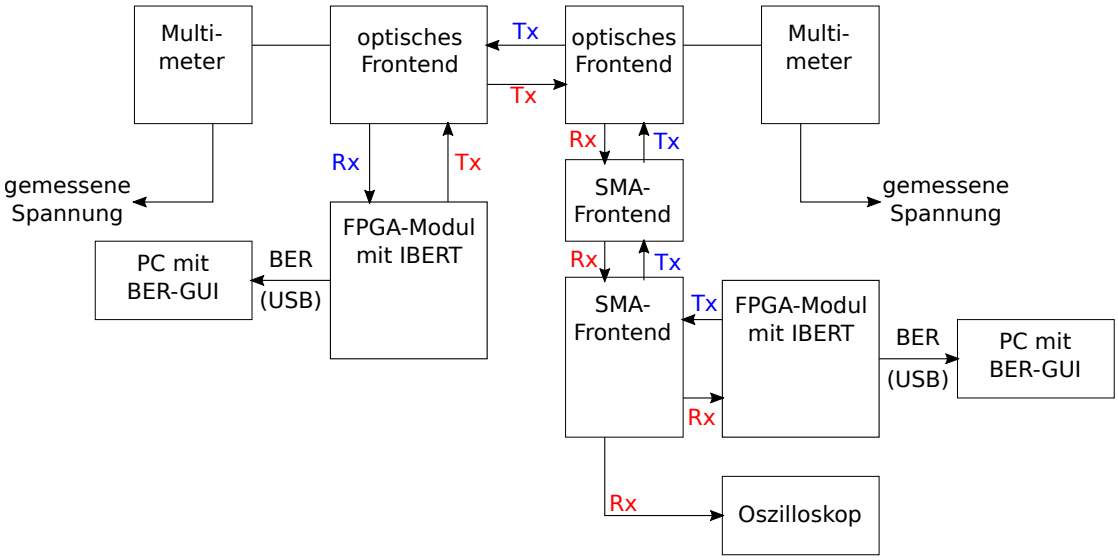
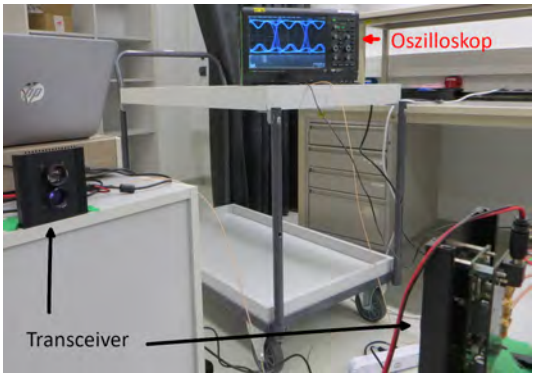


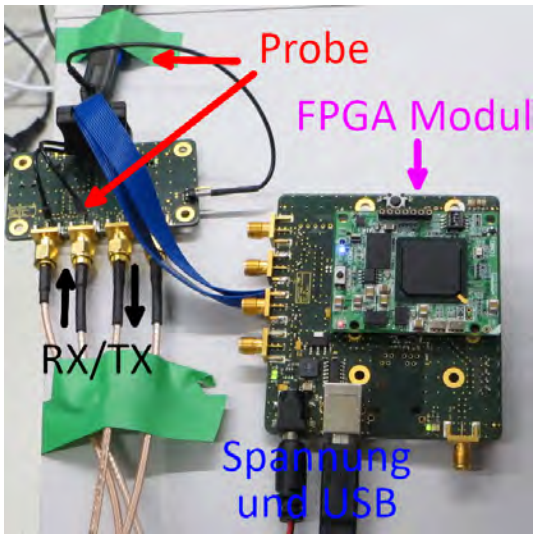
Abbildung 5.14: Blockschaltbild des Messaufbaus zur Charatkerisierung der optischen Transceiver



(a) Gesamtordnung beider Transceiver mit Oszilloskop zur Aufnahme des Augendiagramms

(b) Messaufbau eines Transceiver mit Multimeter, Optik, Spannungsversorgung und PC

(c) Abfallende Spannungsmessung über der Photodiode



(d) Nahaufnahme des FPGA-Moduls mit Probe des Oszilloskops und Spannungsversorgung und USB-Schnittstelle für BER-Messung

Abbildung 5.15: Aufbau zur Charakterisierung der optischen Transceiver

6 Ergebnisse

6.1 Simulation

Für diesen Abschnitt sind alle Signalverläufe der ersten CDR-Konfiguration in blau dargestellt und der zweiten Konfiguration in rot. Die Tabelle 5.5 fasst die getesteten Konfigurationen zusammen und Tabelle 5.4 den Simulationsablauf.

In der Abbildung 6.1 sind die wichtigsten Signalverläufe in den ersten drei Abschnitten der Gesamtsimulation dargestellt. In Abschnitt 0 startet die Übertragung und beide CDRs setzen das *locked*-Signal, da sie eine Übertragung erkennen. Die CDRs liefern die ersten Daten an die PRBS-Checker, welche zunächst Fehler im Signal *err_ out* ausgeben. Nachdem sich die PRBS-Checker auf die Daten synchronisiert haben, wird das *sync*-Signal gesetzt. Anschließend liefern beide Checker keine Fehler, die CDRs gewinnen die ankommenden Daten richtig zurück und übergeben diese korrekt an die Checker. Da der Sendetakt genau mit dem Empfängertakt übereinstimmt, wird die Taktphase zur Rückgewinnung *rec_ clk_ state* in diesem Simulationsabschnitt nicht geändert.

Die Übertragung wird zum Ende des nullten Abschnitt gestoppt, was beide CDRs erkennen, da das *locked*-Signal auf „0“ gesetzt wird. Abschnitt 1 der Simulation wird gestartet. Nun ist der Sendetakt mit 125,79 MHz minimal größer als der Empfangstakt, was in einer stetigen Änderung der zurückgewonnen Taktphase resultiert. Des Weiteren wird durch das Signal *phase_ jump[1]* eine vollständige Drehung der Taktphase angezeigt und es sind ausschließlich *dec*-Signale aus dem Schleifenfilter zu verzeichnen. Beide CDRs liefern keine Fehler, jedoch sind die *dec*-Signale nicht identisch.

In Abschnitt 2 ist der Sendetakt mit 124,22 MHz minimal geringer, was ebenfalls eine stetige Änderung der Taktphase bedeutet. Nun wird das *inc*-Signal regelmäßig getrieben, jedoch ist dieses ebenfalls nicht identisch über beide CDRs. Durch das *phase_ jump[0]*-Signal wird eine vollständige Phasendrehung angezeigt. Nachdem die Checker sich synchronisiert haben, sind keine Fehler zu verzeichnen.

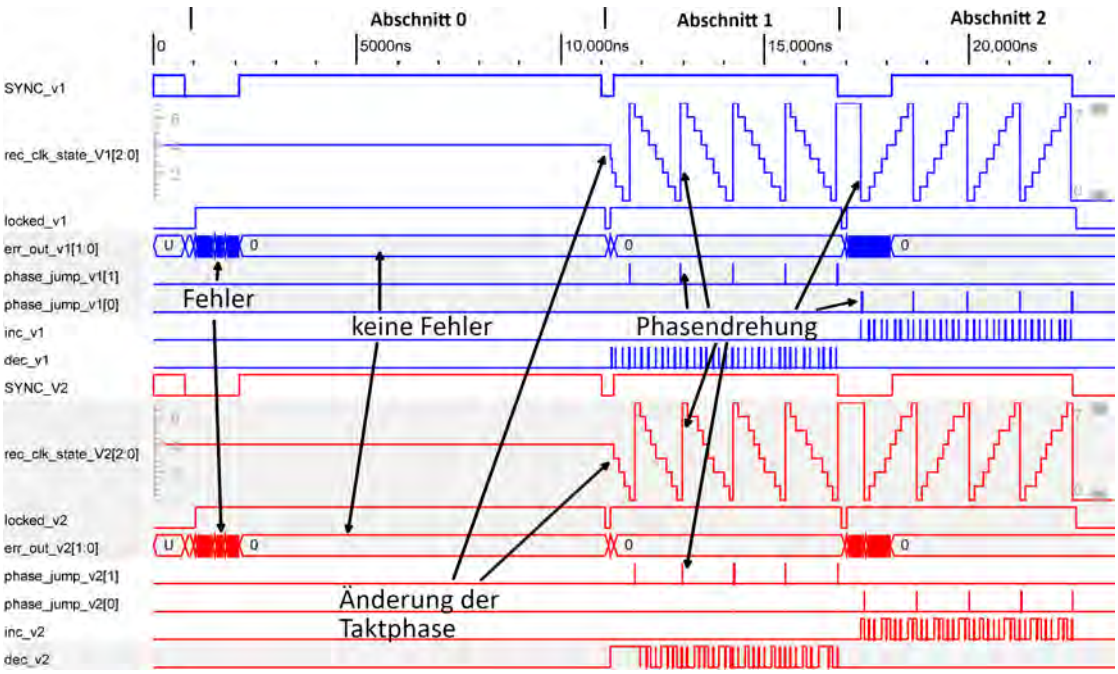


Abbildung 6.1: Allgemeine Verhaltenssimulation mit Taktwanderung

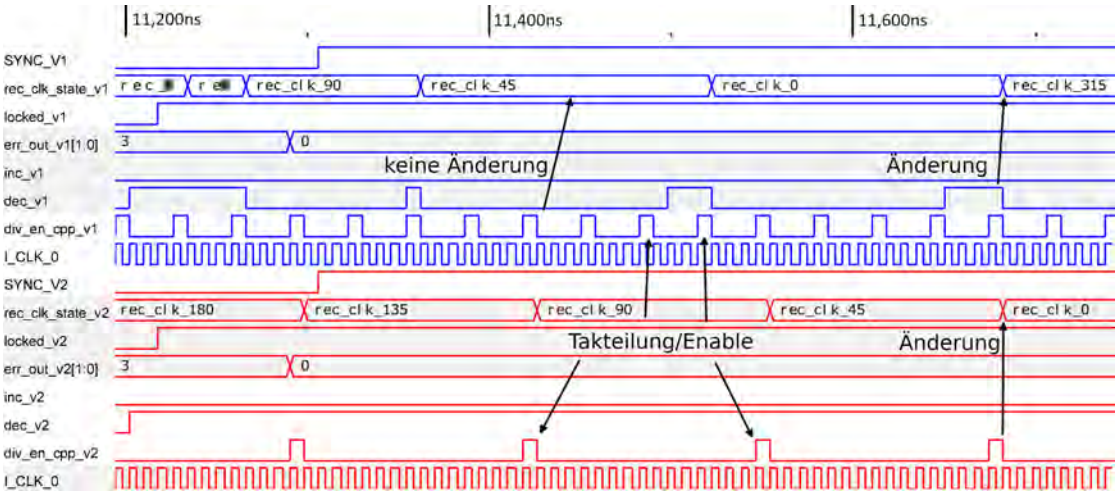


Abbildung 6.2: Vergrößerter Ausschnitt der simulierten Signalverläufe bei Taktwanderung

Die Abbildung 6.2 zeigt eine vergrößerte Aufnahme des Signalverlaufes im zweiten Simulationsabschnitt. Zusätzlich wird das Taktsignal der nullten Taktphase *I_CLK_0* und die enable-Signale *div_en_cpp* beider CDRs dargestellt. Bei der ersten CDR wird wie gewünscht alle vier Taktzyklen das Enable-Signal gesetzt und in der zweiten Konfiguration nur alle 16 Taktzyklen. Nachdem das Enable-Signal aktiv war, wird in der nächsten Taktperiode bei einem aktiven *dec*-Signal eine neue Taktphase gewählt, was in der zweiten CDR im gezeigten Ausschnitt stets der Fall ist. Die erste CDR ändert nicht bei jeder Gelegenheit die Taktphase. Während bei der ersten CDR das *dec*-Signal zunächst inaktiv gesetzt wird, ist es bei der zweiten CDR über den ganzen Zeitraum aktiv. Die zweite CDR reagiert in diesem Ausschnitt spät auf die Taktwanderung, da für die nachfolgenden Taktphasen bereits ein *dec*-Signal im Schleifenfilter vorliegt. So bleibt dieses auch nach der Neuwahl der Taktphase aktiv, womit sich die unterschiedlichen *dec*- und *inc*-Signale aus der Abbildung 6.1 erklären lassen.

In Abbildung 6.3 ist die letzte Phase der Gesamtsimulation gezeigt. Die Übertragung wird erneut mit genau 125 Mbps gestartet, jedoch wird zusätzlich zufälliger Jitter simuliert, der den Sendetakt zufällig verkürzt oder verlängert im gewählten Bereich von ±0,5 ns. Zwar ändern sich die zurückgewonnen Taktphasen beider CDRs sehr häufig, jedoch liefert die erste CDR keine Fehler. Dagegen gewinnt die zweite CDR über den gesamten Zeitraum den Großteil der Daten fehlerhaft zurück, was an *err_out_v2* eindeutig zu erkennen ist. Jedoch ist es auffällig, dass die gewählten Taktphasen sich häufig in eine Richtung ändern, was auf eine zusätzliche Taktwanderung schließen lässt.

Die Abbildung 6.4 zeigt die Ursache für den Fehler der zweiten CDR. Der Sendetakt *clk_prbs* und die gewählte Taktphase zur Rückgewinnung *clk_45* ist dargestellt. Der Cursor TimeA zeigt, dass sich die steigende Taktflanke der gewählten Taktphase nah an der steigenden Taktflanke des Sendetaktes befindet, dementsprechend nah an der Transition im Datenauge bzw. die gewählte Taktphase ist spät. Korrekterweise wird das *dec*-Signal gesetzt, was eine Verkürzung der Taktphase fordert, um von der Kante wegzurücken. Bevor die Taktphase geändert werden kann durch das Enable-Signal *div_en_cpp*, wandert der schnelle Sendetakt vor die positive Flanke des Empfangstaktes, gekennzeichnet durch den CursorB. Dies bedeutet den Verlust eines Bits durch einen Bitslip. Anschließend wird durch die Filtercharakteristik das *dec*-Signal nach einigen Takten entfernt, da sich die gewählte Taktphase nun durch die Transition im Datenauge hindurch bewegt hat und früh ist. Dies wird durch die CDR erkannt und das *inc*-Signal wird nach einigen Taktzyklen gesetzt. Durch den Verlust des Bits werden die nachfolgenden Bits in falscher Reihenfolge zurückgewonnen, was der Checker durch ständige Fehler anzeigt. Die Abschnitte in

denen der Checker keine Fehler anzeigt, können durch eine nicht erfasste Taktwanderung des Sendetaktes in die entgegengesetzte Richtung erklärt werden, was den Bitslip aufhebt durch einen Bitstuff.

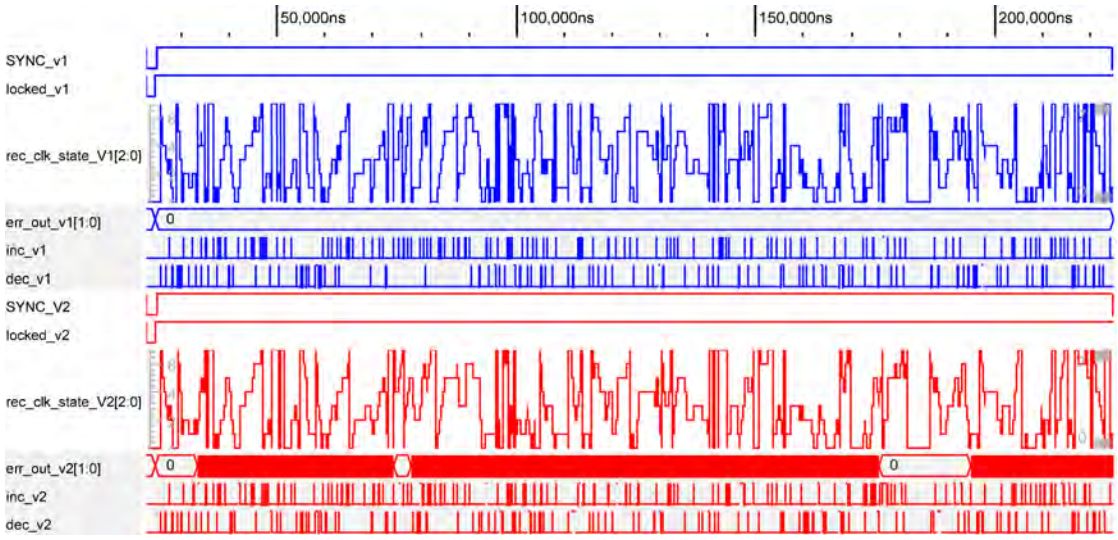


Abbildung 6.3: Simulierte Signalverläufe mit Jitter

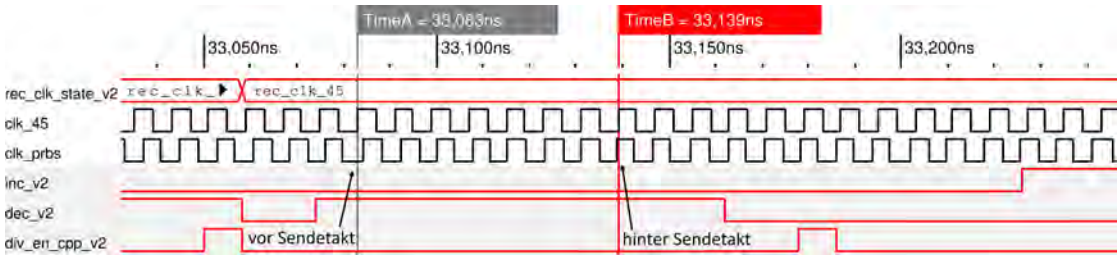


Abbildung 6.4: Erzeugung eines Bitslips durch Taktwanderung

6.2 Funktionale Hardwaretests

Beide CDRs wurden in das IPMS-IBERT-System implementiert und im voll-duplexen Betrieb mit Koaxialkabel getestet. Es wurden 10^{12}-Bits ohne einen gemessenen Bitfehler übertragen, womit eine Übertragung mit einem BER von $< 10^{-8}$ möglich ist. Um eine Aussage für ein geringeres BER zu treffen sind mehr Bits zu übertragen, was die Messdauer deutlich verlängern würde. Des Weiteren funktioniert das System ebenfalls mit einer Datenrate von 200 Mbps mit einem BER von $< 10^{-8}$. Jedoch hat die statische Timinganalyse des Synthesetools ergeben, dass in die kombinatorische Logik zur Errechnung des *inc*- und *dec*-Signals im Schleifenfilter ein Flipflop zum Pipelining gesetzt werden muss.

Eine vollständige Synthese der reinen CDR in ihrer zweiten Konfiguration ohne Taktmanager und ohne das IPMS-IBERT-System und der Festlegung der Taktfrequenzen, ergab eine maximal mögliche Taktfrequenz von 215 MHz ohne zusätzliches Pipelining. Mit einem Pipelining von zwei eingesetzten Flipflops in die kombinatorische Logik des Schleifenfilters, ist laut der statischen Timinganalyse des Synthesetools eine Taktrate von 280 MHz zu erreichen. Bei höheren Taktraten wird die Setup-and-Hold-Zeit bei der Synchronisation der Samples im Phasenkomparator verletzt. Die Implementation auf dem Spartan-6 benötigt 130 *Slice*-Register und zusätzlich 68 *Slice*-LUTs. Werden die beiden Pipelinestufen im Schleifenfilter entfernt wird die CDR mit 106 *Slice*-Register und 76 *Slice*-LUTs implementiert. Insgesamt werden weniger als 1 % der zur Verfügung stehenden Ressourcen auf dem Spartan-6 für die Implementation der CDR benötigt.

6.3 Labormessung

6.3.1 Jittertoleranzmessung

Die Abbildung 6.5 zeigt die aufgenommenen Kurven der Jitteroleranzmessung beider Konfigurationen der CDR und zusätzlich ist die Mindestanforderung der Jittertoleranz nach SIL-5 [21] als gelbe Kurve eingezeichnet. Die aufgetragene Jitteramplitude ist die eingestellte Amplitude am Signalgenerator, da Messungen von Jitteramplituden $>1\,\mathrm{UI_{pp}}$ nicht vom Oszilloskop erfasst werden können und es sich um Taktwanderungen handelt. Es ist deutlich zu erkennen, dass im niederfrequenten Frequenzbereich beide Systeme über den Anforderungen liegen. Ab einer Jitterfrequenz von 300 kHz fällt die Kurve der ersten CDR unter die Anforderungen des SIL-5 mit einem Amplitudenjitter von $0{,}65\,\mathrm{UI_{pp}}$. In höheren Frequenzbereichen fällt die Kurve weiterhin stark ab, bis eine korrekte Übertragung nicht mehr möglich ist. Die gemessene Jittertoleranz der zweiten Konfiguration liegt in den niederfrequenten Frequenzbereichen unter der Jittertoleranz der ersten Konfiguration, jedoch erfüllt sie weiterhin die SIL-5-Spezifikation. Bei einer Jitterfrequenz von 400 kHz fällt die Kurve unter die Spezifikation und nähert sich einem maximal tolerierbaren Jitter von $0{,}4\,\mathrm{UI_{pp}}$ an. Für Jitterfrequenzen $> 10\,\mathrm{MHz}$ konnte die Jittertoleranz der zweiten CDR durch die Begrenzung der Modulation des Signalgenerators nicht mehr ermittelt werden. Die Jittertoleranz liegt jedoch weiterhin deutlich über der Toleranz der ersten CDR-Konfiguration.

In Abbildung 6.6 ist das gemessene Augendiagramm und Jitterhistogramm des unmodulierten PRBS-Signales dargestellt. Der Jitter formt sich im Histogramm zu einer Normalverteilung, was auf reinen zufälligen Jitter hindeutet. Ebenso ist die horizontale Augenöffnung mit 7,6 ns wie erwartet sehr groß bei einem nominalen UI von 8 ns. Die Abbildung 6.7 zeigt das aufgenommene Augendiagramm und Histogramm für beide CDR-Modelle im selben Arbeitspunkt bei einer Jitterfrequenz von 500 kHz. Beide Messungen wurden an der Grenze der Jittertoleranz mit einem BER von 10^{-8} aufgenommen. Der gemessene totale Jitter und die gemessene Augenweite ergeben zusammen eine Zeitdauer von $\approx 7\,\mathrm{ns}$, wobei die nominale Periode 8 ns beträgt. Dies kann auf einen Konfigurations- oder Messfehler zurückgeführt werden. So können einzelne Samples innerhalb des Auges erfasst worden sein, die die Augenöffnung zusätzlich verkleinern, sich aber im Jitterhistogramm nicht niederschlagen. Diesen Effekt zeigt die Abbildung 6.7b, in denen einzelne, erfasste Transitionen von den Übrigen abweichen und näher in der Mitte des Auges liegen.

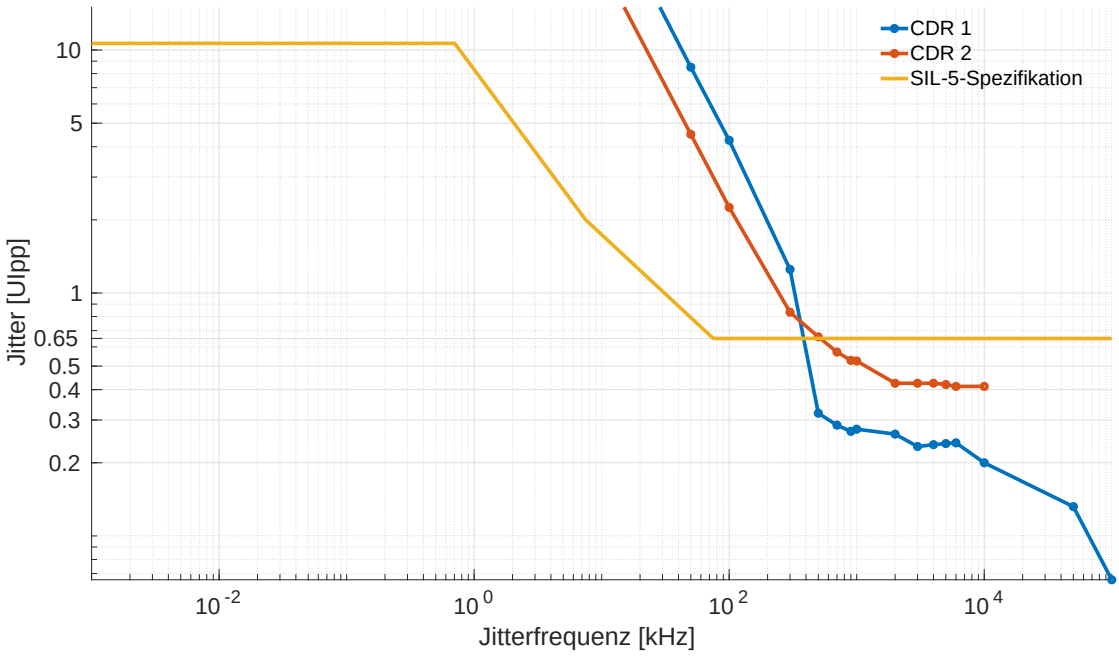


Abbildung 6.5: Jittertoleranzmessung zweier CDR Konfigurationen

Die gemessenen Jitteramplituden TJ_{Mess} werden nachfolgend aus der Differenz der nominalen Periode T_{nom} und der gemessenen Augenweite T_{auge}, dividiert durch die nominale Periode ermittelt, siehe Gleichung 6.1.

$$TJ_{\text{Mess}} = \frac{T_{\text{nom}} - T_{\text{auge}}}{T_{\text{nom}}} \tag{6.1}$$

An beiden Jitterhistogrammen in Abbildung 6.7 ist deutlich zu erkennen, dass der Jitter an beiden Maxima am häufigsten auftritt, was auf die sinusförmige Modulation zurückzuführen ist. Beim ersten CDR-Modell in Abbildung 6.7a ist die Toleranzgrenze bei einer eingestellten Jitteramplitude von 0,32 UI_{pp} erreicht. Es wird eine horizontale Augenöffnung von 4,91 ns gemessen, was eine gemessene Jitteramplitude von 0,39 UI_{pp} bedeutet, unter der Betrachtung der nominalen Taktperiode von 8 ns. Es ergibt sich eine Diskrepanz zwischen eingestellter und gemessener Jitteramplitude von 0,07 UI_{pp}, was auf Filtercharakteristiken des Oszilloskops, Modulationsabweichungen des Signalgenerators oder den beschriebenen Messfehler zurückzuführen ist. Trotz der Diskrepanz lassen sich qualitative Aussagen treffen und die Performance beider CDRs vergleichen. Die zweite Konfiguration zeigt eine deutlich höhere Jittertoleranz bei einer Jitterfrequenz von 500 kHz, dargestellt

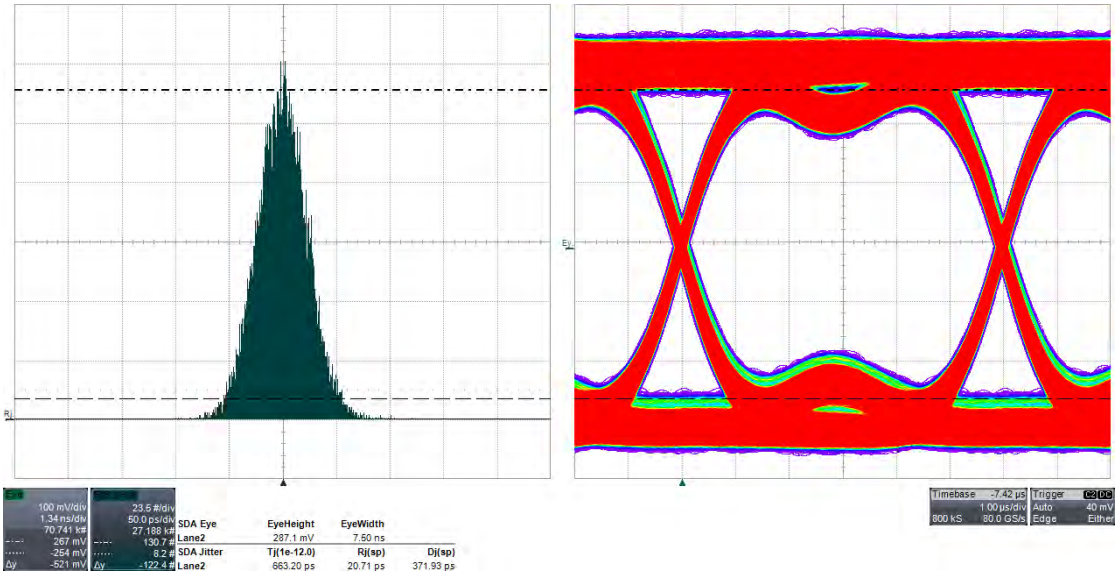
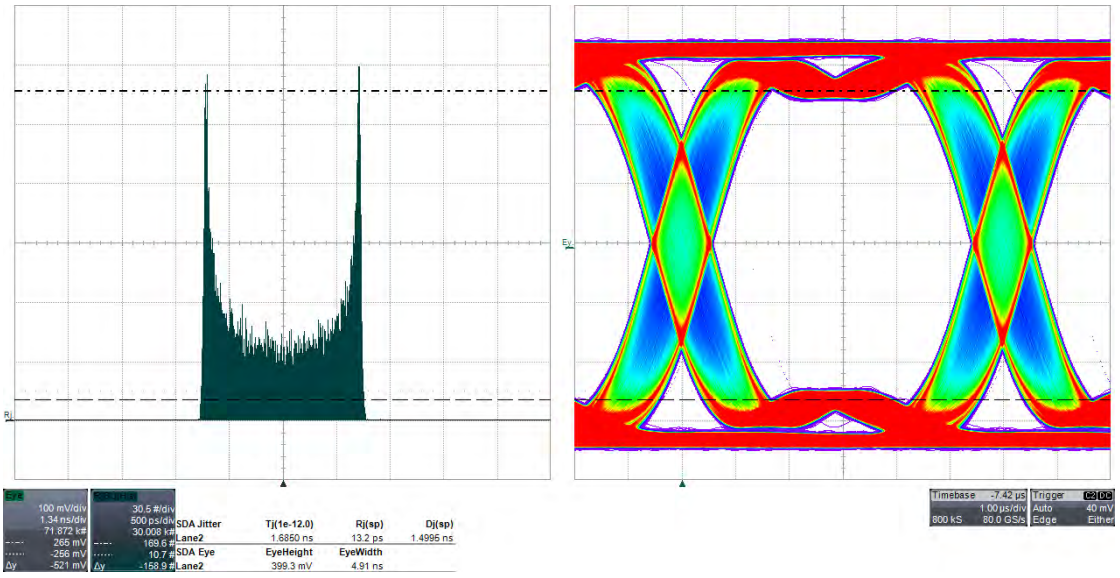


Abbildung 6.6: Augendiagramm und Jitterhistogramm bei unmodulierten PRBS-Signal mit totalem Jitter = 663,20 ps, Augenweite = 7,50 ns, Histogramm = 50 ps/div, Augendiagramm = 1,34 ns/div

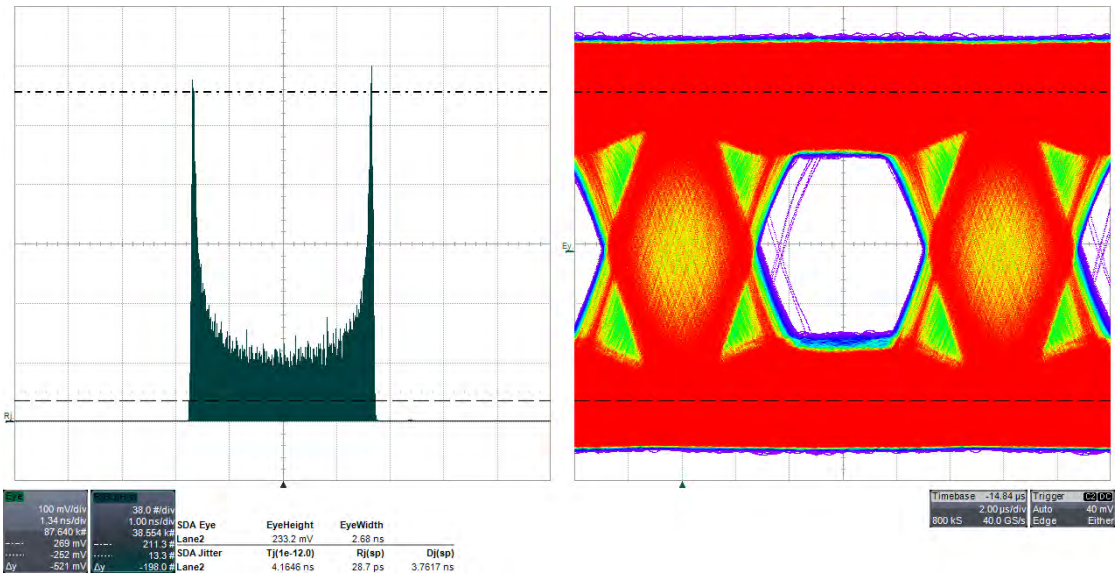
in Abbildung 6.7b. Die eingestellte Jitteramplitude beträgt 0,66 UI_{pp} und die gemessene Jitteramplitude bei einer Augenöffnung von 2,68 ns beträgt ebenfalls 0,66 UI_{pp}. Des Weiteren ist zu sehen, dass der eingestellte Jitter ebenfalls zu einem Schließen der vertikalen Augenöffnung führt. Die Toleranzfähigkeit der zweiten CDR ist in diesem Arbeitspunkt mehr als doppelt so hoch.

Ein ähnlicher Zusammenhang zeigt sich auch an den Augendiagrammen und Jitterhistogrammen in Abbildung 6.8. Die Jitterfrequenz beträgt 5 MHz und die Histogramme zeigen erneut die Charakteristik des sinusförmigen Jitters. Die erste CDR toleriert eine eingestellte Jitteramplitude von 0,24 UI_{pp} und einer gemessenen Jitteramplitude von 0,325 UI_{pp}, gezeigt in Abbildung 6.8a. In Abbildung 6.8b ist das Augendiagramm und Jitterhistogramm bei einer eingestellten Jitteramplitude von 0,42 UI_{pp} und einer gemessenen Jitteramplitude von 0,57 UI_{pp} dargestellt.

Die erste CDR-Konfiguration erhält eine Übertragung bei einer Datenrate zwischen 124,1 Mbps und 125,8 Mbps aufrecht. Die zweite Konfiguration zeigt hier einen geringen Funktionsbereich und die minimale Datenrate beträgt 124,25 Mbps und die maximal Datenrate 125,56 Mbps.

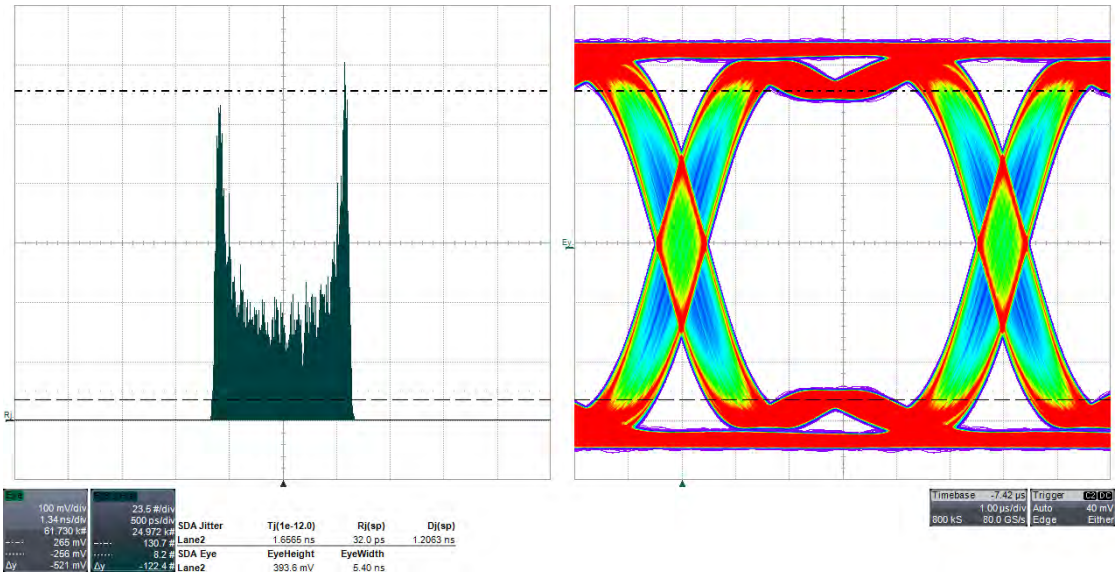


(a) CDR-Konfiguration Nr. 1 mit totalem Jitter = 0,39 UI_{pp}, Augenweite = 4,91 ns, Histogramm = 500 ps/div, Augendiagramm = 1,34 ns/div

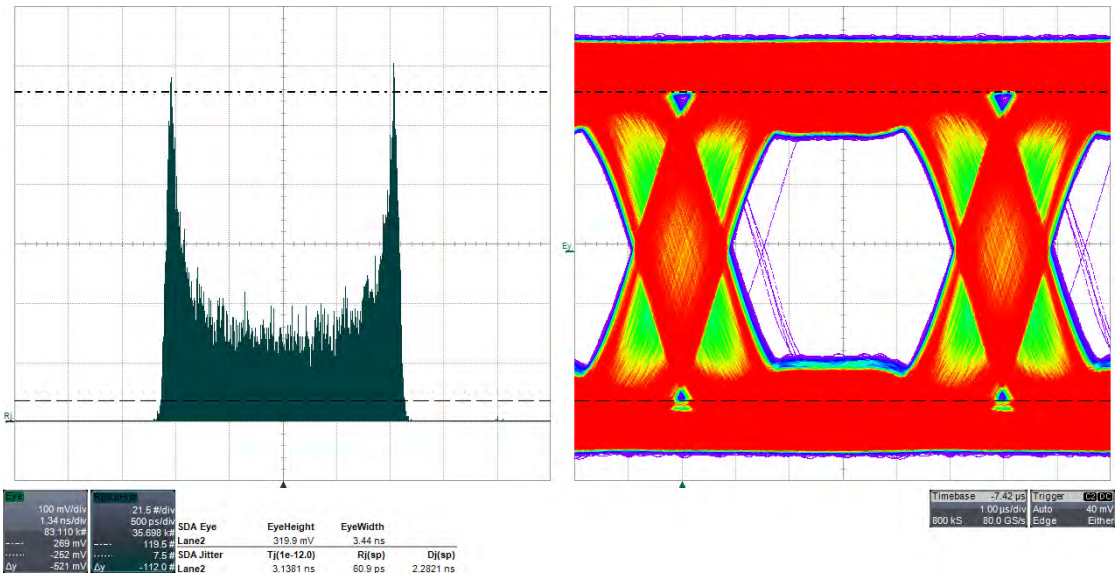


(b) CDR-Konfiguration Nr. 2 mit totalem Jitter = 0,66 UI_{pp}, Augenweite = 2,68 ns, Histogramm = 1 ns/div, Augendiagramm = 1,34 ns/div

Abbildung 6.7: Augendiagramm und Jitterhistogramm für ein BER $< 10^{-8}$ bei einer Jitterfrequenz von 500 kHz mit beiden CDR Konfigurationen



(a) CDR-Konfiguration Nr. 1 mit totalem Jitter = 0,33 UI_{pp}, Augenweite = 5,40 ns, Histogramm = 500 ps/div, Augendiagramm = 1,34 ns/div



(b) CDR-Konfiguration Nr. 2 mit totalem Jitter = 0,57 UI_{pp}, Augenweite = 3,44 ns, Histogramm = 500 ps/div, Augendiagramm = 1,34 ns/div

Abbildung 6.8: Augendiagramm und Jitterhistogramm für ein BER $< 10^{-8}$ bei einer Jitterfrequenz von 5 MHz mit beiden CDR Konfigurationen

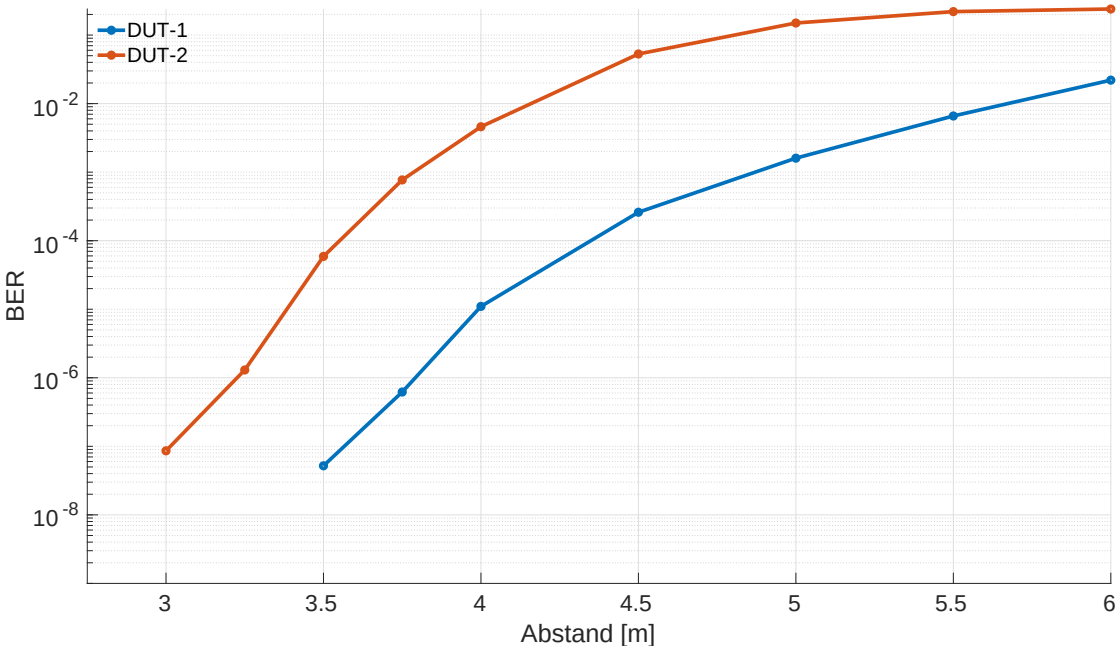


Abbildung 6.9: Gemessenes BER in Abhängigkeit des Abstandes der optischen Transceiver zueinander

6.3.2 Charakterisierung der optischen Transceiver

Die Abbildung 6.9 zeigt das gemessene BER über den Abstand beider optischer Transceiver DUT-1 und DUT-2 zueinander. So ist mit DUT-1 eine Übertragung mit einem BER $< 10^{-8}$ bis 3,25 m möglich. Danach steigt das BER und erreicht bei 4 m einen Wert von 10^{-5} und nimmt mit zunehmenden Abstand der Transceiver weiter zu. Bis zu einer Entfernung von 2,5 m ist mit DUT-2 eine Übertragung mit einem BER $< 10^{-8}$ möglich. Mit höherem Abstand zur Sendequelle steigt das BER und erreicht bei einem Abstand von 4 m bereits ein BER von $5 \cdot 10^{-3}$, welches um einen Faktor von $5 \cdot 10^{2}$ höher ist als das BER des DUTs-1 im selben Arbeitspunkt.

Die optische Empfangsleistung beider Transceiver über den Abstand zueinander ist in Abbildung 6.10 dargestellt. So ist die Empfangsleistung beider Testobjekte in jedem Messpunkt nahezu identisch und unterliegt nur geringen Schwankungen, was auf eine gleiche Arbeitsweise der LEDs, Photodiode und Optik zum Empfangs- und Sendebetrieb beider Transceiver hindeutet.

Die optische Empfangsleistung ist über das BER in Abbildung 6.11 aufgetragen und es bildet sich ein ähnlicher Zusammenhang beider DUTs wie in Abbildung 6.9. Die Kurve

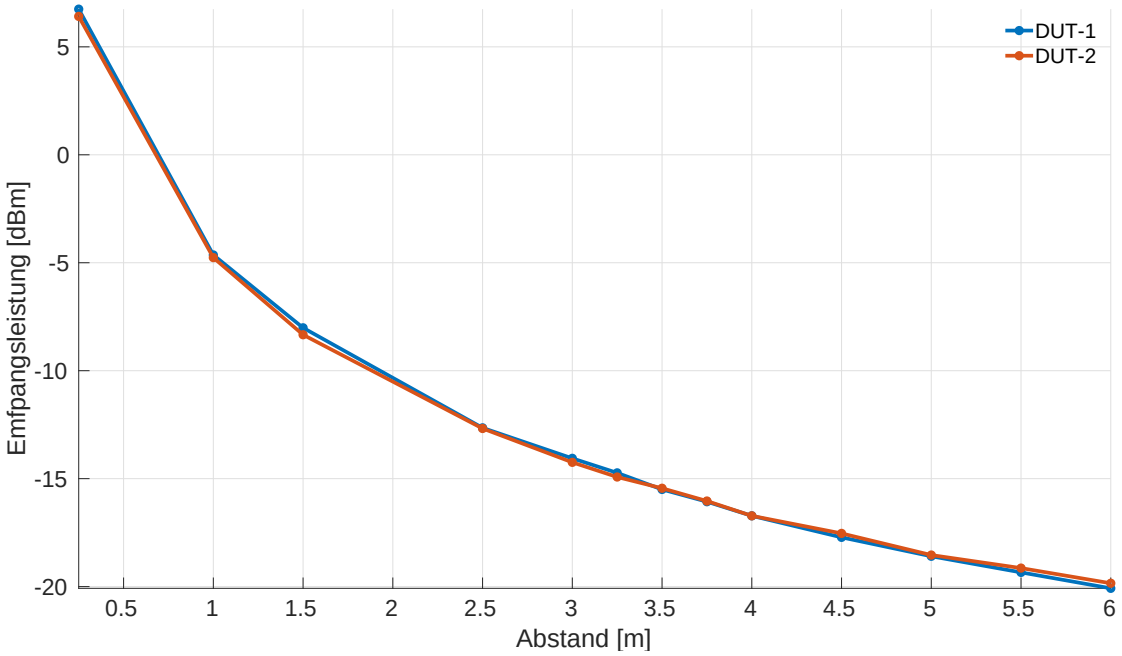


Abbildung 6.10: Optische Empfangsleistung in Abhängigkeit des Abstandes beider optischer Transceiver

des DUT-1 liegt unter der Kurve des DUT-2 und erreicht damit bei gleicher optischer Empfangsleistung ein geringeres BER. Zusätzlich zu den Messpunkten aus der Abstandsmessung, wurden die Transceiver noch in ihrer Ausrichtung zueinander variiert und wenige zusätzliche Datenpunkte aufgenommen, welche gelb und lila gekennzeichnet sind. So liegen die zusätzlichen Messpunkte jeweils in der Nähe der zugehörigen Kurve aus der Abstandsmessung. Abweichungen können durch zusätzliche optische Interferenzen, wie schwankendes Umgebungslicht und Reflexionen erklärt werden, die durch die unterschiedlichen topologischen Änderungen verstärkt oder abgeschwächt worden sind. Abschließend lässt sich aussagen, dass das DUT-1 eine bessere Performance als das DUT-2 aufweist.

Die Abbildung 6.12 zeigt unterschiedliche Augendiagramme bei unterschiedlicher optischer Empfangsleistung und entsprechend unterschiedlichem BER. Die Augendiagramme wurden hinter dem Limitierungsverstärker des DUT-1 aufgenommen, bei 2,5 GS/s über eine Dauer von 60 s. Das Über- und Unterschwingen in den Augendiagrammen kann auf die eingebauten Filtercharakteristiken der analogen Verstärkerschaltung des DUT zurückgeführt werden. Die Abbildung 6.12a zeigt das Augendiagramm bei einer Empfangsleistung von $-14{,}5$ dBm. Das Auge ist weit geöffnet und der totale Jitter beträgt ≈ 2 ns. In Abbildung 6.12b ist bereits eine Verschlechterung des Auges zu erkennen, so

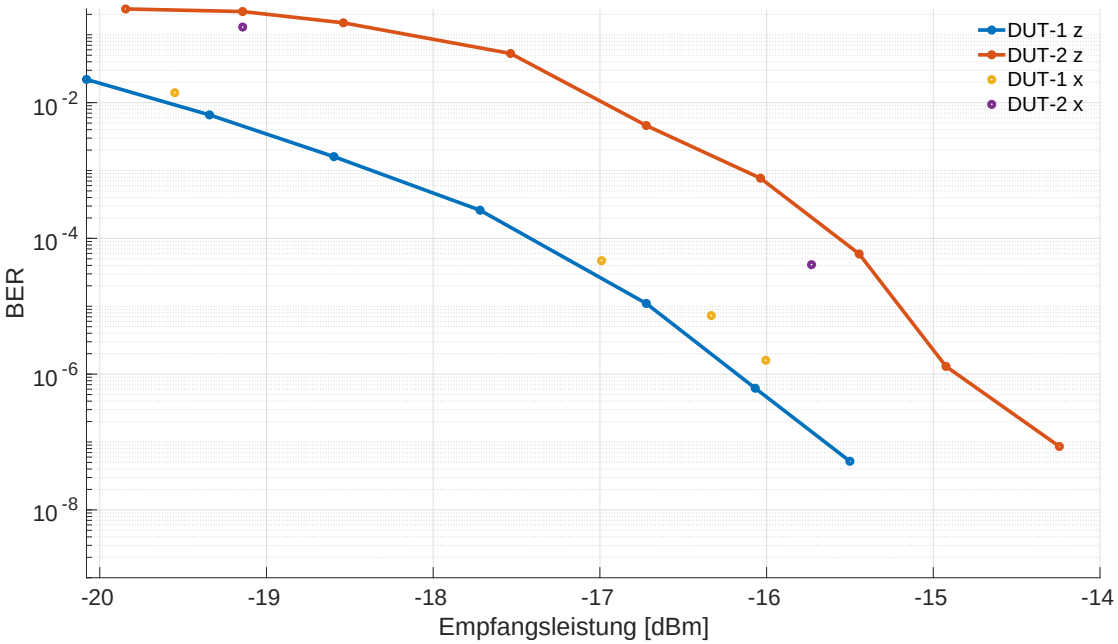
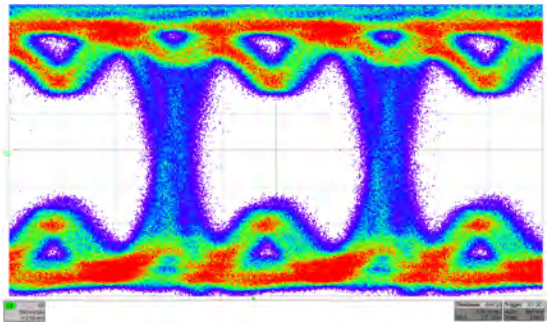
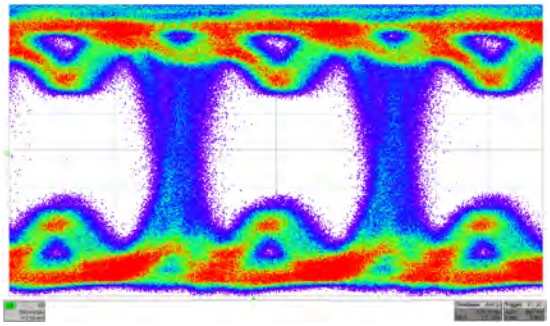


Abbildung 6.11: Gemessenes BER in Abhängigkeit der optischen Empfangsleistung beider optischen Transceiver zueinander

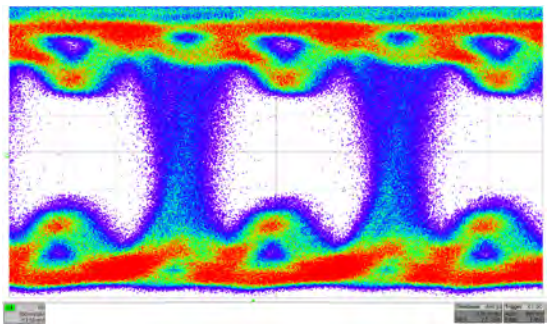
liegen einzelne Samples an den Übergängen verstreuter und bewegen sich zur Mitte des Auges. Bei einer Empfangsleistung von $-16{,}3\,\mathrm{dBm}$ in Abbildung 6.12c liegen nun einzelne Samples in der Mitte des Auges, was sich in einem höheren BER von $7.3 \cdot 10^{-6}$ äußert. Die Übergänge sind zunehmend verrauscht in Abbildung 6.12d bei einer geringen Empfangsleistung von $-17{,}6\,\mathrm{dBm}$ und einem BER von $5.8 \cdot 10^{-4}$.



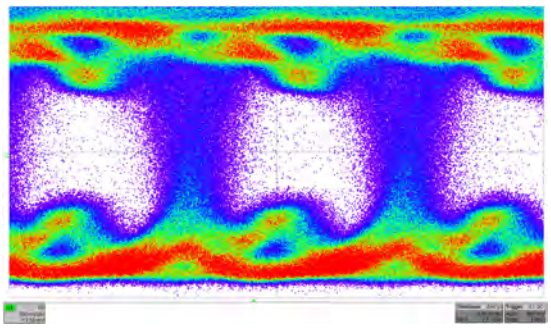
(a) BER $< 10^{-8}$, $-14{,}5\,\mathrm{dBm}$, $2\,\mathrm{ns/div}$



(b) BER $= 10^{-7}$, $-15{,}5\,\mathrm{dBm}$, $2\,\mathrm{ns/div}$



(c) BER $= 7.3 \cdot 10^{-6}$, $-16{,}3\,\mathrm{dBm}$, $2\,\mathrm{ns/div}$



(d) BER $= 5.8 \cdot 10^{-4}$, $-17{,}6\,\mathrm{dBm}$, $2\,\mathrm{ns/div}$

Abbildung 6.12: Augendiagramme der optischen Übertragung bei unterschiedlichem BER und optischer Empfangsleistung

7 Diskussion der Ergebnisse

7.1 Simulation

Die Simulation der gesamten CDR zeigt die grundlegende Funktionstüchtigkeit des entwickelten Systems und deren Möglichkeit auf Jitter und Taktwanderung angemessen zu reagieren. Die erste CDR zeigt erwartungsgemäß eine bessere Reaktion auf Taktwanderung, was auf den kürzeren Aktualisierungszeitraum des CPPs zurückzuführen ist. Weiterhin zeigt sich, dass die Simulation des Jitters optimiert werden muss, da hier eine signifikante Taktwanderung erzeugt wird, die das Simulationsergebnis im dritten Simulationsabschnitt verfälscht. Durch die vorangegangen Simulationsabschnitte ist bereits gezeigt, dass die erste CDR-Konfiguration schneller auf Taktwanderungen reagieren kann und daher auch ein besseres Verhalten im dritten Simulationsabschnitt zeigt. Eine abschließende Abschätzung über die Performance beider CDRs gegenüber Jitter lässt sich mit der gewählten Simulation nicht erreichen.

7.2 Funktionale Tests

Die funktionalen Tests zeigen, dass das entwickelte System für höhere Datenraten verwendbar ist und die allgemeine Funktion mit dem IPMS-IBERT-System gewährleistet ist. Das Einsetzen einer Pipelinestufe in die kombinatorische Logik im Schleifenfilter, um die Länge des kombinatorischen Pfades zu verkürzen, führt jedoch zur einer Verringerung der Sensitivität der CDR, da die Errechnung des *inc*- und *dec*-Signales um einen Takt verzögert wird. Datenraten >200 Mbps konnten wie erwartet mit dem entwickelten System auf der gewählten Plattform nicht erreicht werden, da die DCMs zur Taktgeneration eine maximale Ausgangsfrequenz von 200 MHz aufweisen. Höhere Datenraten wären durch die vorhandene Logik auf dem Spartan-6 möglich, was die Ergebnisse der statischen Timinganalyse bei der Synthese der reinen CDR zeigen. Um Taktraten über

279 MHz zu erreichen ist die Synchronisation der Samples mit dem Systemtakt durch eine weitere Zwischenstufe zu optimieren. Die interne Logik des Spartan-6 ist für maximale Taktfrequenzen von 375 MHz ausgelegt [5].

7.3 Jittertoleranzmessung

Durch die Jittertoleranzmessung wird deutlich, dass die erste CDR-Konfiguration sehr sensitiv auf hochfrequenten Jitter reagiert, bis hin zum Versagen des Systems und die zweite CDR ein deutlich besseres Verhalten aufwies. Die höhere Sensitivität der ersten Konfiguration hat zur Folge, dass diese im niederfrequenten Jitterbereich besser auf Taktwanderung reagieren kann als die zweite Konfiguration. Jedoch erfüllen beide CDRs im niederfrequenten Jitterbereich die gewählte Spezifikation mit einem deutlichen Übermaß und eine Optimierung der CDR ist notwendig, durch Anpassung der gewählten Parameter. Durch Reduzierung der allgemeinen Sensitivität ist zu erwarten, dass die CDR im niederfrequenten Jitterbereich schlechter arbeitet und sich der Spezifikationkurve von oben annähert. Dagegen ist im hochfrequenten Jitterbereich eine Annäherung an die Spezifikationskurve von unten erwartbar, was schließlich zur allgemeinen Erfüllung der Spezifikation über den gesamten Frequenzbereich führen kann.

Die erste Konfiguration zeigt einen größeren Funktionsbereich in Bezug auf die Datenrate als die zweite Konfiguration, was auf die erhöhte Sensitivität der ersten CDR zurückzuführen ist. Zu beachten ist, dass die nominale Mittenfrequenz 125 Mbps beträgt und sich der Funktionsbereich der CDR gleichmäßig für schnellere und langsamere Datenraten ausprägen sollte. Dies ist hier nicht der Fall, was auf einen leichten Taktversatz zwischen der 125 MHz des Signalgenerators und der des FPGAs zurückzuführen ist. Des Weiteren ist zu beachten, dass bei einer höheren oder niedrigeren Datenrate die gemessene Jittertoleranz nicht mehr gegeben ist. Wird bei einer anderen Datenrate die Jittertoleranz gemessen, muss die CDR sowohl die Taktwanderung als auch den Jitter ausgleichen, wodurch die Toleranzgrenze der CDR schneller erreicht ist.

7.4 Charakterisierung der optischen Transceiver

Durch die Messungen mit den optischen Transceivern wurde die allgemeine Funktionstüchtigkeit der CDR für optische kabelungebundene Kommunikationssysteme gezeigt. Des Weiteren ist eine Charakterisierung der optischen Transceiver mit dem entwickelten System möglich. So wird durch die Messungen deutlich, dass die beiden getesteten Transceiver unterschiedliches Verhalten aufweisen. Da in beiden Systemen die gleiche CDR verwendet wird, kann das unterschiedliche Verhalten nur durch die eigentlichen optischen Transceiver oder andere Störeinflüsse erklärt werden. Diese sind jedoch nicht vollständig für das unterschiedliche Verhalten verantwortlich, da sich die zusätzlich aufgenommenen Messpunkte, die durch die topologische Variation der Transceiver erzeugt wurden, sich den Messpunkten aus der Variation der Distanz annähern. Der grundlegende Einfluss auf die Performance ist dadurch auf die analoge Schaltung beider Transceiver zurückzuführen. Der zunehmende Jitter bei geringerer optischer Empfangsleistung in den Augendiagrammen ist auf das abnehmende SNR zurückzuführen und damit einer Transformation von Amplitudenrauschen in Phasenrauschen am Entscheider. Die Abbildung 7.1 verdeutlicht diesen Prozess. Durch die Verstärkerschaltung wird das Rauschen auf dem Nutzsignal stets genauso stark verstärkt wie der eigentliche Empfangspegel, was am Entscheider des Limitierungsverstärkers zu einer Verwischung der Pegelgrenze dV führt. Bei abfallender Empfangsleistung und geringerem SNR verstärkt sich dieser Effekt. Dadurch schaltet der Entscheider zunehmend zu anderen Zeitpunkten dt was sich als Phasenrauschen und damit als Jitter im Empfangssignal äußert. Ist der dadurch entstehende Jitter $>1\,\mathrm{UI_{pp}}$ ragt die Transition über die ursprüngliche Mitte des Auges hinaus und das Auge schließt sich. Die CDR erzeugt dann einen Bitfehler, sollte jedoch die gewählte Taktphase weiterhin beibehalten. Da der Jitter sich im Auge an der linken und rechten Transition gleich stark ausprägt, ist nicht mit einer Änderung der Taktphase zu rechnen und einem womöglichen Herausspringen der CDR aus dem Auge. Aufgrund dieser Charakteristik, sind durch Anpassungen der CDR keine signifikanten Verbesserungen in der Performance der optischen Transceiver zu erwarten.

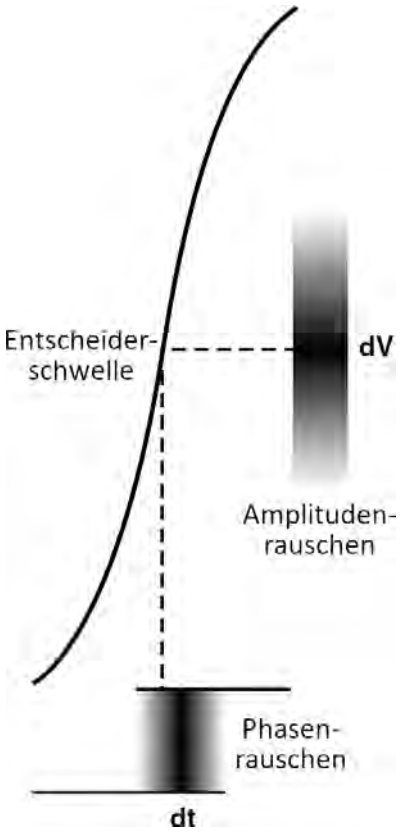


Abbildung 7.1: Übersetzen von Amplitudenrauschen zu Phasenrauschen, modifiziert nach [26, S. 3]

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Es ist in der Arbeit die Konzeptionierung und Realisierung einer voll-digitalen Clock-Data-Recovery (CDR) auf einem FPGA gezeigt worden. Die entwickelte CDR verwendet eine Phaseninterpolator-Architektur und arbeitet mit acht zyklisch angeordneten Taktphasen, die eine Überwachung der Position des Datenauges ermöglichen und anhand dessen, eine Wahl der Taktphase zur Rückgewinnung der eigentlichen Daten getroffen werden kann. Durch die zyklische Anordnung der Taktphasen können langsame Taktwanderungen kontinuierlich verfolgt werden und die CDR eignet sich für den Einsatz einer kontinuierlichen Datenübertragung ohne Unterbrechungen. Dadurch wird der Einsatz der CDR in Echtzeitsystemen möglich und die Latenz zur Rückgewinnung eines Datenbits beträgt maximal fünf Taktperioden. Durch Anpassung der Datenausgabe kann diese Latenz noch weiter verringert werden und sollte, je nach Gesamtsystem, einer zusätzlichen Betrachtung unterzogen werden. Zudem benötigt das entwickelte System wenige Hardwareressourcen und kann durch die allgemeine Beschreibung in VHDL in andere Technologien und Systeme übertragen werden.

Auf dem gewählten FPGA ist das System mit einer maximalen Datenrate von 200 Mbps getestet und verifiziert worden. Zudem konnte durch die Integration der CDR in einen bestehenden Bitfehlerratentester die CDR auf ihre Jittertoleranzfähigkeit getestet werden, unter der Betrachtung des SIL-5 [21]. Es wurden zwei Konfigurationen getestet, welche eine zu hohe Sensitivität auf niederfrequente Taktwanderungen und hochfrequenten Jitter zeigen. Dies äußert sich durch eine übermäßige Erfüllung der Spezifikation in niederfrequenten Jitterbereichen und einer Nichterfüllung im hochfrequenten Jitterbereich. Jedoch ist eine Anpassung der Parameter durch die generische Beschreibung des Systems in VHDL unkompliziert möglich.

Die grundlegende Funktionstüchtigkeit der CDR in Verbindung mit optischen Transceivern wurde gezeigt und des Weiteren wurde das System mit dem integrierten Bitfehlerratentester genutzt, um zwei Prototypen optischer Transceiver zu charakterisieren und ihr Übertragungsverhalten zu messen. Es hat sich gezeigt, dass ein Transceiver eine bessere Performance aufwies als der zweite Transceiver. Da beide Transceiver mit der gleichen CDR arbeiteten und äußerliche Störungen ausgeschlossen werden konnten, ist die Diskrepanz in den Messerergebnissen auf die analoge Schaltung zurückzuführen. Zusätzlich wurde gezeigt, dass die gemessene Bitfehlerrate grundlegend mit der Empfangsleistung zusammenhängt, welche durch die Variation der topologischen Anordnung beider Transceiver beeinflusst werden kann.

8.2 Ausblick

Durch einen Wechsel der Plattform auf einen anderen FPGA, der die Taktgeneration von acht fest phasenverschobenen Takten über 200 MHz ermöglicht, ist eine Datenrate >200 Mbps realisierbar. Des Weiteren können dem Spartan-6 über seine Taktpins die acht Taktphasen zu Verfügung gestellt werden, was eine externe Taktgeneration nötig macht, jedoch ist hier auf einen möglichen Taktversatz durch die Eingangspins zu achten. Ebenso ist es möglich die CDR neu zu konfigurieren, so dass diese mit sieben, anstatt acht Taktphasen arbeitet, was die Verwendung aller sieben Ausgänge einer PLL ohne Kaskadierung von zwei DCMs ermöglicht. Mit einer PLL auf dem Spartan-6 ist es möglich sieben fest-phasenverschobene Taktphasen zu erzeugen [5]. Jedoch ist mit der Reduzierung der Taktphasen ein allgemeiner Abfall der Auflösung des Systems verbunden und es ist nicht mehr gewährleistet, dass dieses weiterhin korrekt arbeitet. Dieser Ansatz bedarf weiterer Untersuchungen und ist daher nicht empfehlenswert. Des Weiteren kann durch die zusätzliche Verwendung der fallenden Taktflanke die Anzahl der benötigten Taktphasen auf vier reduziert werden. So können mit einer PLL die vier benötigten Taktphasen von 0° bis 135° erzeugt werden und im Sampler werden die restlichen vier Taktphasen durch die Verwendung der fallenden Taktflanke modelliert. Bei einem Duty-Cyle von 50 % der vier erzeugten Ausgangstakte lassen sich die acht benötigten Taktphasen erreichen. Jedoch ist hier auf die operationelle Umsetzung im FPGA zu achten. Normalerweise wird sämtliche Logik auf einem FPGA mit der steigenden Taktflanke getaktet und zur Umsetzung der fallenden Taktflanke wird ein zusätzlicher Inverter vor den Takteingang der CLBs des FPGAs geschaltet. Nun darf dieser Inverter nur einen vernachlässigbar kleinen Taktversatz hervorrufen, um eine fehlerfreie Operation weiterhin zu gewährleisten und

das Verhalten des Systems nicht grundlegend ändern. Zusätzlich ist durch diese Neukonfiguration, die CDR nicht mehr einfach auf andere FPGAs und Systeme übertragbar und müsste bei einem Wechsel der Plattform zunächst umfassender verifiziert werden.

Die Jittertoleranz der CDR kann durch Anpassung der generischen Parameter verbessert werden, indem die allgemeine Sensitivität reduziert wird. Durch die Erhöhung des Aktualisierungszyklus des CPPs, mit einer Verkleinerung der Registergröße im Schleifenfilter und der Verwendung einer Taktteilung im Schleifenfilter lässt sich die Sensitivät reduzieren. Eine Taktteilung im Schleifenfilter führt dazu, dass nicht mehr alle erzeugten Kontrollsignale des Phasenkomparators zur Entscheidungsfindung in Betracht bezogen werden und der Einfluss hochfrequenter Störeinflüsse minimiert wird. Diese Maßnahmen würden das System träger modellieren, was schließlich zu einer Erfüllung der gewählten Spezifikation führen kann.

Eine Erhöhung der allgemeinen Performance der optischen Transceiver ließe sich durch eine höhere Sendeleistung erzielen, da bei einem höheren SNR ein niedrigeres BER möglich ist. Des Weiteren kann die optische Empfangsleistung durch den Einsatz von Linsen erhöht werden, welche in einer konzentrierteren und gerichteten IR-Strahlung resultiert. Dadurch ist im Gegenzug jedoch zu erwarten, dass sich der Empfangskegel verkleinert und die optischen Transceiver stets korrekt ausgerichtet sein müssen.

Literaturverzeichnis

[1] C. Maxfield, *The Design Warrior's Guide to FPGAs - Devices, Tools and Flows*, 1st ed. Amsterdam: Elsevier, 2004.

[2] P. Cem Ünsalan and P. Bora Tar, *Digital System Design with FPGA: Implementation Using Verilog and VHDL*, 1st ed. New York: McGraw-Hill Education, 2017.

[3] Xilinx Inc., "XC2000 Logic Cell Array Families," 1985. [Online]: https://www.datasheets360.com/pdf/-5096621001055706963 Zugriff am 15.12.2019.

[4] Xilinx Inc., "Spartan-6 Family Overview," 2011. [Online]: https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf Zugriff am 15.12.2019.

[5] Xilinx Inc., "Spartan-6 FPGA Data Sheet: DC and Switching Characteristics," 2015. [Online]: https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf Zugriff am 15.12.2019.

[6] Xilinx Inc., "7 Series FPGAs Data Sheet: Overview," 2018. [Online]: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf Zugriff am 15.12.2019.

[7] N. Da Dalt and A. Sheikholeslami, *Understanding Jitter and Phase Noise: A Circuits and Systems Perspective*. Cambridge University Press, 2018.

[8] Mohammad Azadeh, *Fiber Optics Engineering*. Springer, 2009.

[9] R. G. Gallager, *Principles of Digital Communication*. Cambridge University Press, 2008.

[10] B. P. Lathi and Z. Ding, *Modern Digital And Analog Communication Systems*, international fourth ed. Oxford University Press, 2010.

[11] M. Hsieh and G. E. Sobelman, "Architectures for multi-gigabit wire-linked clock and data recovery," *IEEE Circuits and Systems Magazine*, vol. 8, no. 4, pp. 45–57, Fourth 2008. [Online]: https://ieeexplore.ieee.org/document/4639004 Zugriff am 04.12.2019.

[12] M. Kubicek and Z. Kolka, "Blind oversampling data recovery with low hardware complexity," *Radioengineering*, vol. 19, 04 2010. [Online]: https://www.radioeng.cz/fulltexts/2010/10_01_074_078.pdf Zugriff am 04.01.2020.

[13] R. Ramirez-Iniguez, S. M. Idrus, and Z. Sun, *Optical Wireless Communications*. Auerbach Publications, 2008.

[14] K. Khandelwal and S. Jain, "A review paper on li-fi technology," 02 2016. [Online]: https://www.researchgate.net/publication/331628546_A_Review_Paper_on_Li-Fi_Technology/citation/download Zugriff am 15.12.2019.

[15] HuMANDATA, *XCM-111 Series User's Manual*, v1.1 ed., 2014. [Online]: https://www.hdl.co.jp/en/spc/XCM/xcm-111/XCM111R1-MAN-EN-V11.pdf Zugriff am 05.01.2020.

[16] Xilinx Inc., "An Attribute-Programmable PRBS Generator and Checker," 2011. [Online]: https://www.xilinx.com/support/documentation/application_notes/xapp884_PRBS_GeneratorChecker.pdf Zugriff am 05.01.2020.

[17] Agilent Technologies, *Understanding Jitter and Wander Measurements and Standards*, 2nd ed. Agilent Technologies UK LTD, 2003. [Online]: http://literature.cdn.keysight.com/litweb/pdf/5988-6254EN.pdf Zugriff am 09.02.2020.

[18] Y. Miki, T. Saito, H. Yamashita, F. Yuki, T. Baba, A. Koyama, and M. Sonehara, "A 50-mw/ch 2.5-gb/s/ch data recovery circuit for the sfi-5 interface with digital eye-tracking," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 4, pp. 613–621, April 2004. [Online]: https://ieeexplore.ieee.org/document/1278579 Zugriff am 04.01.2020.

[19] S. I. Ahmed and T. A. Kwasniewski, "A multiple-rotating-clock-phase architecture for digital data recovery circuits using verilog-a," in *BMAS 2005. Proceedings of the 2005 IEEE International Behavioral Modeling and Simulation Workshop, 2005.*, Sep. 2005, pp. 112–117. [Online]: https://ieeexplore.ieee.org/document/1518197 Zugriff am 04.01.2020.

[20] The Optical Internetworking Forum, "Serdes Framer Interface Level 5 Phase 2 (SFI-5.2)," The Optical Internetworking Forum, Standard, Oct. 2006. [Online]: https://www.oiforum.com/wp-content/uploads/2019/01/OIF-SFI5-02.0.pdf Zugriff am 05.02.2020.

[21] The Optical Internetworking Forum, "System Interface Level 5 (SxI-5): Common Electrical Characteristics," The Optical Internetworking Forum, Standard, Oct. 2002. [Online]: https://www.oiforum.com/wp-content/uploads/2019/01/OIF-SxI5-01.0.pdf Zugriff am 31.01.2020.

[22] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 94–101, Aug 2002. [Online]: https://ieeexplore.ieee.org/document/1024421 Zugriff am 15.12.2019.

[23] Xilinx Inc., "Spartan-6 FPGA Clocking Resources," 2015. [Online]: https://www.xilinx.com/support/documentation/user_guides/ug382.pdf Zugriff am 26.01.2020.

[24] Roland E. Best, *Phase-Locked Loops - Design, Simulation and Applications*, 5th ed. McGraw-Hill Professional, 2003.

[25] P. Novellini, A. Di Fresco, and G. Guasti, "Clock and Data Recovery Unit based on Deserialized Oversampled Data," 2017. [Online]: https://www.xilinx.com/support/documentation/application_notes/xapp1240-k7-us-clk-data-recovery.pdf Zugriff am 19.01.2020.

[26] B. Brannon, "Sampled Systems and the Effects of Clock Phase Noise and Jitter," Analog Devices, Application Note, Oct. 2004. [Online]: https://www.analog.com/media/en/technical-documentation/application-notes/AN-756.pdf Zugriff am 03.02.2010.

[27] Keysight Technologies, "33600A Series Trueform Waveform Generators DATA-SHEET." [Online]: https://www.keysight.com/us/en/assets/7018-04123/data-sheets/5991-3272.pdf Zugriff am 09.02.2020.

[28] Teledyne LeCroy, "LabMaster 10 Zi-A DATA-SHEET." [Online]: https://cdn.teledynelecroy.com/files/pdf/labmaster-10zi-a-datasheet.pdf Zugriff am 09.02.2020.

[29] Teledyne LeCroy, "HDO6000A DATA-SHEET." [Online]: https://teledynelecroy.com/doc/docview.aspx?id=11221 Zugriff am 11.02.2020.

[30] Keysight Technologies, "U1242C DATA-SHEET." [Online]: https://literature.cdn.keysight.com/litweb/pdf/U1241-90107.pdf?id=2281052 Zugriff am 25.02.2020.

A Anhang

A.1 Verwendete Geräte der Jittertoleranzmessung

Gerätebezeichnung	Anzahl	Beschreibung
Keysight TrueForm 33600 [27]	1	Signalgenerator zur Erzeugung des LVDS-Signals mit Frequenzmodulation
LabMaster 10 Zi-A [28]	1	Oszilloskop zur Aufnahme des Augendiagramms und Jittercharakteristik[1]
Entwicklungsboard mit Spartan-6 und SMA-Frontend	1	implementierte CDR mit IPMS-IBERT-System
Windows-PC	1	Auslesen und Darstellung des gemessenen BERs

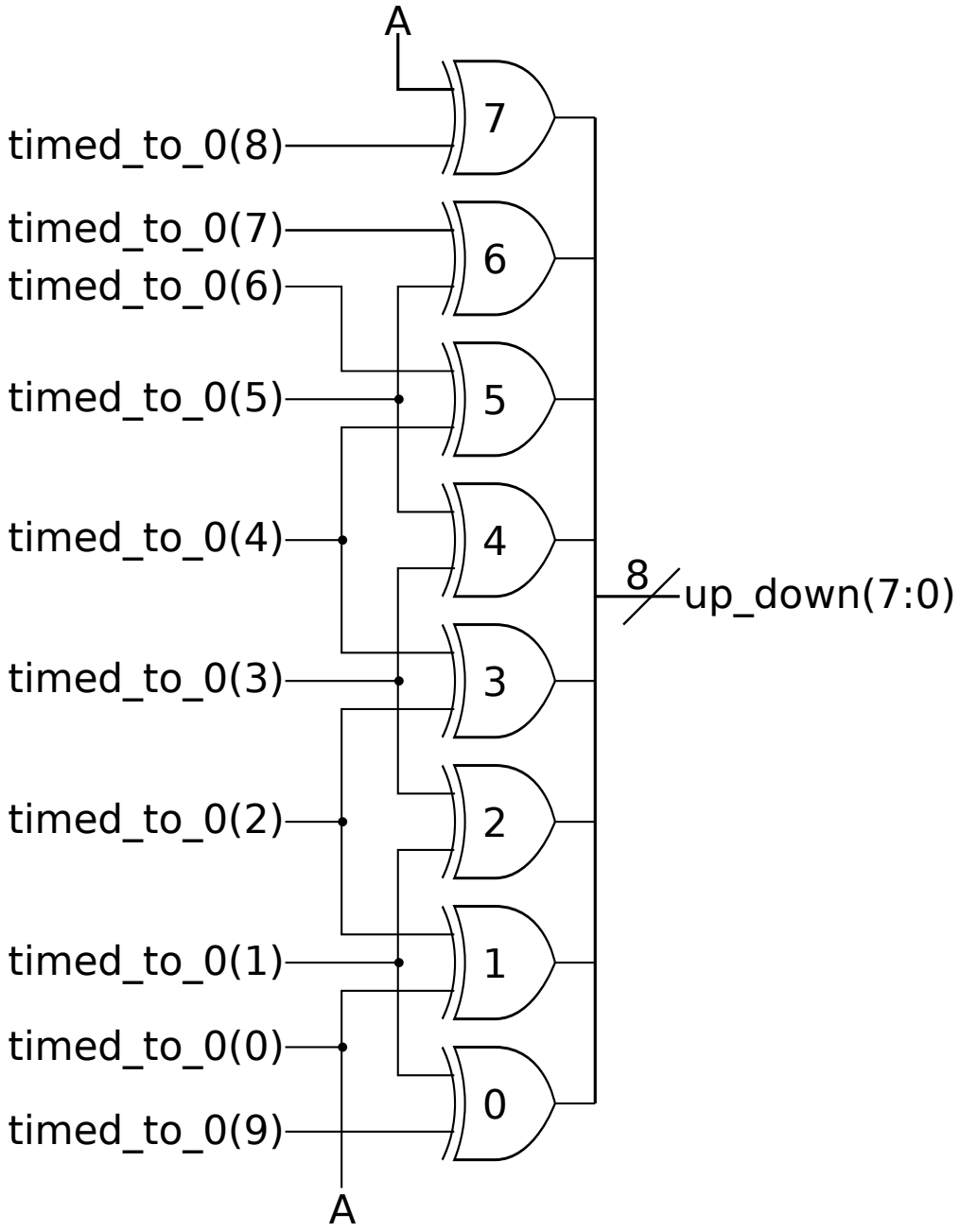
[1] 36 GHz-Oszilloskop mit 13 GHz-Probe und zusätzlicher Bandbreitenbegrenzung auf 1 GHz

A.2 Verwendete Geräte zur Charakterisierung der optischen Transceiver

Gerätebezeichnung	Anzahl	Beschreibung
Optische Frontends	2	zu charakterisierende optische Transceiver
SMA Frontends	2	für vereinfachten Anschluss der Probe des Oszilloskops
LeCroy HD6054 [29]	1	Oszilloskop zur Aufnahme des Augendiagramms bei bestimmten BER[1]
Entwicklungsboard mit Spartan-6	2	implementierte CDR mit IPMS-IBERT-System
Keysight U1242C True RMS Multimeter [30]	2	Messung der abfallenden Spannung über der Photodiode
Windows-PC	2	Auslesen und Darstellung des gemessenen BERs an beiden Transceivern

[1] 500 MHz-Oszilloskop mit 500 MHz-Probe, ohne Bandbreitenbegrenzung

A.3 Implementierter Phasendetektor



Glossar

Augendiagramm Das Augendiagramm lässt eine qualitative Aussage über eine digitale Übertragung zu. Zur Erzeugung werden die elektrischen Signalverläufe einer Übertragung mit einer Zeitdauer eines Unit-Intervals übereinander gelegt. S. 12

BER Das Bitfehlerratenverhältnis (engl. bit-error-ratio) beschreibt das Verhältnis zwischen den insgesamt empfangenen Bits und der Anzahl der fehlerhaften empfangenen Bits. Es dient zur Charakterisierung von digitalen Übertragungssystemen. S. 23

Bitslip Ein Bitslip ist der Verlust eines Bits bei einer Daten- und Taktrückgewinnung. Das Gegenstück ist der Bitstuff, hierbei wird ein Bit doppelt erfasst und ausgeben.

CDR Eine Daten- und Taktrückgewinnung (engl. clock-data-recovery) gewinnt bei einer digitalen Übertragung die Daten im Empfänger zurück und synchronisiert diese mit der Systemtakt. S. 24

CLB In einem konfigurierbaren Logikblock (engl. configurable-logic-block) eines FPGAs lassen sich unterschiedliche logische Funktionen abbilden. Ein FPGA besteht aus einer Matrix mehrerer CLBs. S. 6

CMT Ein Taktmanagerblock (engl. clock-management-tile) ist ein spezieller Funktionsblock eines FPGAs, welcher zur Taktgeneration genutzt wird. S. 9

FPGA Ein integrierter Schaltkreis, mit rekonfigurierbarer, digitaler Logik. S. 5

HDL Hardwarebeschreibungssprache (engl. hardware description language) zur Beschreibung von Operationen in integrierten Schaltungen und deren Design. S. 5

Jitter Jitter ist die zeitliche Schwankung eines Taktes, erkennbar an den Flanken des Taktes und überträgt sich auf Datenübertragungen. Jitter wird in der Regel in UI_{pp} gemessen. S. 11

Jittertoleranzkurve Die Jittertoleranzkurve dient zur Charakterisierung einer CDR und deren Verhalten bei unterschiedlich stark ausgeprägtem Jitter und Frequenzen. S. 38

Logikzelle (LC) Eine Logikzelle ist ein Teilelement eines Slices in einem FPGA. Es besteht in der Regel aus einer LUT, einem Schieberegister oder einem RAM-Block mit einem Multiplexer und einem Flipflop. Zwischen den Logikzellen eines Slices lassen sich die elektrischen Verbindungen programmieren. S. 6

LVDS Das Low-Voltage-Differential-Signaling ist ein Übertragungsstandard für Hochgeschwindigkeits-Datenübertragungen.

Pipelining Pipelining wird in der FPGA-Entwicklung verwendet, um den kombinatorischen Pfad zu verkürzen und höhere Taktfrequenzen des Systems zu erreichen. Durch Setzen eines oder mehrerer Flipflops wird der kritische Pfad verkürzt.

PRBS Ein pseudozufälliger Bitstrom (engl. pseudo-random bit stream) ist in seiner Abfolge bekannt und wird in der Regel bei der Messung des BERs in digitalen Kommunikationssystemen übertragen.

Setup-and-Hold-Zeit Die Setup-and-Hold-Zeit beschreibt den Zeitrahmen, den ein Signal vor und nach dem Erscheinen einer Taktflanke an dem Eingang eines Flipflops stabil anliegen muss. Wird die Zeit verletzt

ist ein korrekter Schaltvorgang des Flipflops nicht gewährleistet. S. 16

Slice Ein Slice ist ein Teilelement eines CLBs und besteht aus mehreren Logikzellen. Zwischen den Slices eines CLBs bestehen programmierbare elektrische Verbindungen. S. 6

SNR Das Signal-Rausch-Verhältnis (engl. signal-noise-ratio) ist ein Maß für die Qualität eines analogen Nutzsignales und beschreibt das Verhältnis zwischen der mittleren Leistung des Nutzsignales und der mittleren Leistung des Rauschens. S. 23

UI Ein *Unit-Intervall* beschreibt die Zeitdauer zur Übertragung eines Symbols und/oder Bits. S. 11,12

VHDL VHDL ist eine Hardwarebeschreibungssprache. S. 5

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort	Datum	Unterschrift im Original