Forschungs-Bericht

Universität, Gebäude 37
D-66123 Saarbrücken

Telefon +49 (0) 6 81/93 02-0
Telefax +49 (0) 6 81/93 02-0
e-mail:
Datum: 30.09.2008

Titel/Thema
Aufbau, Inbetriebnahme und Verifizierung einer optischen Datenstrecke mit Hilfe des Fibre Channel Protokolls

Auftraggeber	**Projektnummer**	Berichtsnummer/Revision	08126-TW	Textseiten	86
Dipl.-Ing. C. Weingard	165956	Vertraulichkeitsgrad		Anlagen	CD-ROM

Verfasser	**Abteilung**	**Telefon**	**Unterschrift**
Schmidt, Markus	Gerätebau		M. Schmidt

Freigabe für Inhalt		**Freigabe für externe Verteilung**	
Dipl.-Ing. C. Weingard		Dipl.-Ing. W. Bähr	W. Bähr, 29.09.08
Name	Unterschrift	Name	Unterschrift

Stichworte (max. 8):
Fibre Channel, SCSI, optische Datenverbindung

Zusammenfassung

Das Ziel der vorliegenden Diplomarbeit bestand darin, zusammen mit einem Kommilitonen eine optische Datenverbindung mit Hilfe des Fibre Channel Standards herzustellen.

Die verschiedenen, umfangreichen Aufgabengebiete wurden unterteilt in die PC-seitige Anbindung einer Fibre Channel Kaufkarte (Host Bus Adapter) und dem Design der optischen Schnittstelle in eine FPGA-Hardware.

Diese Diplomarbeit beschäftigt sich mit dem Impementieren einer optische Schnittstelle in ein FPGA. Außerdem müssen die protokollspezifischen Mechanismen des Fibre Channel Protokolls erfüllt werden. Dafür soll unter anderem ein Fibre Channel Core von Xilinx verwendet werden. Am Ende wird ein Interface angestrebt, welches erlaubt, die Schnittstelle als komplettes Modul in andere Projekte zu importieren.
Im Verlauf der Diplomarbeit stellte sich heraus, dass es nötig war, auf Basis des Fibre Channel Protokolls, ein SCSI Protokoll zu verwenden. Auf Grund von Schwierigkeiten beim Zugriff auf den Host Bus Adapter soll nun das FPGA an das Betriebssystem eine Authentifizierung als Festplatte senden.

Verteiler[1]

[1] 1 x Auftraggeber, 1 x IZFP, falls nur Zusammenfassung zur Kenntnisnahme: „z.K." anfügen.

Fraunhofer Institut für zerstörungsfreie Prüfverfahren

Gerätebau und Systementwicklung

Hochschule für Technik und Wirtschaft des Saarlandes

University of Applied Sciences

Fachbereich Elektrotechnik

Diplomarbeit

2008

Aufbau, Inbetriebnahme und Verifizierung einer optischen Datenstrecke mit Hilfe des Fibre Channel Protokolls

Markus Schmidt
geb. in D-66822 Lebach

Matrikelnummer:
3419525

Betreuer: Dipl. - Ing. (FH) Christoph Weingard — IzfP Saarbrücken
Dipl. - Ing. (FH) Hendrik Theado — IzfP Saarbrücken

Referent: Prof. Dr. - Ing. Volker Schmitt
Hochschule für Technik und Wirtschaft des Saarlandes

Eidesstattliche Erklärung

Die vorliegende Arbeit habe ich eigenständig verfasst.
Dabei standen mir ausschließlich die genannten Hilfsmittel zur Verfügung.
Das Thema wurde in gleicher oder ähnlicher Form noch nicht an anderer Stelle vorgelegt.

Saarbrücken, den 30.09.08

_____

Markus Schmidt

Inhaltsverzeichnis

1 Einleitung

Das Fraunhofer Institut für zerstörungsfreie Prüfverfahren in Saarbrücken entwickelt Prüfgeräte zur Charakterisierung und Analyse von Werkstoffen. Da der Anspruch an die moderne Messtechnik stetig ansteigt immer schneller genauere Messwerte zu liefern, wird proportional dazu auch die produzierte Datenmenge größer. Diese Daten müssen über eine leistungsstarke Datenverbindung von dem Messsystem zu einem Computer gesendet werden, der diese erfasst und auswertet.
Die zur Zeit eingesetzte Technik zur Datenkommunikation ist eine Ethernetverbindung mit 100 Mbit/s, auf deren Basis das Netzwerkprotokoll UDP eingesetzt wird. Diese Datenübertragung erreicht nun auf Grund des Fortschrittes ihre maximale Kapazität. Deswegen ist das Fraunhofer Institut auf der Suche nach einem zukunftssicheren Kommunikationssystem, welches die Ansprüche erfüllt, um das aktuelle System abzulösen.

Als Abschluss des Studiums der Elektrotechnik, Fachrichtung Mikro- und Telekommunikationstechnik, soll nun eine optische Datenübertragungsstrecke mit bis zu 4 GB/s unter der Verwendung des Fibre Channel Protokolls realisiert werden. Die Verbindung soll zwischen einem FPGA-Entwicklungsboard und einem Fibre Channel fähigen PCI-Express Host Bus Adapter etabliert werden.
Das Thema wird im Rahmen von insgesamt zwei Diplomarbeiten bearbeitet. Dabei befasst sich eine Diplomarbeit mit dem softwaremäßigen Zugriff auf den Host Bus Adapter. Anschließend soll eine Testsoftware programmiert werden und ein weiterverwendbares Softwaremodul zur Einbindung in anderen Projekten.
Diese Diplomarbeit beschäftigt sich mit dem zweiten Schwerpunkt. Dazu soll eine optische Schnittstelle in ein FPGA implementiert werden. Außerdem müssen die protokollspezifischen Mechanismen des Fibre Channel Protokolls erfüllt werden. Schließlich wird ein Interface angestrebt, welches erlaubt, die Schnittstelle als komplettes Modul in andere Projekte zu importieren. Zur Realisierung dieser Aufgaben soll unter anderem ein Fibre Channel Core von Xilinx verwendet werden.
Im Verlauf der Diplomarbeit stellte sich heraus, dass es nötig ist, auf Basis des Fibre Channel Protokolls, ein SCSI Protokoll zu verwenden. Auf Grund von Schwierigkeiten

beim Zugriff auf den Host Bus Adapter soll nun das FPGA an das Betriebssystem eine Authentifizierung als Festplatte senden.
Im ersten Teil der Diplomarbeit wird auf das Fibre Channel und SCSI Protokoll eingegangen, um die theoretischen Grundlagen zu schaffen. Der zweite Teil beschäftigt sich mit dem Implementieren der optischen Schnittstelle. Dafür sind folgende Schritte notwendig. Das Konfigurieren der FPGA-Hardware, das Implementieren des Fibre Channel Core und schließlich das Überprüfen der Funktionalität mittels verschiedener Loopbacktests. Der dritte Teil beginnt mit einer Beschreibung eines flexiblen Interfaces mit der Zielsetzung, die Login Prozedur an das Betriebssystem zu senden und auf entsprechende Kommandos zu reagieren.

2 Fibre Channel Protokoll [2, 3, 4, 10,11]

2.1 Struktur und Konzept

Fibre Channel ist eine serielle, High-Speed Datenverbindung, die gleichermaßen für Netzwerke und Massenspeicher geeignet ist. Die Übertragungsrate erreicht zur Zeit eine Geschwindigkeit von bis zu 10 GB/s. Darüber hinaus ist es durch die Unterstützung verschiedener Netzwerkstrukturen sehr flexibel einsetzbar. Die drei verschiedene Netzwerkstrukturen (Abbildung 2.1 - 2.3), in der Fibre Channel betrieben werden kann, sind Switched Fabric, Arbitrated Loop und Point to Point.

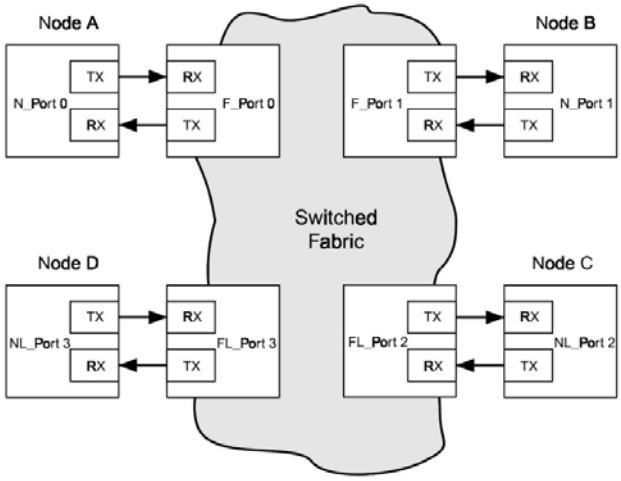

Abbildung 2.1: Switched Fabric Struktur [2]

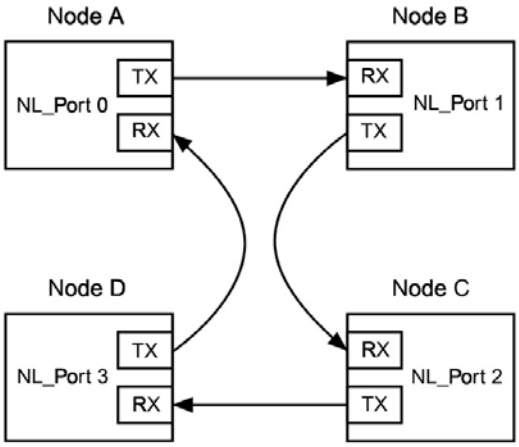

Abbildung 2.2: Arbitrated Loop Struktur [2]

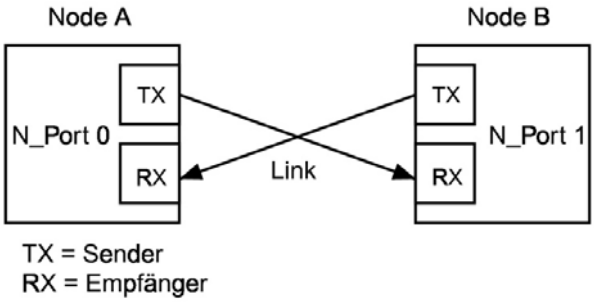

Abbildung 2.3: Point to Point Struktur [2]

Als Übertragungsmedium kann nicht nur Glasfaser genutzt werden, sondern auch Kupferkabel wie Twisted-Pair- oder Koax-Kabel sind zulässig.
Fiber Channel ist ein offener Standard, der durch ANSI definiert ist und alle wichtigen höheren Protokolle unterstützt, wie zum Beispiel IP (Internet Protocol), ATM (Asynchronous Transfer Mode), HIPPI (High Performance Parallel Interface), SCSI (Small Computer System Interface) und so weiter. Das heißt, Fibre Channel verfügt über keinen eigenen Befehlssatz, sondern stellt lediglich den Datentransfer zwischen den einzelnen Teilnehmern her. Wegen der Unterstützung unterschiedlicher Paketlängen, von prinzipiell 0 Bytes bis 2048 Bytes pro Paket, ist der Standard daher einerseits optimal geeignet für kleine I/Os, wie sie typischerweise bei Datenbanken auftreten; andererseits ist er aber auch in der Lage größere Datenmengen, wie sie zum Beispiel in Video-Applikationen vorkommen, effektiv und ohne viel zusätzlichen Overhead zu übertragen.
In den bereits erwähnten Netzwerkstrukturen kann der Anwender verschiedene Verbindungstypen aufbauen, die als sogenannte Serviceklassen bezeichnet werden. Fibre Channel unterscheidet zwischen fünf Serviceklassen (Tabelle 2.1).

Tabelle 2.1: Übersicht der Serviceklassen

Serviceklasse	Beschreibung
1	verbindungsorientiert
2	verbindungslos mit Bestätigung
3	verbindungslos ohne Bestätigung
4	parallele Übertragung
5	parallel und isochron

Die Serviceklasse 1 stellt eine dedizierte Verbindung zwischen Sender und Empfänger her. Alle gesendeten Pakete werden vom Empfänger quittiert. Während des Bestehens dieser Verbindung können keine anderen Teilnehmer die verbundenen Partner ansprechen.

Die Serviceklasse 2 ist eine „verbindungslose“ Methode mit Bestätigung des Datentransfers. Dies bedeutet, dass der Weg, den die Datenpakete nehmen, unbestimmt ist. Die verfügbare Bandbreite kann hierbei unter mehreren Teilnehmern aufgeteilt werden.

Die Serviceklasse 3 ist ähnlich wie die Klasse 2, jedoch ohne Bestätigung des Datentransfers. Diese Verbindungsklasse wird in der Regel bei Massenspeichersystemen genutzt. Da die Empfangsbestätigung durch das höhere SCSI-Protokoll ausgeführt wird, ist auf der unteren FC-Protokollebene keine Empfangsbestätigung notwendig.

In Serviceklasse 4 werden Datenpakete zwischen zwei Teilnehmern in einem Netzwerk unter Ausnutzung mehrerer Verbindungsmöglichkeiten bei garantierter Bandbreite ausgetauscht.

Serviceklasse 5 ist ähnlich wie Klasse 4, jedoch bei zusätzlicher isochroner Datenübertragung.

Da Fibre Channel durch unterschiedliche Kabel, Netzwerkstrukturen und Übertragungsmodi sehr vielseitig anwendbar ist, haben die Entwickler die Definition des FC-Protokolls in fünf verschiedene Protokollschichten (Abbildung 2.4) untergliedert.

In der FC-0-Schicht werden die Definitionen der physikalischen Verbindungen (Kabel, Stecker, Sender und Empfänger) zusammengefasst.

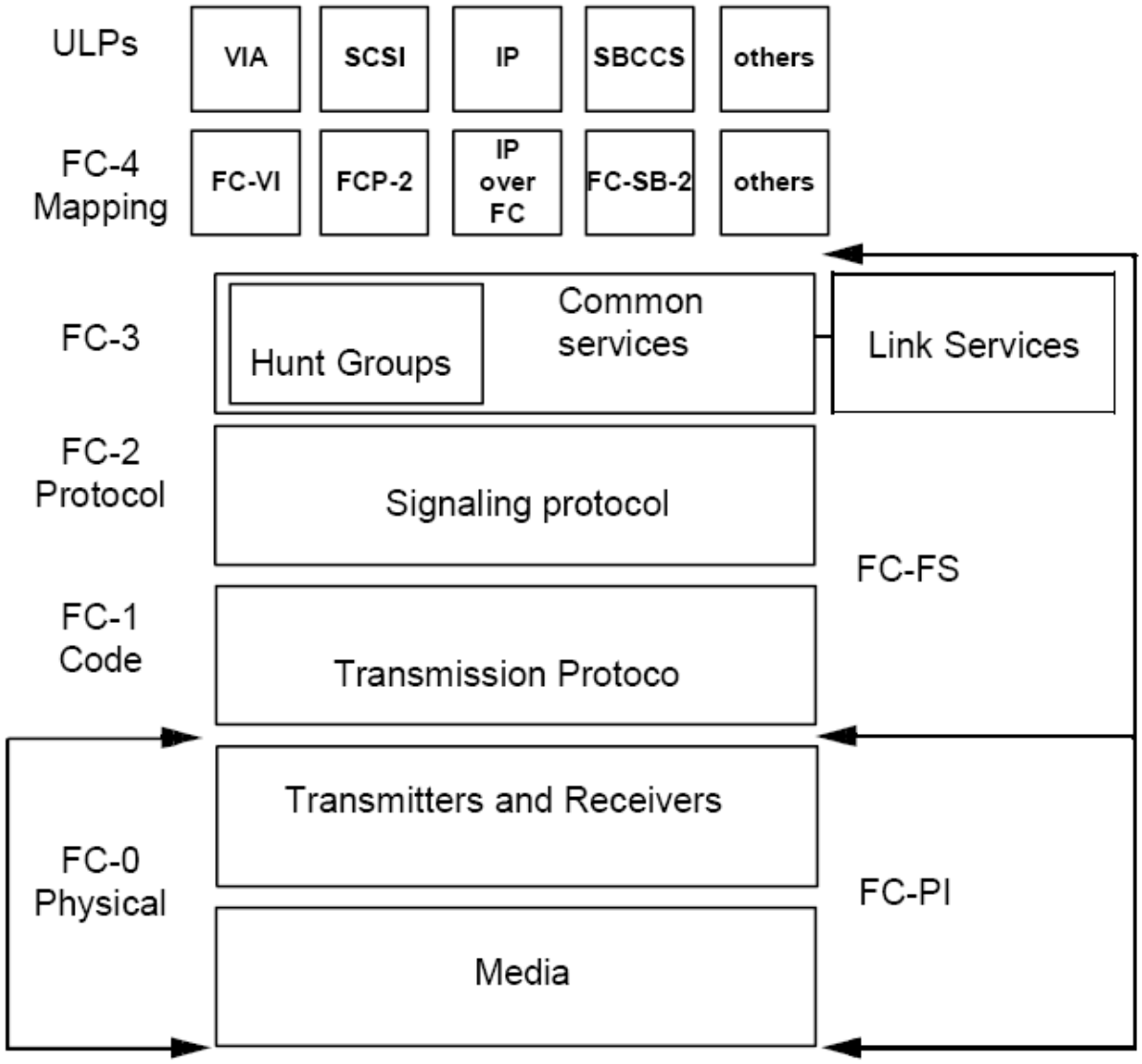

Abbildung 2.4: FC - Protokollschichten [3]

Die FC-1-Schicht regelt die Einkopplung der Datenbits in den Übertragungstakt und steuert die 8/10-Bit-Kodierung-/Dekodierung. Hierdurch wird eine Signalbalance von 50 Prozent erreicht (gleich viele High wie Low Pegel bei der Signalübertragung), was zu einer extrem niedrigen Bit-Fehlerrate führt. Durch diese Methode wird allerdings die zu transportierende Datenmenge um 25 Prozent erhöht, was durch eine entsprechend schnellere Übertragungsrate kompensiert werden muss.

Die FC-2-Schicht ist für die Steuerung des Datenflusses verantwortlich. Hier werden die einzelnen Transferpakete mit Adresse, Daten und CRC-Information zusammengestellt. Diese Schicht übernimmt auch die Bestätigung der Übertragung.

In der FC-3-Schicht werden gemeinsame Funktionalitäten von Gruppen von Netzwerkteilnehmern definiert.

In der FC-4-Schicht wird schließlich die Unterstützung der höheren Protokolle (IP, IEEE 802, HIPPI oder SCSI) geregelt. Insgesamt können in dieser Schicht bis zu 255 verschiedene höhere Protokolle definiert werden, was noch genügend Raum für zukünftige Protokolle lässt.

2.2 8B/10B-Code

Der 8B/10B-Code ist ein Leitungscode der im Fibre Channel Protokoll eingesetzt wird. Dabei werden 8 Bit Daten mit 10 Bit so kodiert, dass zum einen ein Gleichspannungsausgleich gewährleistet ist und zum anderen eine Taktrückgewinnung aus dem Datensignal möglich ist. Die Anzahl von Einsen pro Symbol unterscheidet sich maximal um zwei von der Anzahl der Nullen. Dies ergibt drei Typen von Symbolen, neutrale, positive und negative Ungleichheit (Disparity). Die maximale Länge gleicher Bits ist auf fünf beschränkt und es muss ein Pegelwechsel erfolgen. Dies ermöglicht eine einfache Taktrückgewinnung aus dem Datensignal.
Ein uncodiertes Informationsbyte besteht aus acht Informationsbits (A, B, C, D, E, F, G, H) und einer Kontrollvariable (Z). Die Namensnotation ist mit Zxx.y festgelegt und wird wie in Abbildung 2.5 gebildet. Als Kontrollvariable Z ist entweder ein D oder ein K zulässig. Der Buchstabe D bedeutet, dass es sich um ein Datenwort handelt und der Buchstabe K signalisiert ein Steuerzeichen.

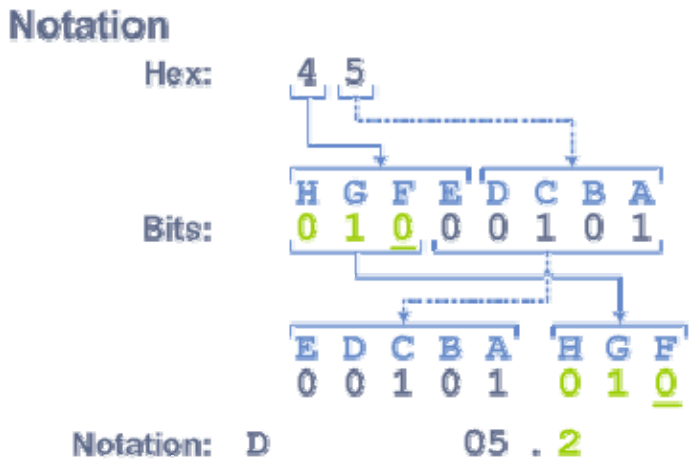

Abbildung 2.5: Beispiel zur Namensnotation

Für xx wird der Dezimalwert aus den Informationsbits A - E eingesetzt und an der Stelle y steht der Dezimalwert aus F - H. Aus diesen Informationen wird dann ein gültiger 10 Bit-Code gebildet. Es gibt Daten- bzw. Steuerzeichen mit einem Codewort, diese besitzen dann eine gleiche Anzahl von Einser und Nullen. Die meisten Daten- bzw. Steuerzeichen besitzen jedoch zwei mögliche 10 Bit Codewörter. Eine mit mehr Einsen als Nullen und eine invertierte Version mit mehr Nullen als Einsen. Der Sender wählt immer so die Codewörter aus, dass ein Gleichspannungsausgleich gewährleistet ist.

2.3 Ordered Sets

Ein Ordered Set (Befehlssatz) ist eine Kombination aus einem Steuerwort und aus drei Datenwörtern und ist somit vier Byte groß. Es werden dabei drei Typen unterschieden.

Als erstes sind die Frame Delimiters (Rahmen-Begrenzungssymbole) zu nennen, die ein Start of Frame (SOF) oder ein End of Frame (EOF) signalisieren und auch die Serviceklasse bestimmen.

Ein weiterer Typ sind die Primitive Signals (Grundsignale). Sie werden benutzt um Ereignisse und Aktionen anzuzeigen. Insgesamt gibt es 16 unterschiedliche Signale. Die beiden wichtigsten sind zum einen IDLE, das gesendet wird, wenn sich der Port im Leerlauf befindet. Zum anderen R_RDY, was anzeigt, dass der Empfänger seinen Speicher geleert hat und bereit für neue Daten ist.

Als letzter Typ gibt es vier sogenannte Primitive Sequences (Grundsequenzen). Diese zeigen den aktuellen Zustand des Anschlusses an und werden solange übertragen, bis sich der Zustand ändert (siehe Abschnitt 2.4).
In Tabelle 2.2 ist eine Übersicht der wichtigsten Befehlssätze zusammengestellt, die auch später verwendet werden.

Tabelle 2.2: Übersicht einiger Befehlssätze

Abkürzung	Funktion	Befehlssatz	Hex Wert
SOF_{i3}	SOF Initiate Class 3	K28.5 - D21.5 - D22.2 - D22.2	BC B5 56 56
SOF_{n3}	SOF Normal Class 3	K28.5 - D21.5 - D22.1 - D22.1	BC B5 36 36
EOF_t	EOF Terminate	K28.5 - D21.4 - D21.3 - D21.3 K28.5 - D21.5 - D21.3 - D21.3	BC 95 75 75 BC B5 75 75
EOF_n	EOF Normal	K28.5 - D21.4 - D21.6 - D21.6 K28.5 - D21.5 - D21.6 - D21.6	BC 95 D5 D5 BC B5 D5 D5
IDLE	Idle	K28.5 - D21.4 - D21.5 - D21.5	BC 95 B5 B5
R_RDY	Receiver Ready	K28.5 - D21.4 - D10.2 - D10.2	BC 95 4A 4A
CLS	Close	K28.5 - D5.4 - D21.5 - D21.5	BC 85 B5 B5
NOS	Not Operational	K28.5 - D21.2 - D31.5 - D5.2	BC 55 BF 45
OLS	Offline	K28.5 - D21.1 - D10.4 D21.2	BC 35 8A 55
LR	Link Reset	K28.5 - D9.2 - D31.5 - D9.2	BC 49 BF 49
LRR	Link Reset Response	K28.5 - D21.1 - D31.5 - D9.2	BC 35 BF 49

2.4 Initialisierung der Verbindung

Es wird hier und auch im weiteren Verlauf der Diplomarbeit immer von einer Point to Point Struktur ausgegangen. Die Initialisierung und die Loginprozedur kann sich bei anderen Netzwerkstrukturen stark unterscheiden und soll hier nicht näher betrachtet werden.

Nach dem Einschalten des Ports beginnt dieser mit dem Senden von NOS Sequenzen (Abbildung 2.6). NOS bedeutet, dass ein Link-Fehler vorliegt bzw. dass der Empfänger zur Zeit nicht erreichbar ist. Diese werden solange gesendet, bis entweder auch ein NOS empfangen wird und der Teilnehmer in den Zustand OLS wechselt oder der Empfänger hat die NOS Sequenzen bereits erhalten, und signalisiert mit OLS, dass er bereits im Offline Zustand ist. Der Teilnehmer, der als erster in den Offline Zustand wechselt, wartet nun bis der andere Teilnehmer dies erkennt und einen Link Reset durchführt. Dies wird dann durch das Senden von LR Sequenzen deutlich gemacht. Der Link Reset muss nun von dem Empfänger in Form einer LRR Sequenz quittiert werden. Wurde diese Quittierung wiederum erkannt, beginnt dieser Teilnehmer mit dem Senden von Idle Sequenzen. Der Empfänger erkennt das und wechselt nun auch in den Idle Zustand. Beide Teilnehmer befinden sich nun im Leerlauf und sind bereit, Daten zu senden bzw. zu empfangen. Die Idle Sequenzen werden nun weiter als Füllwörter gesendet.

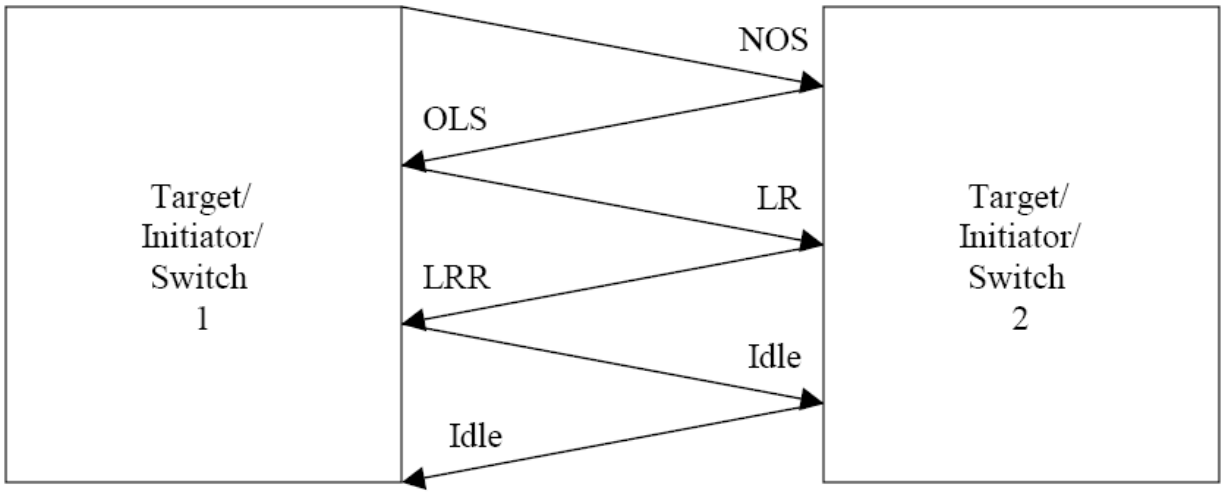

Abbildung 2.6: Ablauf der Initialisierung

2.5 Der Übertragungsrahmen

Der Fibre Channel Rahmen ist wie in Abbildung 2.7 aufgebaut. Die ersten vier Bytes müssen einen Start of Frame Begrenzungssymbol enthalten, um den Anfang des Rahmens zu definieren. Demzufolge müssen auch die letzten vier Bytes einen End of Frame Begrenzungssymbol enthalten, um das Ende des Rahmens zu markieren.

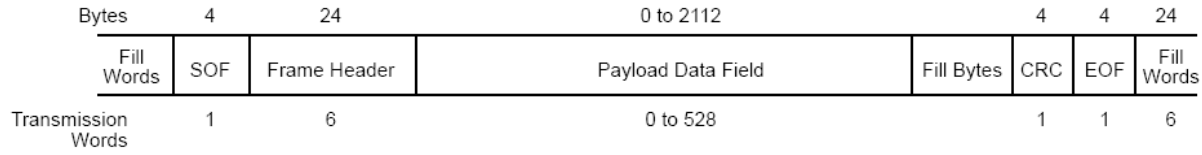

Abbildung 2.7: Aufbau eines Übertragungsrahmens [10]

Im Anschluss an den Start of Frame wird der Header übertragen. Der Header ist 24 Byte groß und soll im nächsten Abschnitt genauer erläutert werden. Nach dem Header folgen die eigentliche Nutzdaten (Payload). Dabei muss beachtet werden, dass maximal 2112 Bytes pro Rahmen übertragen werden können. Danach ist es erforderlich, einen neuen Rahmen zu generieren. Werden weniger Bytes übertragen, ist darauf zu achten, dass die Anzahl der übertragenen Bytes ein ganzes Vielfaches von Vier ist. Um das zu erreichen, werden Füllbytes hinzugefügt. Sollen zum Beispiel 13 Bytes übertragen werden, müssen noch drei Füllbytes ergänzt werden, um so auf eine Gesamtlänge von 16 Bytes zu kommen. Nach den Nutzdaten folgt anschließend ein vier Byte großer CRC (Cyclic Redundancy Check) Wert. Mit dem CRC Wert wird festgestellt, ob ein Fehler während der Übertragung aufgetreten ist. Nach dem CRC Wert wird, das Ende des Rahmens mit einem End of Frame markiert.
Mehrere Rahmen können nun zu einer Sequenz (Abbildung 2.8) zusammengefasst werden. Da ein Rahmen bis zu 2112 Byte Nutzdaten übertragen kann, ist es notwendig bei größeren Datenmengen diese in mehrere Rahmen zu verteilen. Um nun zu gewährleisten, dass diese auch in der richtigen Reihenfolge beim Empfänger ankommen, werden die Rahmen (max. 65536) in einer Sequenz zusammengefasst und mit einer Nummer im Header versehen. So können in einer Sequenz 138 MByte versendet werden.

Ein Exchange ist eine offene Verbindung zwischen zwei Teilnehmern und besteht wiederum aus mehren Sequenzen (Abbildung 2.8). Ein Exchange wird von dem Initiator gestartet. Die erste Sequenz beinhaltet in der Regel einen Befehl. Alle Aktionen, die sich daraufhin auf diesen Befehl beziehen, müssen innerhalb diesem Exchange bleiben. Ist der Befehl abgeschlossen, wird auch der Exchange beendet.

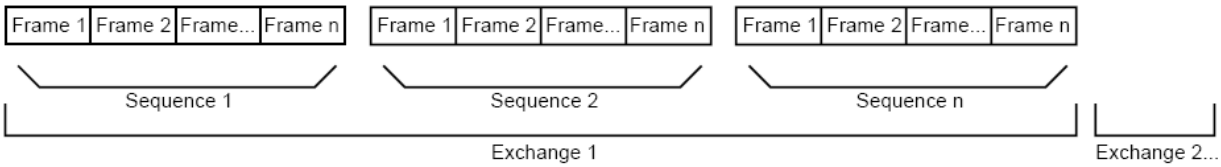

Abbildung 2.8: Gliederung in Sequenz und Exchange [10]

2.6 Der Header

Der Header dient zur Kennzeichnung der Nutzdaten. Er enthält Informationen über den Sender, den Empfänger, über die Art der Daten und wie diese zu behandeln sind. Der Aufbau ist in Abbildung 2.9 gezeigt. Im Folgenden werden die einzelnen Felder kurz erklärt und eine Auswahl an möglichen Inhalten angegeben, die für diese Diplomarbeit relevant sind.

Bits / Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	R_CTL	D_ID		
1	CS_CTL/Priority	S_ID		
2	TYPE	F_CTL		
3	SEQ_ID	DF_CTL	SEQ_CNT	
4	OX_ID		RX_ID	
5	Parameter			

Abbildung 2.9: Aufbau des Headers [3]

R_CTL (Word 0 / Bits 31 - 24)

Das acht Bit große Routing Control Feld identifiziert die Kategorie des Rahmens (Tabelle 2.3) und ist wiederum in zwei vier Bit Felder untergliedert. Von Bit 31 - 28 ist das Routing Feld und von Bit 27 - 24 das Informationsfeld definiert.

Tabelle 2.3: Verwendete Routing Codes

Routing Feld	Informationsfeld	Hex Wert	Frametype	Beschreibung
1000	0001	81	Basic Link Services	Abort Sequence
0010	0010	22	Extended Link Services	Abfrage (Request)
0010	0011	23	Extended Link Services	Antwort (Reply)
0000	0001	01	Device Data Frame	SCSI Data Frame
0000	0101	05	Device Data Frame	SCSI XFER-RDY Frame
0000	0110	06	Device Data Frame	SCSI Commando Frame
0000	0111	07	Device Data Frame	SCSI Response Frame

D_ID (Word 0 / Bits 23 - 0)

In diesem Feld wird die Empfängeradresse (Destination ID) versendet. Dabei sind bereits einige Bereiche als Sonderadressen reserviert. Zwei davon sind FF FF FFh (Broadcast) und FF FF FEh (F Port). Die Adressen 00 00 00h bis FF FC 00h können bedenkenlos verwendet werden.

CS_CTL / Priority (Word 1 / Bit 31 - 24)

Abhängig von Bit 17 im Frame Control Feld hat das Feld die Bedeutung CS_CTL oder Priority. Da jedoch dieses Feld im Rahmen dieser Arbeit nicht verwendet wird, soll es hier auch nicht näher erläutert werden. Es wird als Defaultwert immer 00h angenommen.

S_ID (Word 1 / Bit 23 - 0)

Dieses Feld ist für die Adresse des Absenders (Source ID) reserviert. Es gilt der gleiche Adressbereich und die Besonderheiten wie bei der Empfängeradresse (D_ID).

TYPE (Word 2 / Bit 31 - 24)

Das Feld hat die gleiche Aufgabe wie das R_CTL Feld. Es kennzeichnet den Typ des aktuellen Rahmen. Dabei müssen die Angaben im R_CTL Feld und in diesem Feld konsistent sein. Verwendet werden hier nur drei verschiedene Typen, die in der Tabelle 2.4 aufgelistet sind.

Tabelle 2.4: Verwendete Type Codes

Type Code	Frametype
00	Basic Link Services
01	Extended Link Services
08	SCSI Frame

F_CTL (Word 2 / Bit 23 - 0)

Das Frame Control Feld enthält verschiedene Kontrollinformationen in Bezug auf den Rahmen. Es werden vor allem Angaben über den Exchange und die Sequenz gemacht. In den letzten beiden Bits wird auch über die Anzahl der Füllbytes informiert, die am Ende der Nutzdaten hinzugefügt werden müssen, um ein Vielfaches von Vier zu erreichen. Die Tabelle 2.5 verschafft einen Überblick zu den einzelnen Bits und deren Bedeutung. Auch hier werden nur die Bits näher betrachtet, die im späteren Verlauf von Bedeutung sind.

Tabelle 2.5: Übersicht der Frame Control Bits

Bit	Bit Name	Beschreibung
23	Exchange Context	0 = Frame vom Exchange Absender 1 = Frame vom Exchange Empfänger
22	Sequence Context	0 = Sequenz Absender 1 = Sequenz Empfänger
21	First Sequence	0 = Nicht die erste Sequenz des Exchanges 1 = Erste Sequenz des Exchanges
20	Last Sequence	0 = Nicht die letzte Sequenz des Exchanges 1 = Letzte Sequenz des Exchanges
19	End Sequence	0 = Nicht der letzte Frame des Exchanges 1 = Letzter Frame des Exchanges
18	End Connection	Nicht verwendet, wird auf Null gesetzt
17	CS_CTL / Priority enable	Nicht verwendet, wird auf Null gesetzt
16	Sequence Initiative	0 = Sequenzinitiative behalten 1 = Sequenzinitiative abgeben
15	X_ID reassigned	Nicht verwendet, wird auf Null gesetzt
14	Invalidate X_ID	Nicht verwendet, wird auf Null gesetzt
13-12	ACK Form	Nicht verwendet, wird auf Null gesetzt
11	Data Compression	Nicht verwendet, wird auf Null gesetzt
10	Data Encryption	Nicht verwendet, wird auf Null gesetzt
9	Retransmitted Sequence	Nicht verwendet, wird auf Null gesetzt
8	Unidirectional Transmit	Nicht verwendet, wird auf Null gesetzt
7-6	Continue Sequence Condition	Nicht verwendet, wird auf Null gesetzt
5-4	Abort Sequence Condition	Nicht verwendet, wird auf Null gesetzt
3	Relative offset present	0 = Parameterfeld ist unbedeutend 1 = Parameterfeld beinhaltet den relativen Offset

2	Exchange reassembly	Nicht verwendet, wird auf Null gesetzt
1-0	Fill Data Bytes	Die Anzahl der Füllbytes am Ende des Payloads 00 = Null Füllbytes 01 = Ein Füllbyte 10 = Zwei Füllbytes 11 = Drei Füllbytes

SEQ_ID (Word 3 / Bit 31 - 24)

Die Sequenz Nummer wird vom Sequenzinitiator vergeben und kennzeichnet den Rahmen. Alle Übertragungsahmen, die sich auf den Initiatorrahmen beziehen, müssen die gleiche Sequenznummer in diesem Feld haben.

DF_CTL (Word 3 / Bit 23 - 16)

Das Data Control Feld kennzeichnet den optionalen Header. Dieser wird jedoch bei SCSI over Fibre Channel nicht benötigt und bekommt den Wert 00h zugewiesen.

SEQ_CNT (Word 3 / Bit 15 - 0)

Der Sequenzzähler bestimmt die Reihenfolge der Rahmen einer Sequenz. Beginnend mit Null, wird bei jedem neuen Rahmen in der Sequenz der Zähler inkrementiert. So ist es dem Empfänger möglich, die Rahmenreihenfolge zu kontrollieren und gegebenenfalls in die richtige Reihenfolge zu ordnen.

OX_ID (Word 4 / Bit 31 - 16)

Die OX_ID (Originator Exchange ID) wird vom Exchange Absender vergeben und kennzeichnet den Exchange. Alle Sequenzen, die sich auf diesen Exchange beziehen, müssen die gleiche OX_ID besitzen.

RX_ID (Word 4 / Bit 15 - 0)

Die RX_ID (Responder Exchange ID) wird vom Exchange Empfänger vergeben. Da dieses Funktion nicht genutzt wird, muss das Feld den Wert FF FFh beinhalten.

Parameter (Word 5 / Bit 31 - 0)

In dem Parameter Feld kann ein relativer Offset definiert werden, der das erste Byte der Nutzdaten kennzeichnet. Soll diese Funktion genutzt werden, muss das Bit 3 im F_CTL gesetzt werden. Im anderen Fall hat dieses Feld keine Funktion.

2.7 Link Service

Die Fibre Channel Link Service Rahmen werden dazu verwendet, um Aufgaben der Fibre Channel Schichten auszuführen. Sie werden benutzt, um zwischen zwei Geräten die Betriebsparameter festzulegen, Kanalfehler zu reparieren und den physikalischen Status zu überprüfen. Die Link Service Rahmen sind in Basic Link Service und Extended Link Service unterteilt. Die Auswahl des Service findet im Header statt. Die Felder R_CTL (Tabelle 2.3) und TYPE (Tabelle 2.4) bestimmen, welcher Service verwendet werden soll.

Der Basic Link Service umfasst sechs Funktionen, die über das R_CTL Feld ausgewählt werden. Interessant für diese Diplomarbeit ist nur die Abbruchsequenz (ABTS). Die Abbruchsequenz wird vom Initiator gesendet, wenn ein ausstehender Exchange nach einer Zeitüberschreitung noch unbeantwortet bzw. unvollständig ist. Bei der Sequenz werden keine Nutzdaten gesendet. Alle nötigen Informationen werden im Header übertragen. Das OX_ID Feld beinhaltet dabei die Nummer des fehlerhaften Exchange.

Die Funktionen des Extended Link Services werden im Gegensatz zum Basic Link Service nicht im R_CTL Feld ausgewählt, sondern über das erste Datenwort in den Nutzdaten. In den Feldern R_CTL und TYPE wird lediglich eingestellt, dass ein Extended Link Service verwendet wird und R_CTL hat zusätzlich noch die Aufgabe, den Frame als eine Anfrage oder eine Antwort zu kennzeichnen. Die Bits 31 - 24 im ersten Datenwort der Nutzdaten müssen den Extended Link Service Commando Code (ELS Code) beinhalten.

Tabelle 2.6: Einige ELS Codes

Hex Wert	Abkürzung	Beschreibung
02	LS_ACC	Link Service Accept
03	PLOGI	N_Port Login
04	FLOGI	F_Port Login
05	LOGO	Logout
20	PRLI	Process Login
21	PRLO	Process Logout

Die wichtigsten Codes sind in Tabelle 2.6 aufgeführt. Die Informationen zu dem ausgewählten Service werden in dem Nutzdatenblock übertragen. Der Block besitzt, je nach Service, eine feste Struktur, die eingehalten werden muss. Die Login Strukturen werden im folgenden Abschnitt näher erläutert.

2.8 Login Prozedur

2.8.1 Fabric Login

Nachdem eine Verbindung initialisiert wurde (siehe Abschnitt 2.4) versucht der Teilnehmer einen Login in ein vorhandenes Netzwerk zu starten und sendet einen Fabric Login. In der Tabelle 2.7 ist der Aufbau eines Fabric Login Rahmens mit den verwendeten Werten dargestellt. Die Werte XX, YY und ZZ sind dabei abhängig vom Sender und haben in jeder Anfrage unterschiedliche Inhalte. Alle Beispiele werden immer aus der Sicht des Empfängers, also dem programmierten FPGA dargestellt.
In dem Fabric Login werden die Übertragungsparameter festgelegt. Diese sind unter anderem die verwendete Serviceklasse, Buffergröße, Buffer to Buffer Credit, Node und Port Name.
Der Teilnehmer sendet nun solange diesen Rahmen, bis er eine Antwort bekommt. Bei einer Point to Point Struktur ist jedoch kein bestehendes Netzwerk vorhanden, daher muss die Antwort auf den Fabric Login darauf hinweisen. Dies teilt der andere Teilnehmer mit, indem das Bit 28 des zweiten Datenwortes in den Common Service Parameter auf Null gesetzt wird. Des Weiteren wird als ELS Code ein Link Service Accept (LS_Accept) eingesetzt. Der LS_Accept Rahmen ist wie der Login Rahmen strukturiert. Auch hier werden bestimmte Übertragungsparameter vorab festgelegt bzw. bestätigt. Als Beispiel wäre hier der Node und Port Name der Gegenstelle zu nennen. Ist die Abwicklung des Fabric Login erfolgreich abgeschlossen, geht der Teilnehmer über zum Port Login.

Tabelle 2.7: Aufbau des Fabric Login Rahmens

Bedeutung			Anfrage				Antwort			
Start of Frame			BC	B5	56	56	BC	B5	56	56
R_CTL	D_ID		22	FF	FF	FF	23	00	00	00
CS_CTL	S_ID		00	00	00	00	00	00	00	00
TYPE	F_CTL		01	29	00	00	01	99	00	00
SEQ_ID	DF_CTL	SEQ_CNT	XX	00	00	00	XX	00	00	00
OX_ID		RX_ID	YY	YY	FF	FF	YY	YY	FF	FF
Parameter			00	00	00	00	00	00	00	00
ELS Commando Code			04	00	00	00	02	00	00	00
Common Service Parameter			20	20	00	08	20	20	00	08
			00	00	02	00	88	00	00	00
			00	00	00	00	00	00	27	10
			00	00	00	00	00	00	07	D0
Port Name			21	00	00	1B	21	00	00	00
			32	10	5A	D5	00	00	00	01
Node Name			20	00	00	1B	20	00	00	00
			32	10	5A	D5	00	00	00	01
Class 1 + 6 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Class 2 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Class 3 Service Parameter			88	00	00	00	88	00	00	00
			00	00	02	00	00	00	02	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Class 4 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Vendor Version Level			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
CRC			ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ
End of Frame			BC	95	75	75	BC	95	75	75

XX = Sequenz ID

YY YY = OX_ID

ZZ ZZ ZZ ZZ = CRC Wert

2.8.2 Port Login

Der Port Login läuft ähnlich ab wie der Fabric Login. Während der Fabric Login im eigentlichen Sinne die Parameter zwischen Port und Switch festlegt, werden hier die Parameter zwischen Port und Port bestimmt. Auch der Rahmenaufbau (Tabelle 2.8) unterscheidet sich nicht vom Fabric Login. Anzumerken ist hier, dass der Initiator eine Port ID an den Empfänger Port vergibt. Dies wird im D_ID Feld übermittelt.

Tabelle 2.8: Aufbau des Port Login Rahmens

Bedeutung			Anfrage				Antwort			
Start of Frame			BC	B5	56	56	BC	B5	56	56
R_CTL	D_ID		22	00	00	E8	23	00	00	EF
CS_CTL	S_ID		00	00	00	EF	00	00	00	E8
TYPE	F_CTL		01	29	00	00	01	99	00	00
SEQ_ID	DF_CTL	SEQ_CNT	XX	00	00	00	XX	00	00	00
OX_ID		RX_ID	YY	YY	FF	FF	YY	YY	FF	FF
Parameter			00	00	00	00	00	00	00	00
ELS Commando Code			03	00	00	00	02	00	00	00
Common Service Parameter			20	20	00	08	20	20	00	08
			88	00	02	00	88	00	02	00
			00	FF	00	1F	00	FF	00	1F
			00	00	07	D0	00	00	07	D0
Port Name			21	00	00	1B	21	00	00	00
			32	10	5A	D5	00	00	00	01
Node Name			20	00	00	1B	20	00	00	00
			32	10	5A	D5	00	00	00	01
Class 1 + 6 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Class 2 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Class 3 Service Parameter			88	00	00	00	88	00	00	00
			00	00	02	00	00	00	02	00
			00	FF	00	00	00	FF	00	00
			00	01	00	00	00	01	00	00
Class 4 Service Parameter			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
Vendor Version Level			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
CRC			ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ
End of Frame			BC	95	75	75	BC	95	75	75

2.8.3 Prozess Login

Der Prozess Login folgt nach erfolgreicher Port Login Prozedur. Die wichtigste Funktionalität des Prozess Logins ist die Definition eines Protokolls, dass für höhere Protokollschichten geeignet ist. In diesem Fall wird ein SCSI Protokoll verwendet, um mit dem Host Bus Adapter kommunizieren zu können.

Der Prozess Login Frame ist wie in Tabelle 2.9 aufgebaut. In der Service Parameter Page werden wieder wichtige Definitionen vereinbart. Das Byte 0 wird zum Beispiel mit einer 08h besetzt, um zu kennzeichnen, dass SCSI als weiteres Protokoll verwendet wird. Eine weitere wichtige Vereinbarung ist die Zuweisung von Target und Initiator. Dies geschieht mit dem Setzen von Bit 4 für Target bzw. Bit 5 für Initiator im letzten Datenwort der Service Parameter Page. Als Initiator wird hier der Host Bus Adapter festgelegt, da es somit später einfacher sein wird, auf Anfragen zu reagieren. Die FPGA Karte soll nur als Target mit Festplatten Charakter reagieren.

Tabelle 2.9: Aufbau des Prozess Login Rahmens

Bedeutung			Anfrage				Antwort			
Start of Frame			BC	B5	56	56	BC	B5	56	56
R_CTL	D_ID		22	00	00	E8	23	00	00	EF
CS_CTL	S_ID		00	00	00	EF	00	00	00	E8
TYPE	F_CTL		01	29	00	00	01	99	00	00
SEQ_ID	DF_CTL	SEQ_CNT	XX	00	00	00	XX	00	00	00
OX_ID		RX_ID	YY	YY	FF	FF	YY	YY	FF	FF
Parameter			00	00	00	00	00	00	00	00
ELS Commando Code			20	01	00	14	02	10	00	14
Service Parameter Page			08	00	20	00	08	00	21	00
			00	00	00	00	00	00	00	00
			00	00	00	00	00	00	00	00
			00	00	00	22	00	00	00	12
CRC			ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ	ZZ
End of Frame			BC	95	75	75	BC	95	75	75

3 SCSI Protokoll [4, 5, 6, 10, 11]

3.1 Allgemeines zu SCSI

SCSI steht für Small Computer System Interface und wurde ursprünglich als Standard für eine parallele Schnittstelle entwickelt, um Peripheriegeräte wie Drucker oder Scanner an den Computer anzuschließen. Bis heute wurde SCSI immer weiterentwickelt. Der Standard taucht mittlerweile in sehr unterschiedlichen Anwendungsgebieten auf und es gibt zahlreiche abgeleitete Standards wie SAS oder iSCSI.

Definitionen über Stecker, Kabel, Netzwerkstrukturen und Übertragungsparameter sind hier jedoch unbedeutend, da diese bereits mit Fibre Channel festgelegt sind und die SCSI Informationen in den Fibre Channel Rahmen versendet werden. Daher ist nur die Befehlsstruktur von SCSI von Bedeutung und soll im folgenden näher betrachtet werden.

Das SCSI Protokoll kennt vier Arten von Rahmen, den Kommandorahmen (FCP CMD), Datenrahmen (FCP DATA), Bestätigungsrahmen (FCP XFER_RDY) und Statusrückmelderahmen (FCP RSP). Die Arten des Rahmens wird im Header des Fibre Channel Rahmens gekennzeichnet. Diese vier Arten werden in den folgenden Abschnitten näher betrachtet.

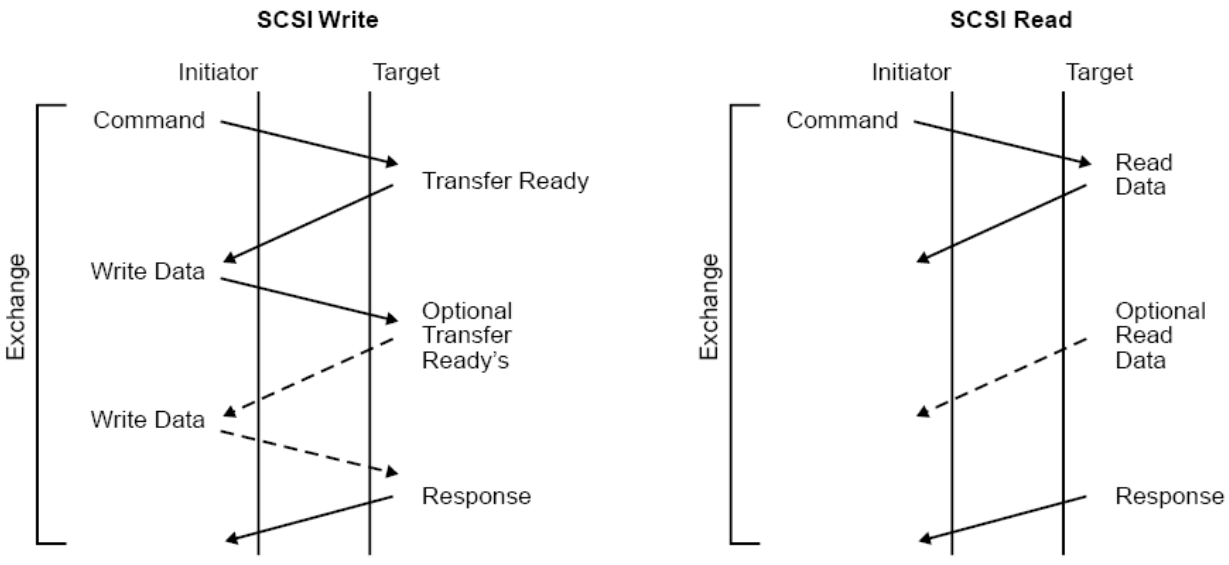

Abbildung 3.1: Schematische Darstellung eines Lese- und Schreibzyklus [10]

Die SCSI Informationen werden Eins zu Eins in den Fibre Channel Nutzdatenblock kopiert. Das bedeutet, dass erste Byte vom SCSI Rahmen wird im ersten Byte des Fibre Channel Nutzdatenblocks übertragen, und so weiter.
In Abbildung 3.1 ist eine schematische Darstellung eines Lese- und Schreibzyklus zu erkennen. Der Lesezyklus wird vom Initiator mit dem Senden eines Kommandos gestartet. Daraufhin sendet der Empfänger die angeforderten Informationen in einem Datenrahmen zurück. Sind alle Daten übertragen, wird noch ein Rückmelderahmen gesendet, um den Status an den Initiator mitzuteilen.
Auch der Schreibzyklus geht vom Initiator aus. Der Empfänger muss daraufhin dem Initiator mittels eines XFER_RDY Rahmens signalisieren, dass er für die Datenaufnahme bereit ist. Der Initiator beginnt daraufhin mit dem Senden der Informationen. Nach jedem Datenrahmen kann optional ein weiterer Bestätigungsrahmen an den Initiator gesendet werden. Dies wird vorher im Prozess Login vereinbart. Nach Beendigung des Schreibzyklus sendet der Empfänger noch ein Statusrückmelderahmen an den Initiator.

3.2 FCP CMD

Der Header eines Kommandorahmens ist in Abbildung 3.2 gezeigt. Er wird durch eine 06h im R_CTL Feld kenntlich gemacht. Das F_CTL Feld muss den Wert 29 00 00h besitzen. Das kennzeichnet den Rahmen als erste Sequenz eines Exchange und gleichzeitig als letzter Rahmen dieser Sequenz. Dazu wird die Sequenzinitiative an den Empfänger übermittelt, der nun auf dieses Kommando reagieren soll. Die Felder D_ID, S_ID, SEQ_ID und OX_ID müssen natürlich ebenfalls vom Initiator einen gültigen Wert zugewiesen bekommen. Alle sonstigen Felder werden nicht benötigt und werden auf einen Standardwert gesetzt.
Der Nutzdatenblock gestaltet sich wie in Abbildung 3.3. Die Logical Unit Number (LUN) ist die physikalische Adresse des Gerätes. Für die Diplomarbeit ist nur eine LUN 0 gültig. Bei einer anderen Adresse muss im FCP_RSP Rahmen der Status Check Condition zurückgegeben werden.
Die Felder Read Data und Write Data werden in Abhängigkeit des verwendeten Befehls gesetzt und kennzeichnen somit einen Lese- oder Schreibzyklus.

Bit / Byte	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	0
	R_CTL							
1	(MSB)							
2	D_ID							
3								(LSB)
4	0	0	0	0	0	0	0	0
	Reserved							
5	(MSB)							
6	S_ID							
7								(LSB)
8	0	0	0	0	1	0	0	0
	Type							
9	(MSB)							
10	F_CTL							
11								(LSB)
12	SEQ_ID							
13	0	0	0	0	0	0	0	0
	DF_CTL							
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
	SEQ_CNT							
16	(MSB)							
17	OX_ID							(LSB)
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
	RX_ID							
20	(MSB)							
21	Parameter							
22								
23								(LSB)

Abbildung 3.2: Header des FCP CMD Rahmens [10]

Das letzte Feld (DL) im Datenblock spezifiziert die maximale Datenlänge, die der Initiator vom Empfänger, auf Grund des gesendeten Befehls, erwartet. Werden nun mehr bzw. weniger Daten versendet, muss dies im darauf folgenden FCP RSP Rahmen gekennzeichnet sein.

Das Feld CDB steht für Command Descriptor Block und beinhaltet den auszuführenden Befehl. Die Befehle werden in acht Gruppen unterteilt und je nach Gruppe ist der CDB Aufbau unterschiedlich.

Die wichtigsten Gruppen sind die Gruppe 0, die alle 6 Byte Befehle umfasst und die Gruppen 1 und 2, die alle 10 Byte Befehle einschließen. Im ersten Byte des CDB (12 Byte im Nutzdatenblock) wird immer der Befehlscode übertragen. Bit 5 - 7 des Befehlscode selektieren die Gruppe und Bit 0 - 4 wählen das Kommando aus. Der Abschnitt 3.6 bietet eine Beschreibung der wichtigsten Befehle, die in der Diplomarbeit verwendet werden. Der weitere Aufbau des CDB ist ab hier von Gruppe zu Gruppe unterschiedlich und im weiteren Verlauf der Diplomarbeit nicht mehr relevant. Alle benötigten Daten wie LUN, Datenlänge, Befehlscode und Lesen oder Schreiben sind bekannt.

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
:	Logical Unit Number							
7								(LSB)
8	0	0	0	0	0	0	0	0
	Reserved							
9	0	0	0	0	0	Task Attribute		
	Reserved							
10	Term Task	Clear ACA	Target Reset	0	0	Clear Task Set	Abort Task Set	0 Reserved
				Reserved				
11	0	0	0	0	0	0	Read Data	Write Data
	Reserved							
12	(MSB)							
:	CDB							
27								(LSB)
28	(MSB)							
29	DL							
30								
31								(LSB)

Abbildung 3.3: Datenaufbau des FCP CMD Rahmens [10]

3.3 FCP XFER_RDY

Ein XFER_RDY Rahmen hat die Aufgabe bei einem Schreibzyklus dem Initiator mitzuteilen, dass der Empfänger für die Datenaufnahme bereit ist.

Dieser Rahmentyp wird durch eine 05h im R_CTL Feld des Headers gekennzeichnet (Abbildung 3.4). Das F_CTL Feld wird auf den Wert 89 00 00h gesetzt und die Sequenz ID bekommt den Wert FFh zugewiesen.

Bit / Byte	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1
	R_CTL							
1	(MSB)							
2	D_ID							
3								(LSB)
4	0	0	0	0	0	0	0	0
	Reserved							
5	(MSB)							
6	S_ID							
7								(LSB)
8	0	0	0	0	1	0	0	0
	Type							
9	(MSB)							
10	F_CTL							
11								(LSB)
12	SEQ_ID							
13	0	0	0	0	0	0	0	0
	DF_CTL							
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
	SEQ_CNT							
16	(MSB)							
17	OX_ID							(LSB)
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
	RX_ID							
20	(MSB)							
21	Parameter							
22								
23								(LSB)

Abbildung 3.4: Header des FCP XFER_RDY Rahmens [10]

Im Nutzdatenblock (Abbildung 3.5).wird ein relativer Offset übertragen und in dem Feld Burst Length wird die Menge an Daten übermittelt, die in der nächsten Sequenz übertragen werden können. Damit wird sichergestellt, dass der Empfänger immer über genügend freie Ressourcen zur Datenverarbeitung verfügt.

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	Relative Offset							
2								
3								(LSB)
4	(MSB)							
5	Burst Length							
6								
7								(LSB)
8	(MSB)							
9	Reserved							
10								
11								(LSB)

Abbildung 3.5: Datenaufbau des FCP XFER_RDY Rahmens [10]

3.4 FCP DATA

Damit die gewünschten Informationen übertragen werden können, müssen sie in einem Datenrahmen untergebracht werden. Die Daten werden dabei byteweise in den Nutzdatenblock gepackt.

Der Header ist wie in Abbildung 3.6 aufgebaut. Der Rahmentyp wird wieder durch das R_CTL Feld mit dem Wert 01h gekennzeichnet. Für Datenpakete vom Initiator zum Empfänger wird im F_CTL Feld der Wert 00 00 0Xh eingetragen. Wird der letzte Rahmen einer Sequenz gesendet, muss der Wert auf 09 00 0Xh geändert werden. Für Daten zum Initiator muss das Feld auf den Wert 80 00 0Xh gesetzt werden. Auch hier wird der letzte Rahmen einer Sequenz wieder durch den Wert 88 00 0Xh gekennzeichnet. Die Sequenzinitiative wird noch behalten, da im Anschluss ein FCP RSP Rahmen gesendet werden muss. Der Wert für das X wird durch die Anzahl der übertragenen Füllbytes bestimmt und kann von Rahmen zu Rahmen unterschiedlich sein.

Die Sequenz ID wird bei Übertragungen zum Empfänger beibehalten. Für Übertragungen zum Initiator wird die Sequenz ID auf 00h gesetzt und für jede weitere Sequenz inkrementiert.

Bit / Byte	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1
	R_CTL							
1	(MSB)							
2	D_ID							
3								(LSB)
4	0	0	0	0	0	0	0	0
	Reserved							
5	(MSB)							
6	S_ID							
7								(LSB)
8	0	0	0	0	1	0	0	0
	Type							
9	(MSB)							
10	F_CTL							
11								(LSB)
12	SEQ_ID							
13	0	0	0	0	0	0	0	0
	DF_CTL							
14	(MSB)							
15	SEQ_CNT							(LSB)
16	(MSB)							
17	OX_ID							(LSB)
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
	RX_ID							
20	(MSB)							
21	RO							
22								
23								(LSB)

Abbildung 3.6: Header des FCP DATA Rahmens [10]

3.5 FCP RSP

Der Rahmen dient dazu, dem Initiator den aktuellen Status des Empfängers mitzuteilen. Hier werden auch Informationen zu aufgetretenen Fehlern und ein Abweichen der vom Empfänger zu erwartenden Datenmenge übermittelt.
Im Header (Abbildung 3.7) wird durch eine 07h im R_CTL Feld kenntlich gemacht, dass es sich um ein FCP RSP Rahmen handelt. Das F_CTL Feld besitzt immer den Wert 99 00 00h, da es immer der letzte Rahmen eines Exchanges ist. Auch die Sequenz ID wird mit dem Wert FFh festgesetzt.

Bit Byte	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	1
	R_CTL							
1	(MSB)							
2	D_ID							
3								(LSB)
4	0	0	0	0	0	0	0	0
	Reserved							
5	(MSB)							
6	S_ID							
7								(LSB)
8	0	0	0	0	1	0	0	0
	Type							
9	(MSB)							
10	F_CTL							
11								(LSB)
12	SEQ_ID							
13	0	0	0	0	0	0	0	0
	DF_CTL							
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
	SEQ_CNT							
16	(MSB)							
17	OX_ID							(LSB)
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
	RX_ID							
20	(MSB)							
21	Parameter							
22								
23								(LSB)

Abbildung 3.7: Header des FCP RSP Rahmens [10]

Der Nutzdatenblock des FCP RSP Rahmens ist in Abbildung 3.8 ersichtlich. In Byte 10 gibt es vier wichtige Flags. Das Flag an Bit 3 muss gesetzt werden, falls weniger Bytes gesendet wurden als vom Empfänger erwartet. Die Differenz muss im Residual Count Feld als Hexwert angegeben werden. Das Bit 2 wird gesetzt, falls eine größere Datenmenge gesendet wurde. Auch hier muss die Differenz im Residual Count Feld mitgeteilt werden. Mit dem Setzen von Bit 1 bzw. 0 wird das entsprechende Feld im Datenblock gültig. Mit dem Eintragen einer Längenangabe in diese Felder können zusätzliche Informationen im Rahmen übertragen werden. Diese werden dann in den Feldern Response Information und Extended Sense Information mitgeteilt. Das ermöglicht eine präzisere Fehlerbeschreibung.

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
:				Reserved				
9								(LSB)
10	0	0	0	0	Resid Under Run	Resid Over Run	Sense Length Valid	RSP Length Valid
			Reserved					
11	SCSI Status							
12	(MSB)							
13				Residual Count				
14								
15								(LSB)
16	(MSB)							
17				Length of Sense Information				
18								
19								(LSB)
20	(MSB)							
21				Length of Response Information				
22								
23								(LSB)
24	(MSB)							
:				Response Information				
31								(LSB)
32	(MSB)							
:				SCSI Extended Sense Information				
51								(LSB)

Abbildung 3.8: Datenaufbau des FCP RSP Rahmens [10]

Der SCSI Status wird in Byte 11 übertragen. Die Tabelle 3.1 zeigt eine Übersicht der drei wichtigsten Zustände und deren Bedeutung.

Tabelle 3.1: SCSI Status

Hex Wert	Name	Beschreibung
00	Good	Das Kommando wurde erfolgreich ausgeführt.
02	Check Condition	Es ist ein Fehler aufgetreten. Die Fehlerursache wird im Extended Sense Information Feld übertragen.
08	Busy	Der Empfänger ist beschäftigt und nicht in der Lage einen Befehl auszuführen.

3.6 SCSI Kommandos

3.6.1 Inquiry

Das Inquiry Kommando (Kommandowort 12h) ist der erste Befehl, der nach einem erfolgreichen Prozess Login gesendet wird. Es werden Informationen bzw. Parameter über das angeschlossene Gerät angefordert. In Abbildung 3.9 ist der CDB Aufbau des Inquiry Kommandos zu erkennen. Das EVPD (Enable Vital Product Data) Flag entscheidet welche Informationen der Initiator benötigt. Ist das Flag nicht gesetzt, wird eine Standard Inquiry Antwort erwartet (Abbildung 3.10).

Bit / Byte	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	1	0
1	0	0	0	0 Reserved	0	0	0	EVPD
2	Page Code							
3	0	0	0	0	0	0	0	0
4	Allocation Length (in bytes)							
5	Control							

Abbildung 3.9: Inquiry Kommando [10]

Im Standard Inquiry Format werden Angaben über das Gerät und deren Funktion gemacht. Der Gerätetyp wird im Byte 0 bestimmt. Die Tabelle 3.2 zeigt eine Auswahl an unterstützten Gerätetypen. Der Herstellername, der Produktname, die Versionsnummer und die Seriennummer sind in den entsprechenden Feldern als ASCII anzugeben. Die Tabelle 3.3 zeigt exemplarisch einen Ablauf einer Standard Inquiry Anfrage. Diese wird mit dem entsprechenden Antwort Rahmen und dazugehörigem FCP RSP Rahmen beantwortet. Die verwendeten Werte sollen eine gewöhnliche Festplatte beschreiben.

Bit Byte	7	6	5	4	3	2	1	0
0	Peripheral Qualifier		Peripheral Device Type					
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	ANSI-Approved Version		
3	0 AENC	0 TRMIOP	0 NACA	0 HiSupport	Response Data Format			
4	Additional Length							
5	0	0	0	0	0	0	0	0
6	BQue	ENCSER	Port	Dual P	0	0	0	0
7	RelAdr	0	0	0	LINKED	0 TrnDis	CMD QUE	Soft Reset
8 : 15	Vendor Identification							
16 : 31	Product Identification							
32 : 35	Product Revision Level							
36 : 43	Drive Serial Number							
44 : 55	Unused Vendor-Specific Area (00h)							
56 : 95	Reserved (00h)							
96 : 143	Copyright Notice							

Abbildung 3.10: Standard Inquiry Antwort [10]

Tabelle 3.2: Gerätetypen

Hex Wert	Gerätetype
00	Direct Access Block Device (z. B. Festplatte)
01	Sequential Access Device (z. B. Magnetband)
02	Drucker
03	Prozessor
04	Write Once Device
05	CD / DVD Laufwerk
06	Scanner
07	Optical Memory
08	Media Change Device
09	Kommunikationsgerät
7F	Fehler, kein Gerät verfügbar

Tabelle 3.3: Beispiel Standard Inquiry

Anfrage	Antwort	Antwort
BCB55656	BCB55656	BCB55656
060000E8	010000EF	070000EF
000000EF	000000E8	000000E8
08290000	08880000	08990000
XX000000	00000000	FF000000
YYYYFFFF	YYYYFFFF	YYYYFFFF
00000000	00000000	00000000
00000000	00000302	00000000
00000000	1F005002	00000000
00000002	4D61726B	00000000
12000000	75732020	00000000
24000000	46504741	00000000
00000000	2D466573	00000008
00000000	74706C61	00000000
00000024	74746520	00000000
ZZZZZZZZ	312E3030	ZZZZZZZZ
BC957575	ZZZZZZZZ	BC957575
	BC957575	

Bei gesetzten EVPD Bit im Inquiry Kommando ist die Antwort vom gesendeten Page Code abhängig. Da es eine Vielzahl von Page Codes gibt, soll hier nur der Page Code 00h betrachtet werden (Abbildung 3.11). Wird dieser Code gesendet, verlangt der Initiator vom Empfänger eine Liste mit Page Codes, die vom Gerät unterstützt werden. Die Codes in dieser Liste müssen aufsteigend sortiert sein und der Page Code 00h muss immer enthalten sein. In Byte 0 muss, wie in der Standard Inquriy Antwort, der Gerätetyp angegeben sein. Das Page Length Feld beinhaltet die Anzahl der Page Codes in der Liste. In dem Beispiel aus Tabelle 3.4 werden neun Page Codes gesendet. Da der Sender mehr Daten erwartet (256 Bytes) als tatsächlich übertragen werden (13 Bytes), muss dies im FCP RSP Rahmen kenntlich gemacht werden.

Bit / Byte	7	6	5	4	3	2	1	0
0	Peripheral Qualifier			Peripheral Device Type				
1	Page Code (00h)							
2	0	0	0	0	0	0	0	0
3	Page Length							
4 : 8	Supported Page List							

Abbildung 3.11: Vital Product Data mit Page Code 00 [10]

Tabelle 3.4: Beispiel Vital Product Data mit Page Code 00

Anfrage
BCB55656
060000E8
000000EF
08290000
XX000000
YYYYFFFF
00000000
00000000
00000000
00000002
12010000
FF000000
00000000
00000000
000000FF
ZZZZZZZZ
BC957575

Antwort
BCB55656
010000EF
000000E8
08880003
00000000
YYYYFFFF
00000000
00000009
00038083
868788D1
D2000000
ZZZZZZZZ
BC957575

Antwort
BCB55656
070000EF
000000E8
08990000
FF000000
YYYYFFFF
00000000
00000000
00000000
00000800
000000F2
00000000
00000008
00000000
00000000
ZZZZZZZZ
BC957575

3.6.2 Test Unit Ready

Das Test Unit Ready Kommando (Kommandowort 00h) ermöglicht dem Initiator zu überprüfen, ob die angeschlossene Einheit betriebsbereit ist (Abbildung 3.12). Der Empfänger teilt seine Bereitschaft mittels eines einfachen FCP RSP Rahmen mit. Dies ist als Antwort ausreichend. Die Tabelle 3.5 zeigt ein Beispiel, indem ein positiver Status zurückgemeldet wird.

Bit Byte	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
	Reserved							
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	Control							

Abbildung 3.12: Test Unit Ready Kommando [10]

Tabelle 3.5: Beispiel Test Unit Ready

Anfrage	Antwort
BCB55656	BCB55656
060000E8	070000EF
000000EF	000000E8
08290000	08990000
XX000000	FF000000
YYYYFFFF	YYYYFFFF
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000008
00000000	00000000
00000000	00000000
ZZZZZZZZ	ZZZZZZZZ
BC957575	BC957575

3.6.3 Report LUN

Mit dem Report LUN Kommando (Kommandowort A0h) hat der Initiator die Möglichkeit einen Bericht über alle angeschlossenen logischen Einheiten anzufordern (Abbildung 3.13). Als logische Einheit kann eine Festplatte betrachtet werden, die an einem RAID mit weiteren Festplatten angeschlossen ist. Es kann auch eine Partition einer Festplatte als logische Einheit beschrieben werden.

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Command Code = A0h							
1-5	Reserved							
6-9	(MSB) Allocation Length (LSB)							
10	Reserved							
11	VU = 0		Reserved = 0				FLAG	LINK

Abbildung 3.13: Report LUN Kommando [11]

Byte	Bit							
	7	6	5	4	3	2	1	0
0-3	(MSB) LUN List Length = 8 (LSB)							
4-7	Reserved							
8-15	(MSB) LUN = 0 (LSB)							

Abbildung 3.14: Report LUN Antwort [11]

Die Antwort (Abbildung 3.14) auf das Kommando enthält eine einfach Liste der logischen Nummern. Die Länge der kompletten Liste muss ebenfalls mitgeteilt werden und berechnet sich aus der maximalen Bytelänge der gesendeten Nutzdaten abzüglich den ersten acht Bytes. Soll, wie in Tabelle 3.6, nur eine LUN verwendet werden, ist für die Länge der Liste 8h einzusetzen. Die verwendete LUN hat in diesem Beispiel die Nummer Null.

Tabelle 3.6: Beispiel Report LUN

Anfrage
BCB55656
060000E8
000000EF
08290000
XX000000
YYYYFFFF
00000000
00000000
00000000
00000002
A0000000
00000000
08000000
00000000
00000800
ZZZZZZZZ
BC957575

Antwort
BCB55656
010000EF
000000E8
08880000
00000000
YYYYFFFF
00000000
00000008
00000000
00000000
00000000
ZZZZZZZZ
BC957575

Antwort
BCB55656
070000EF
000000E8
08990000
FF000000
YYYYFFFF
00000000
00000000
00000000
00000800
000007F0
00000000
00000008
00000000
00000000
ZZZZZZZZ
BC957575

3.6.4 Read Capacity

Das Read Capacity Kommando (Kommandowort 25h) fordert den Empfänger auf, Auskunft über seine Speicherkapazität zu liefern (Abbildung 3.15). Im Gegensatz zu den bisherigen Kommandos gehört dieses zu dem Befehlssatz, die nur bei Direct Access Devices (z.B. Festplatten) verwendet werden können. In dieser Weise gibt es für jeden Gerätetyp eigene Befehlssätze und es gibt Kommandos, die bei allen Geräten vorhanden sein müssen, wie zum Beispiel das Inquiry Kommando.

Bit Byte	7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0	1
1	0	0 Reserved	0	0	0	0	0	RelAdr
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	PMI
9	Control							

Abbildung 3.15: Read Capacity Kommando [10]

Die Abbildung 3.16 zeigt das Format der Antwort. Der Initiator verlangt in den ersten vier Bytes die maximale logische Blockadresse der Festplatte und in den letzten vier Bytes die Länge eines Blockes. Aus diesen beiden Angaben kann die Speicherkapazität berechnet werden. Dazu wird die maximale Blockadresse mit der Blocklänge multipliziert. In dem Beispiel aus Tabelle 3.7 ist die maximale logische Blockadresse 20 00 00h, was dezimal der Zahl 2.097.152 entspricht. Die Blocklänge beträgt 512 Byte, was dem Wert 200h entspricht. Die beiden Zahlen miteinander multipliziert ergeben einen Speicherkapazität von 1 GByte (1.073.741.824 Byte).

Byte	Description
0	Logical Block Address (MSB)
1	Logical Block Address
2	Logical Block Address
3	Logical Block Address (LSB)
4	Block Length (MSB)
5	Block Length
6	Block Length
7	Block Length (LSB)

Abbildung 3.16: Read Capacity Antwort [10]

Tabelle 3.7: Beispiel Read Capacity

Anfrage
BCB55656
060000E8
000000EF
08290000
XX000000
YYYYFFFF
00000000
00000000
00000000
00000002
25000000
00000000
00000000
00000000
00000008
ZZZZZZZZ
BC957575

Antwort
BCB55656
010000EF
000000E8
08880000
00000000
YYYYFFFF
00000000
00200000
00000200
ZZZZZZZZ
BC957575

Antwort
BCB55656
070000EF
000000E8
08990000
FF000000
YYYYFFFF
00000000
00000000
00000000
00000000
00000000
00000008
00000000
00000000
ZZZZZZZZ
BC957575

4 Optische Schnittstelle

4.1 Hardware [7]

Die optische Schnittstelle soll auf einem Entwicklungsboard von Xilinx integriert werden. Das Board muss einige Mindestvoraussetzungen erfüllen, damit der Fibre Channel Core verwendet werden kann und muss mindestens einen SFP Steckplatz (Small Form-factor Pluggable) besitzen.

Das ML 555 (Abbildung 4.1) erfüllt alle Voraussetzungen und kann darüber hinaus vom Fraunhofer Institut für spätere PCI Express Anwendungen weiter genutzt werden. Das Evaluation Board verwendet als FPGA einen Virtex 5 LTX mit der Bezeichnung XC5VLX50T-1FF36 und bietet auch noch weitere Anschlüsse für die unterschiedlichsten Anwendungsgebiete. Die komplette Funktionalität ist im Blockschaltbild (Abbildung 4.2) dargestellt.

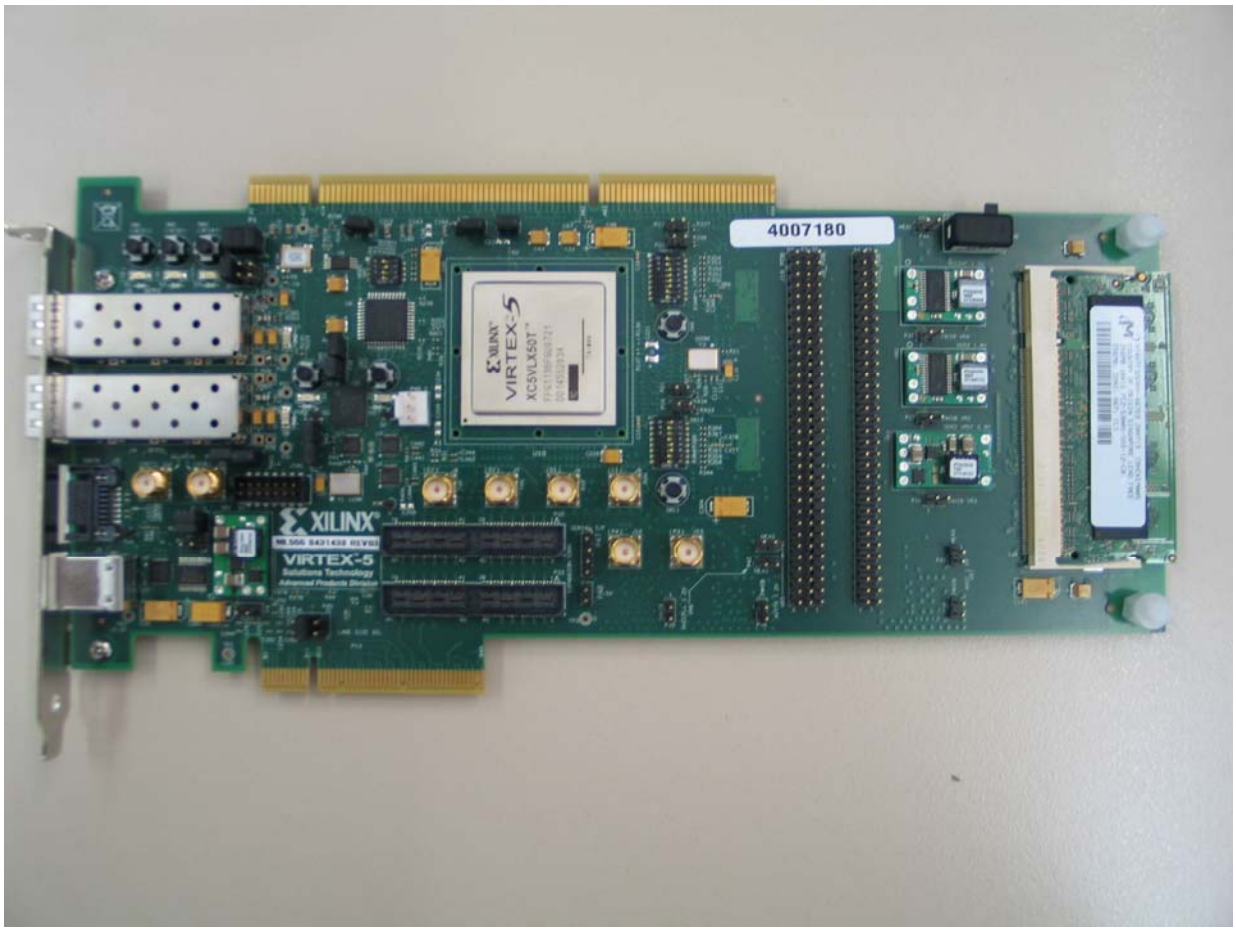

Abbildung 4.1: Evaluation Board ML 555 von Xilinx

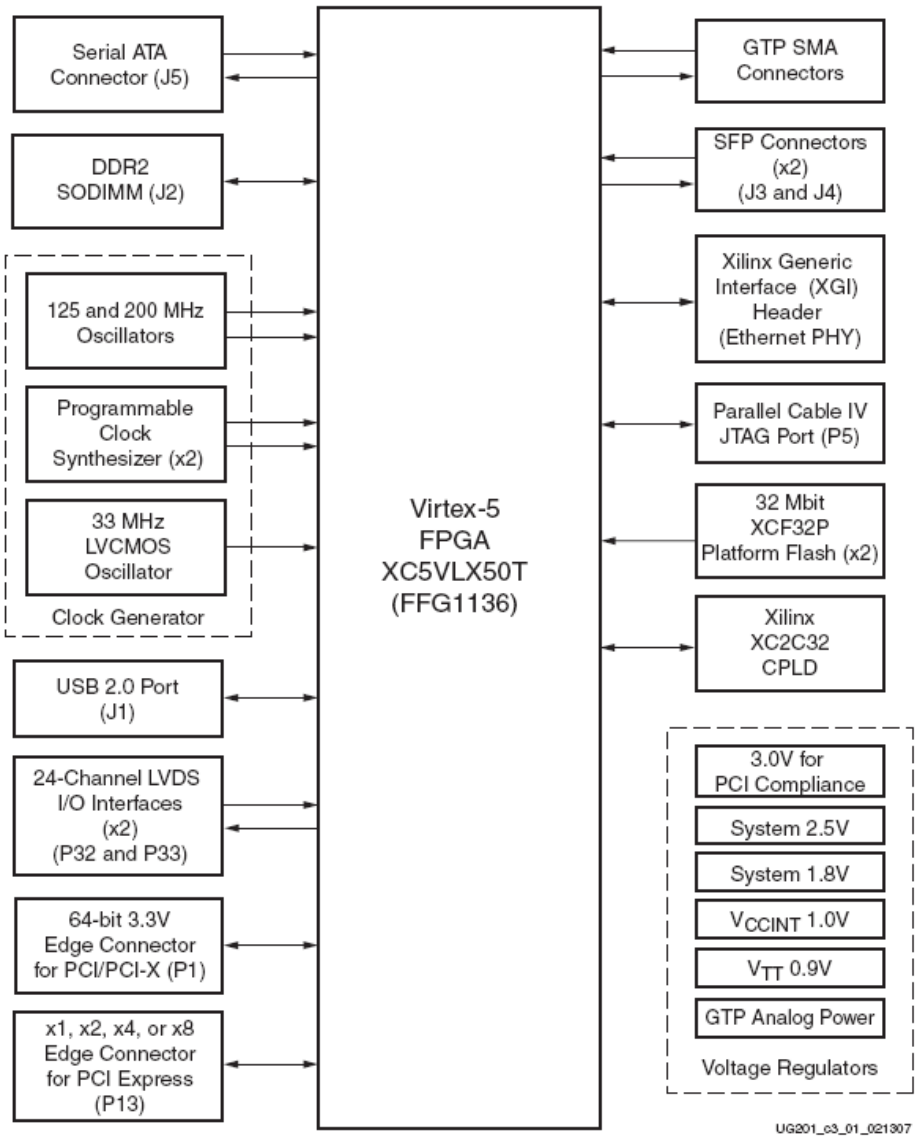

Abbildung 4.2: Blockschaltbild des ML 555 [7]

Das ML 555 muss entweder auf einen PCI oder einen PCI Express Steckplatz installiert werden, da es die Versorgungsspannung über den Steckplatz bezieht. Auf Grund der unterschiedlichen Spannungspegel dieser beiden Busstandards ist es sehr wichtig die Karte vor dem Einbau richtig zu konfigurieren. In der Tabelle 4.1 ist die Konfiguration für einen Einbau in einen PCI Express Steckplatz dargestellt. Die dafür benötigten Schalter sind auf der Abbildung 4.3 kenntlich gemacht.

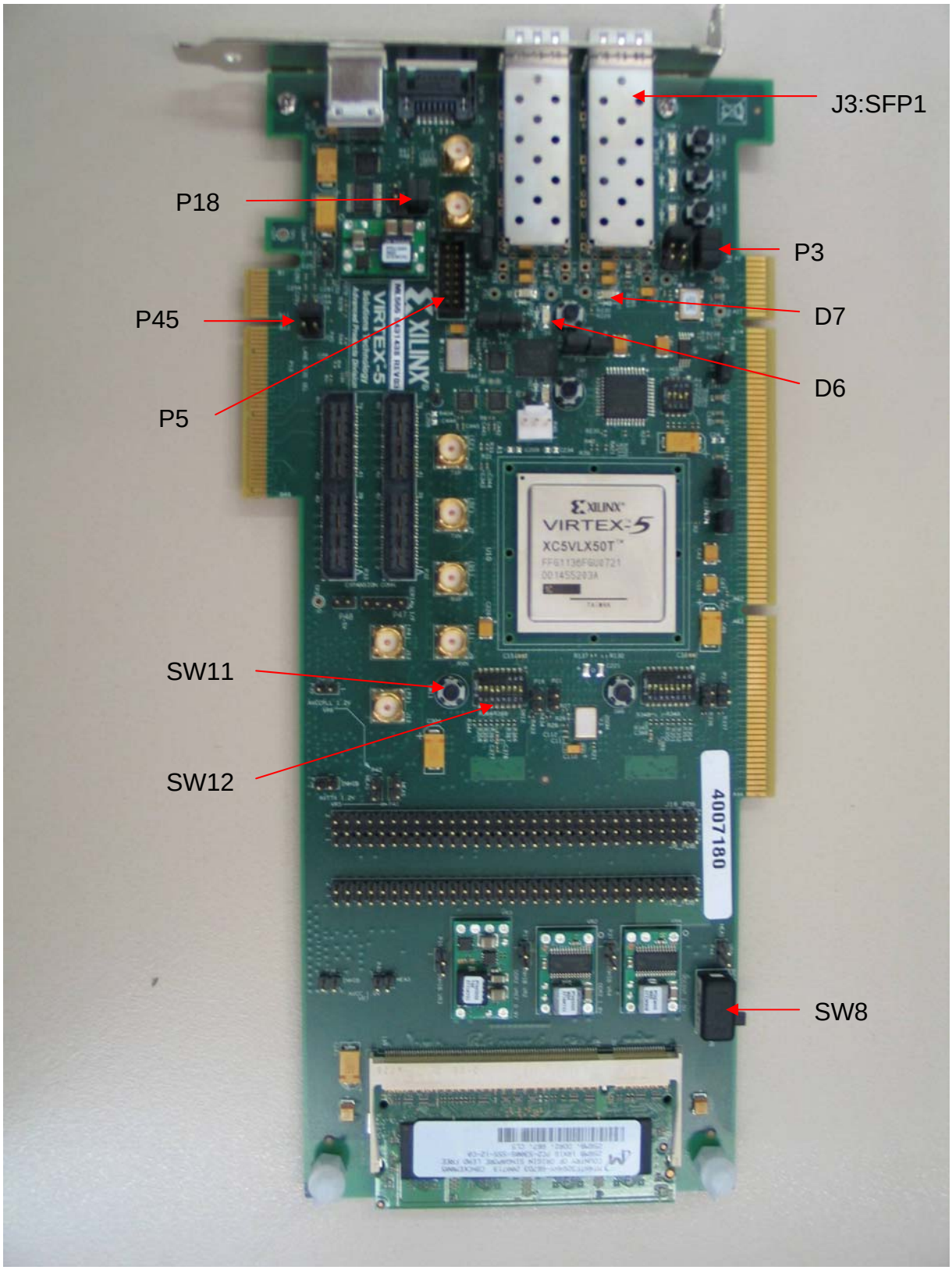

Abbildung 4.3: Positionen der benutzten Schalter, Taster und Jumper

Tabelle 4.1: PCI Express Konfiguration

Bezeichnung			**Bedeutung**
SW8 PCIe	-	-	Wählt die Spannungsquelle für das FPGA
P3-1 - P3.2 ON	P3.3 - P3.4 ON	P3.5 - P3.6 ON	4-Lane Endpoint Design (Default)
P18.1 - P18.2 ON	P18.3 - P18.4 ON	-	Aktiviert 12V zu 5V Wandlung für PCIe
P45.1 - P45.2 OFF	P45.3 - P45.4 ON	P45.5 - P45.6 OFF	4 Aktive PCIe Lanes (Default)

Für das spätere VHDL Design muss der Synthesizer auf dem Board so eingestellt werden, dass die Taktfrequenz 212,5 MHz beträgt. Dafür ist es notwendig, den Schalter SW12 nach der Tabelle 4.2 einzustellen. Damit die Einstellung vom Synthesizer übernommen wird, muss nach dem Einschalten der Spannungsversorgung die Taste SW11 gedrückt werden. Beide Schalter sind in Abbildung 4.3 eingezeichnet.

Tabelle 4.2: Schalterstellung von SW12

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ON	OFF	OFF	OFF	ON	ON	OFF	OFF

Bevor jedoch der Computer eingeschaltet wird und somit die Karte mit Spannung versorgt ist, muss ein passendes SFP Modul auf Steckplatz J3:SFP1 montiert werden. Das SFP Modul (Abbildung 4.4) ist ein standardisierter modularer optischer Transceiver, der die Umwandlung elektrisch in optisch übernimmt. Die SFPs sind in unterschiedlichsten Ausführungen erhältlich.
Von daher ist es notwendig darauf zu achten, dass ein Modul verwendet wird, das zu den optischen Daten auf der Gegenseite passt. Die Wahl ist auf ein SFP Modul (Bezeichnung TXN31115D2) von Intel gefallen. Es arbeitet mit einer Wellenlänge von 850 nm und einer Übertragungsgeschwindigkeit von bis zu 4 GBit/s. Als Leitung wird ein gedrehtes Mulitmode Kabel mit LC-Steckern verwendet.

Abbildung 4.4: SFP Modul

Auf eine Besonderheit muss jedoch geachtet werden. Der Widerstand R394 (4,7k) auf dem Evaluation Board soll den Pin 2 vom SFP auf Low Pegel ziehen und somit den Sender des SFPs aktivieren. Da sich jedoch ebenfalls ein interner Widerstand (ca. 5k) im SFP befindet, der an 3,3V anliegt, hat der Pin 2 eine Spannung von ca. 1.7 V und befindet sich in einem undefinierten Zustand. Daher muss der Widerstand R394 gebrückt werden. Das lässt sich am einfachsten durch eine Brücke zwischen P25.1 und P26.2 realisieren. Im Anhang A1 befindet sich der dazugehörige Schaltplan mit der eingezeichneten Brücke.
Letztendlich wird noch das Programmierkabel über einen USB Anschluss am PC auf den Programmierstecker P5 am ML 555 gesteckt (Abbildung 4.3).
Sind nun alle Komponenten installiert und richtig konfiguriert, leuchten nach dem Einschalten des Computers die LED D6 und D7 (Abbildung 4.3) auf dem Evaluation Board auf. Die LED D6 signalisiert ein erfolgreiches Initialisieren des FPGAs und die LED D7 leuchtet auf, wenn ein Kabel angeschlossen ist und erste optische Signale empfangen werden.

4.2 Rocket IO [8]

4.2.1 Allgemeiner Aufbau

Die Rocket IO GTP Transceiver sind leistungseffiziente Sender und Empfänger, die speziell für Virtex 5 FPGAs von Xilinx entwickelt wurden. Sie sind bereits in der FPGA Architektur integriert und lassen sich beliebig konfigurieren. Dadurch wird dem Anwender eine Vielzahl von Anwendungsmöglichkeiten zur Verfügung gestellt. Die GTPs verwenden serielle CML (Current Mode Logic) Treiber mit denen sich die Terminierung, der Spannungspegel und die Entkopplung frei einstellen lassen. Ebenfalls lassen sich der Vorverstärker des Senders und die Signalentzerrung des Empfängers programmieren. Die Datenrate eines Transceivers kann mindestens 100 MB/s und maximal 3.75 GB/s betragen. Des Weiteren sind Besonderheiten wie 8B/10B Encoder, Comma Alignment, Channel Bonding und Takt Korrektur bereits integriert.

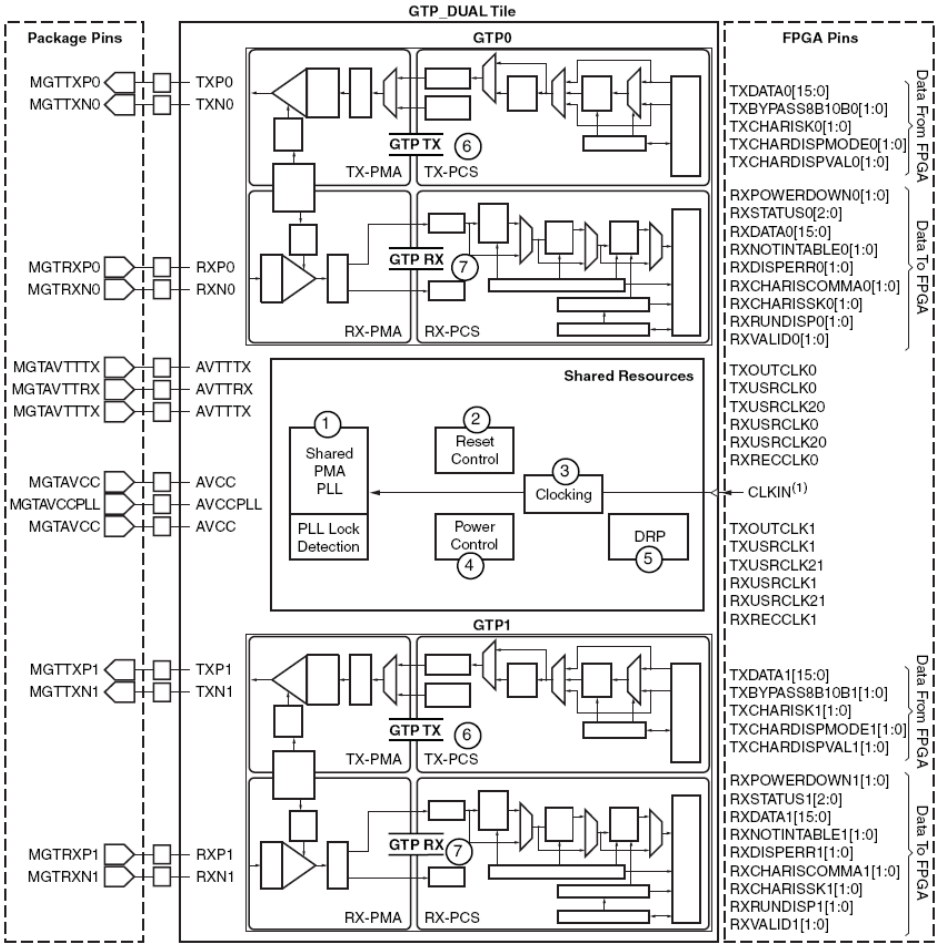

Abbildung 4.5: GTP Transceiver Paar mit PLL [8]

In einem Virtex 5 werden zwei Transceiver immer als Pärchen (DUAL_GTP Tiles) platziert. Diese Anordnung hat den Vorteil, dass sich zwei Transceiver eine PLL teilen können und somit Platz und Leistung gespart wird. In Abbildung 4.5 ist ein solches Paar mit einer gemeinsamen PLL dargestellt. Auf der linken Seite befinden sich die Anschlüsse des FPGAs, diese sind bereits den entsprechenden Pins am FPGA zugewiesen. Auf der rechten Seite befinden sich die Ein- und Ausgänge, die sich im FPGA befinden und programmiert werden können.

Der GTP Transceiver kann über den Xilinx Core Generator konfiguriert und in den VHDL-Code integriert werden. Dazu wird die Option Rocket IO im Xilinx Core Generator ausgewählt. Danach wird ein Wizard (Abbildung 4.6) gestartet, der dem Anwender die Konfiguration der Rocket IOs in 12 Schritten erleichtern soll. Darin kann zum Beispiel ausgewählt werden, welche Rocket IOs im FPGA verwendet werden und mit welcher Übertragungsrate diese arbeiten sollen. Auch optionale Ein- und Ausgänge können freigeschaltet werden und die Datenbusbreite des Senders und Empfängers bestimmt werden.

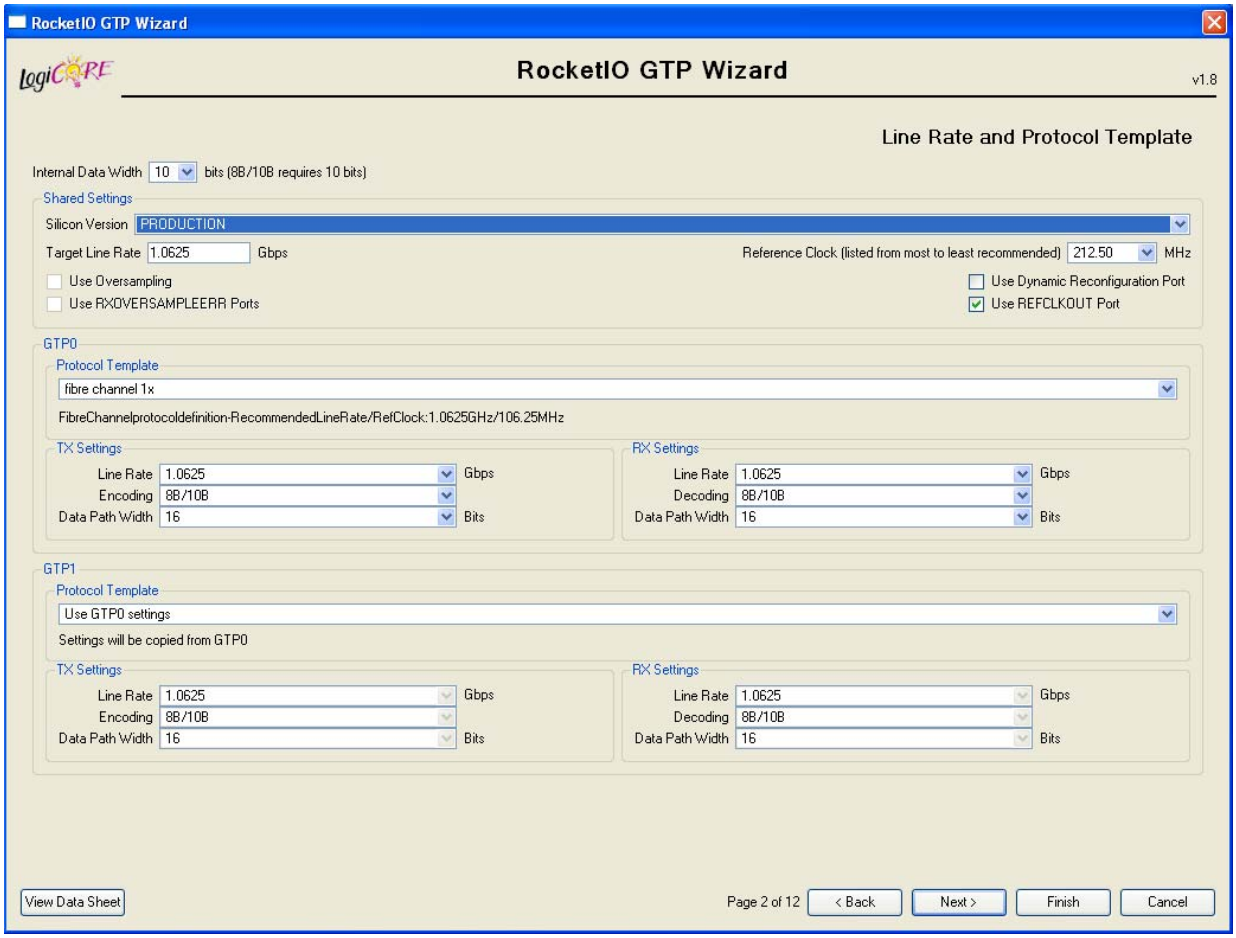

Abbildung 4.6: Der Rocket IO Wizard

Der Wizard bietet dabei bereits einige Vorlagen zu den gängigsten Protokollen an, was das Einstellen zu Beginn erleichtert. Jedoch müssen alle Einstellmöglichkeiten auf das SFP Modul und den Fibre Channel Core überprüft und abgestimmt werden.
Die nun vom Core Generator erzeugte Vorlage wird über eine einfache Komponentendeklaration in das VHDL Programm eingebunden und mittels Signalzuweisungen mit weiteren Komponenten (z.B. Fibre Channel Core) verbunden.

4.2.2 Takteinstellungen

In Abbildung 4.7 ist die Taktverteilung zu den einzelnen GTP Transceiver Paaren zu erkennen. An dem Paar X0Y4 sind die beiden SFP Steckplätze angeschlossen und es wird ein Takt von 212,5 MHz benötigt. Anhand der Abbildung ist nun zu erkennen, dass auf dem ML 555 nur ein Takt von 125 MHz direkt zu dem entsprechenden GTP gelangt. Die einstellbaren Takte befinden sich an X0Y0, X0Y1 und X0Y5. Auf Grund einer weiteren Eigenschaft der GTPs ist es jedoch möglich den richtigen Takt zu bekommen. Ein GTP Pärchen kann den Takt zu einem benachbarten Paar weiterleiten. Somit ist es möglich, den Takt von X0Y5 auf X0Y4 durchzuschleifen. Dies hat jedoch den Nachteil, dass mit X0Y5 zwei weitere Transceiver verwendet werden müssen, um nur einen Takt durchzureichen. Somit werden von vier GTPs (zwei Paare) nur einer wirklich verwendet, was natürlich Ressourcen im FPGA verschwendet. Um zumindest den Leistungsverbrauch und damit die Temperatur niedrig zu halten, können die überflüssigen GTPs über so genannte Powerdown Anschlüsse abgeschaltet werden. Diese Anschlüsse müssen jedoch vorher im Rockt IO Wizard frei geschaltet werden.

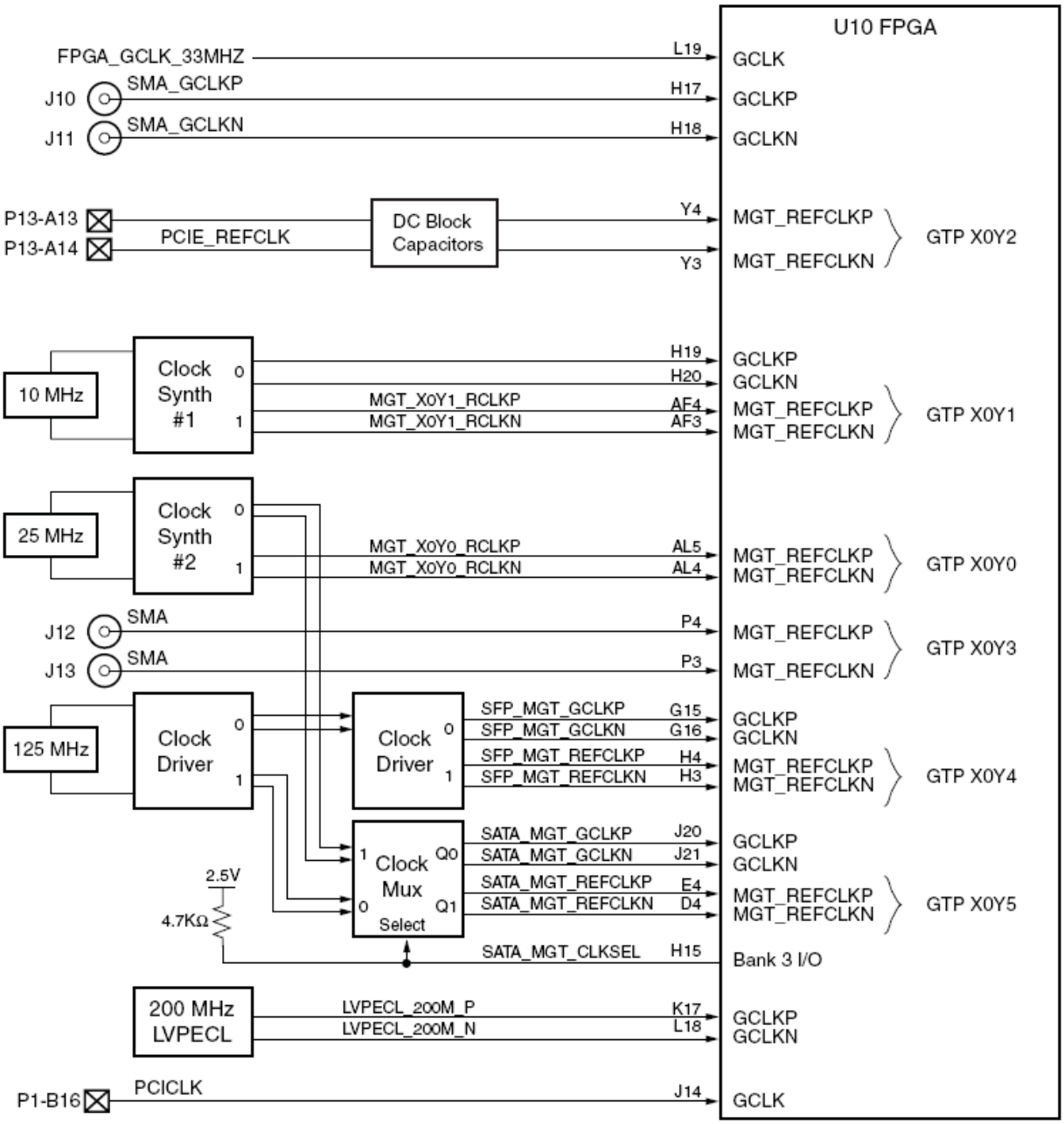

Abbildung 4.7: Clock Aufteilung auf dem Entwicklungsboard [7]

In der PLL (Abbildung 4.8) muss nun für jeden Sender und Empfänger ein Takt von 1,0625 GHz erzeugt werden. Um das zu erreichen müssen die entsprechende Parameter der PLL so gesetzt werden, so dass aus 212,5 MHz eine Frequenz von 1,0625 GHz erzielt wird. Nach Gleichung 4.1 wird der interne PLL Takt berechnet. Aus den Gleichungen 4.2 und 4.3 wird die RX- und TX-Übertragungsrate berechnet. Mit den Werten aus der Tabelle 4.3 wird somit ein Takt von 1,0625 GHz erreicht. Es gibt darüber hinaus auch noch weitere Kombinationen, die ebenfalls einen Takt von 1,0625 GHz bewirken würden.

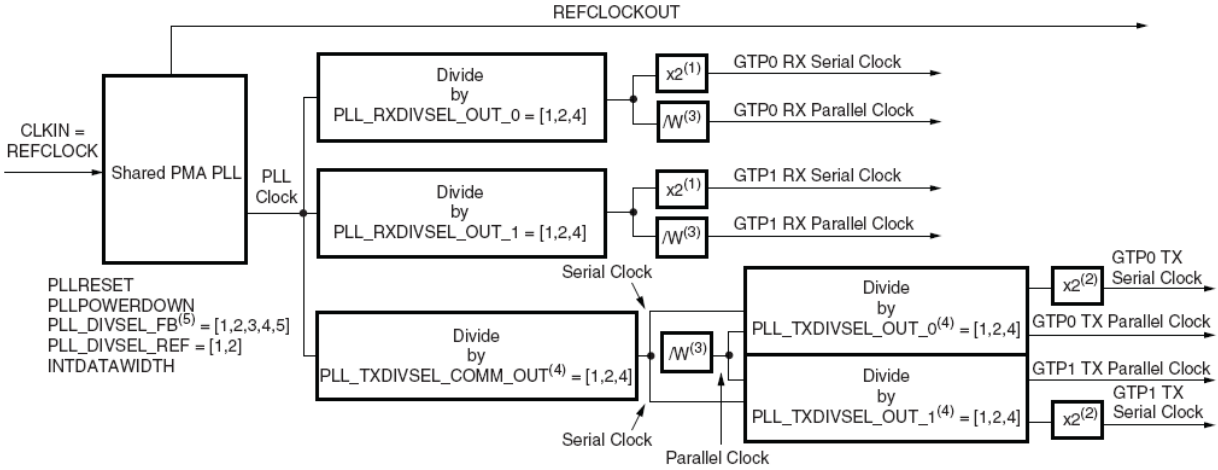

Abbildung 4.8: Gemeinsame PLL zweier GTPs [8]

$$PLLClock = CLKIN \times \frac{PLL\_DIVSEL\_FB \times DIV}{PLL\_DIVSEL\_REF} \qquad (4.1)\ [8]$$

$$TXClock = \frac{2 \times PLLClock}{PLL\_TXDIVSEL\_OUT} \qquad (4.2)\ [8]$$

$$RXClock = \frac{2 \times PLLClock}{PLL\_RXDIVSEL\_OUT} \qquad (4.3)\ [8]$$

Tabelle 4.3: Verwendete Parameter

Parametername	Wert
PLL_DIVSEL_FB	2
DIV	5
PLL_DIVSEL_REF	1
PLL_TXDIVSEL_OUT_0/1	4
PLL_RXDIVSEL_OUT_0/1	4
PLL_TXDIVSEL_COMM_OUT	1

4.2.3 GTP Sender

Jeder GTP Transceiver besitzt einen unabhängigen Sender (Abbildung 4.9), der wiederum in PMA (Physical Media Attachment) und PCS (Physical Coding Sublayer) unterteilt ist. Die Daten gelangen über einen 16 Bit breiten Datenbus (Punkt 1) in den Sender. Der Datentransfer über diesen Bus wird vom Fibre Channel Core gesteuert. Im nächsten Schritt werden die Daten von 8 Bit in 10 Bit kodiert (Punkt 2). Dazu wird der 16 Bit breite Datenbus in zweimal 8 Bit geteilt und in zweimal 10 Bit umgewandelt. Nun kann die Phase kontrolliert und gegebenenfalls korrigiert (Punkt 3) werden oder die Daten gelangen direkt über einen Bypass zum Polaritätskontroller (Punkt 4). Mit dem Pseudo-Zufallszahlengenerator (Punkt 5) wird die Qualität der Verbindung überprüft. Nun werden die kodierten Daten von einem 10 Bit breiten parallelen Signal in ein serielles Signal umgewandelt (Punkt 6) und gelangen über einen Vorverstärker zum Treiber (Punkt 7). Alle Automatismen, die benötigt werden damit die Daten zu dem richtigen Zeitpunkt bereit stehen, werden von dem Sender übernommen.

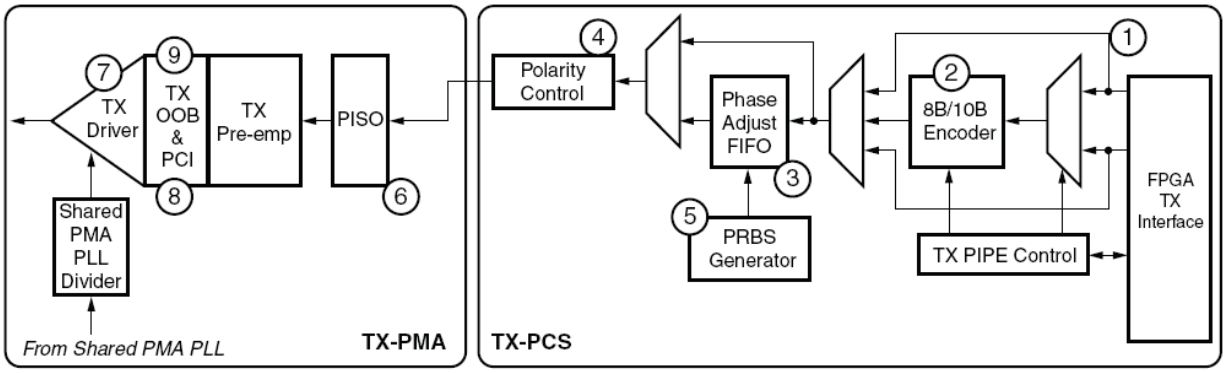

Abbildung 4.9: Blockdiagramm eines Senders [8]

4.2.4 GTP Empfänger

Der Empfänger eines GTP Transceivers (Abbildung 4.10) ist ebenfalls in PMA und PCS aufgeteilt. Als erstes wird das empfangene Signal aufbereitet (Punkt 1) bevor nun die Taktinformationen aus dem Datensignal wiederhergestellt werden (Punkt 3). Dann wird das serielle Signal wieder in ein 10 Bit breites paralleles Signal umgewandelt (Punkt 4). Im Anschluss werden die Daten durch verschiedene Kontrollmechanismen überprüft und wieder von 10 Bit in 8 Bit dekodiert (Punkt 10). Als letztes werden wieder zwei 8 Bit in 16 Bit zusammengefasst. Nun gelangen die empfangenen Daten über ein 16 Bit breiten Datenbus (Punkt 14) zum FPGA und somit zu dem Fibre Channel Core.

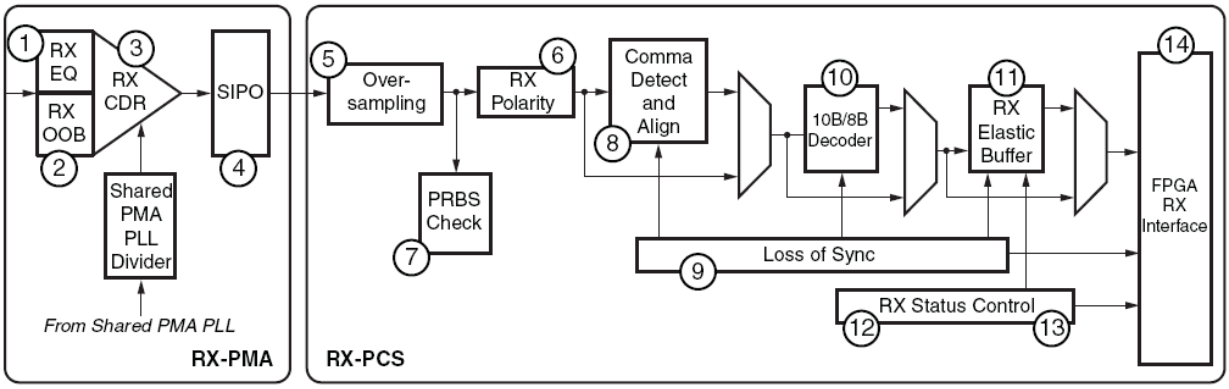

Abbildung 4.10: Blockdiagramm eines Empfängers [8]

4.3 Fibre Channel Core [9]

Der Fibre Channel Core von Xilinx ist flexibel einstellbar und bietet eine Vielzahl von nützlichen Funktionen. Er lässt sich als Single oder Dual Speed Core konfigurieren und unterstützt die Geschwindigkeiten 1 GB/s, 2 GB/s und 4GB/s. Es werden die Serviceklassen 1, 2, 3 und 4 sowie die Standards FC-PH und FC-FS unterstützt. Er übernimmt die Aufgaben der Protokollschichten FC-0, FC-1 und teilweise FC-2 (Abbildung 4.11).

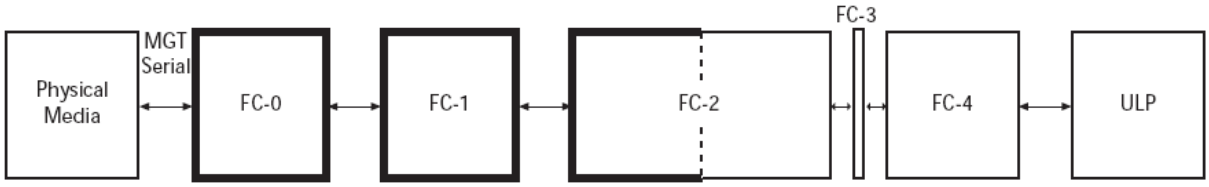

Abbildung 4.11: Fibre Channel System Level Blockdiagramm [9]

In Abbildung 4.12 ist die Architektur eines Single Speed Fibre Channel Cores gezeigt. Der Datenaustausch findet über ein 32 Bit breites Clientinterface statt. Ein optionales Management Interface bietet dem Benutzer einen einfachen Zugriff auf Konfigurationen, Statistikinformationen und Einstellmöglichkeiten. Das Credit Management ist für die Buffer-to-Buffer Flusssteuerung zuständig. Es werden die SOF und R_RDY Signale gezählt bzw. ausgewertet und entschieden, ob weiter Daten empfangen bzw. gesendet werden können. Der MAC Block beinhaltet die Port Zustandsmaschine und ist für die Link Initialisierung zuständig. Der Link Kontroller ist für die Synchronisierung der empfangenen Daten im Zusammenspiel mit den GTPs verantwortlich. Für die zu sendenden Daten wird ein CRC Check durchgeführt und der Wert im CRC Feld eines Fibre Channel Frames eingetragen.

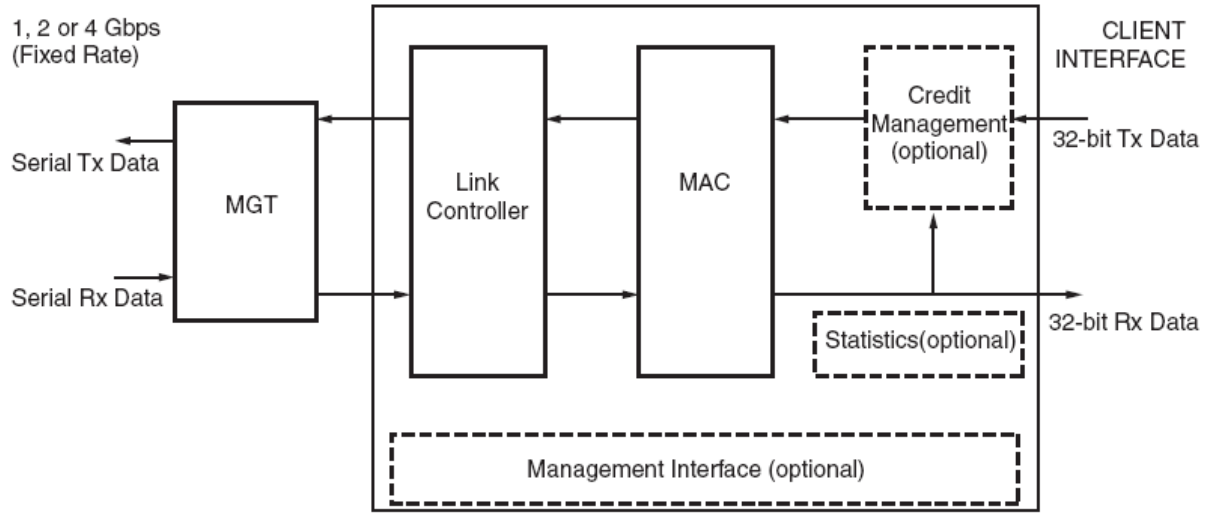

Abbildung 4.12: Single-Speed Architektur [9]

In dieser Diplomarbeit soll ein Singel Speed Core mit der Geschwindigkeit von 1 GB/s verwendet werden. Die optionalen Funktionen Credit Management und Management Interface sollen ebenfalls Verwendung finden. Der aus diesen Vorgaben resultierende Core mit seinen Anschlüssen ist in Abbildung 4.13 ersichtlich.

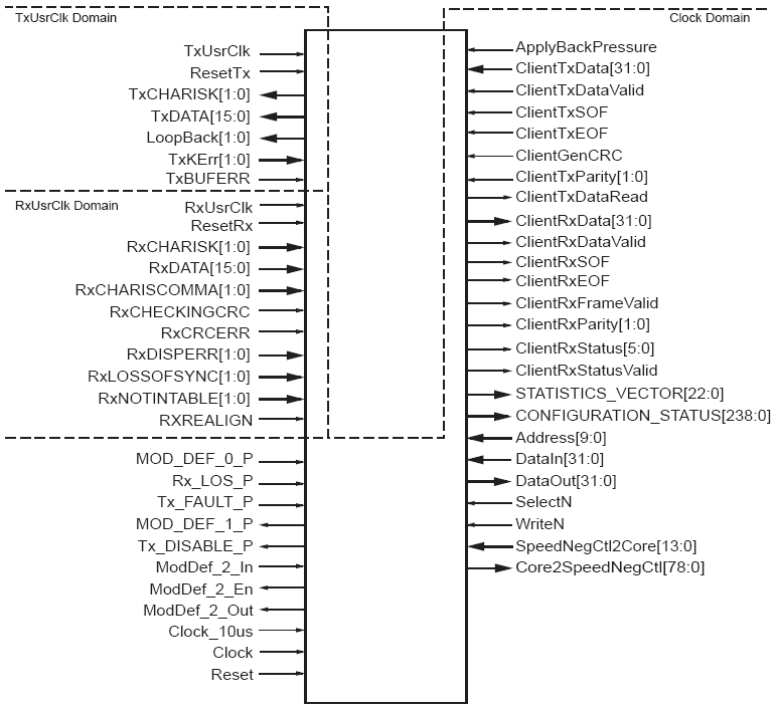

Abbildung 4.13: Übersicht der Anschlüsse des Cores [9]

Um den Fibre Channel Core benutzen zu können wird eine Entwicklungslizenz für den Core benötig. Dies ist, im Gegensatz der der Vollversion, kostenlos möglich und muss bei Xilinx beantragt werden. Diese Entwicklungslizenz ist dann auf drei Monate befristet und der Fibre Channel Core schaltet sich nach 8 Stunden Dauerbetrieb im FPGA ab.

Nach der Lizenzierung kann die Option Fibre Channel Core im Xilinx Core Generator ausgewählt werden. Nun startet ein Wizard, mit dessen Hilfe der Fibre Channel Core konfiguriert werden kann. Jedoch sind die Einstellmöglichkeiten nicht so umfangreich wie bei den Rocket IOs. Es müssen nur Angaben über die Geschwindigkeit, der Anzahl der Fibre Channel Ports und die gewünschte Zusatzoptionen gemacht werden. Aus diesen Informationen erzeugt der Core Generator eine Netzliste. Diese Netzliste wird nun mittels einer Komponentendeklaration in den VHDL Code integriert. Somit kann der Anwender einfach mittels Signalzuweisungen eine Verbindung zu anderen Komponenten (z.B. GTP Transceiver) herstellen.

Die Ein- und Ausgänge des Fibre Channel Cores lassen sich in verschiedene Gruppen unterteilen. Als erstes sind die Signale, die zum GTP Transceiver führen (Abbildung 4.13, linke Seite oben und Mitte) zu nennen. Dabei ist darauf zu achten, dass nicht alle Signale Eins zu Eins an den GTP Transmitter angeschlossen werden können. Einige Signalen müssen, in Abhängigkeit des verwendeten FPGAs, modifiziert werden. Damit ist gemeint, dass zum Beispiel einige zwei Bit Signale gedreht werden müssen oder ein Eingang wird nicht unterstützt und muss daher auf einen festen Pegel gesetzt werden.

Als nächstes sind die Signale zum SFP (Abbildung 4.13, linke Seite unten) zu nennen. Diese haben nur die Aufgabe das SFP zu aktivieren bzw. deren Status über eine serielle 2-Draht Verbindung abzufragen. Diese Leitungen haben keine Bedeutung im Zusammenhang der Datenübertragung, da die Daten über die GTPs zum SFP gelangen. Da einige Signale bei dem ML 555 vom SFP nicht an das FPGA geführt sind, sondern über einen Pull Up Widerstand bereits einen festen Pegel besitzen, müssen einige Eingänge des Cores ebenfalls auf diesen Pegel gesetzt werden. Andernfalls wird vom Core angenommen, dass der SFP nicht bereit ist.

Auf der rechten Seite des Blockschaltbildes finden sich die Anschlüsse des Management Interface und des Clientinterface. Das Management Interface wird im folgenden Abschnitt näher betrachtet und das Clientinterface wird in Kapitel 5 beschrieben.

Die Abbildung 4.14 zeigt die Vorschrift für die Taktverteilung im FPGA. Die GTPs werden mit einer Frequenz von 212,5 MHz getaktet. Von dort gelangt der Takt zu einer DCM (Digital Clock Manager), die der Anwender noch zusätzlich integrieren muss. Mit dem Entwicklungswerkzeug ISE 10.1 von Xilinx wird dem Anwender die Möglichkeit gegeben, fertige Komponenten wie RAM oder DCM über Makros zu integrieren. In der Abbildung 4.15 ist das verwendete Makro für den benötigte DCM dargestellt. Die Einstellungen werden über die Generic Anweisung vorgenommen. Der DCM muss nun einen Takt von 106,25 MHz und von 53,125 MHz erzeugen. Der erste Takt wird wieder zu den GTPs geführt und der zweite Takt wird für den Fibre Channel Core benötigt. Mit diesem Takt wird auch später das Management und Client Interface betrieben.

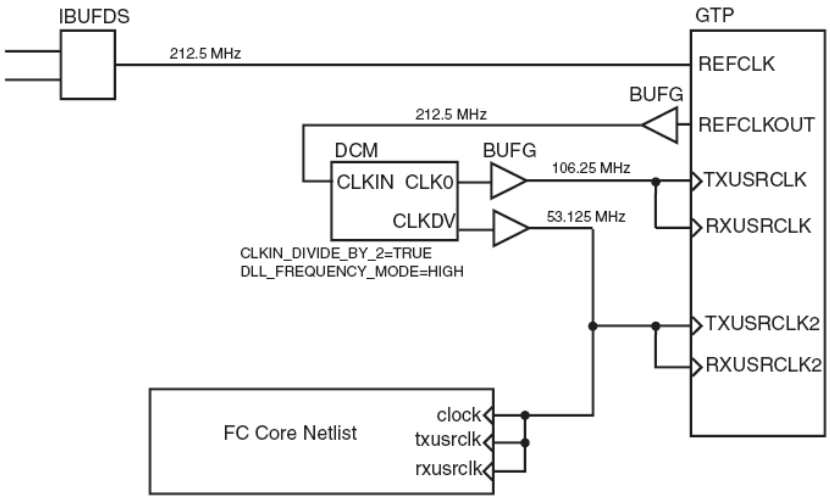

Abbildung 4.14: Taktverteilung im FPGA [9]

```
103
104     dcm1 : dcm_base
105        generic map (clkin_period => 9.412,
106                     clkin_divide_by_2 => true)
107        port map (
108           clkin            => refclkout_bufg,
109           clkfb            => txusrclk_int,
110           rst              => dcm_reset,
111           clk0             => clk0,
112           clk90            => open,
113           clk180           => open,
114           clk270           => open,
115           clk2x            => open,
116           clk2x180         => open,
117           clkdv            => clkdv0,
118           clkfx            => open,
119           clkfx180         => open,
120           locked           => dcm_locked
121        );
122
```

Abbildung 4.15: DCM Makro

4.4 Management Interface

Mit dem Management Interface lässt sich ein Zugriff auf die verschiedenen Register des Fibre Channel Cores einfach realisieren. In Abbildung 4.16 ist ein Zeitdiagramm verschiedener Schreib- und Lesezyklen dargestellt.

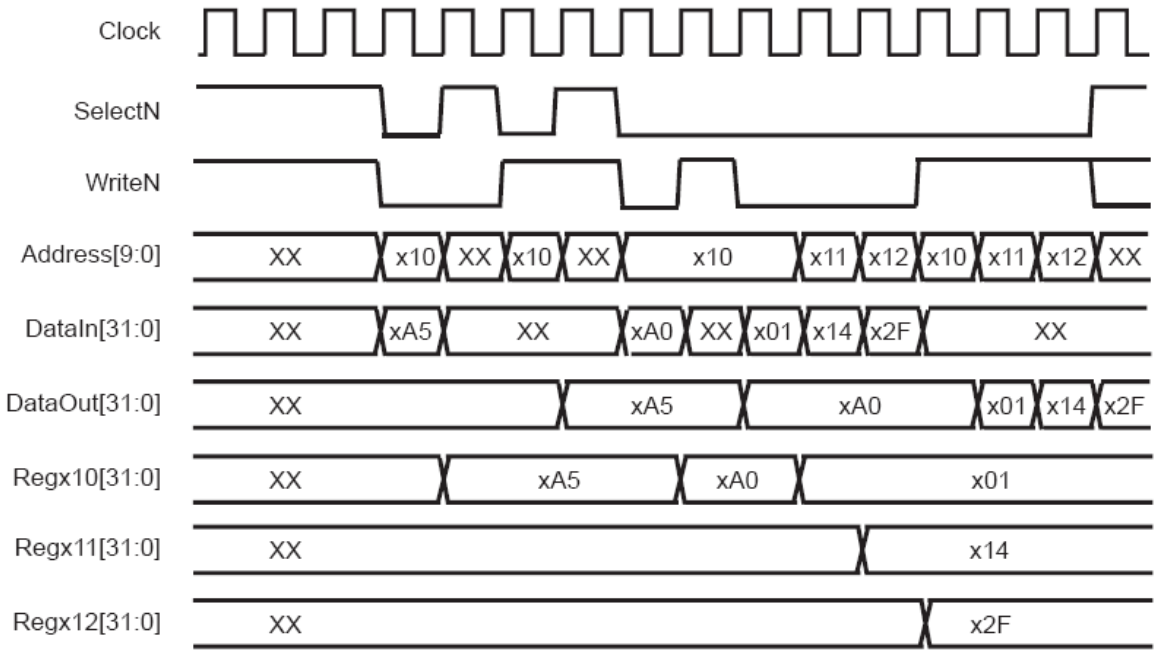

Abbildung 4.16: Zeitdiagramm des Management Interface [9]

Auf den Adressbus wird die Adresse des anzusprechenden Registers geschrieben. Die WriteN-Leitung besitzt eine Eins, wenn ein Register gelesen und eine Null, wenn das Register beschrieben werden soll. Die Daten, die in das Register geschrieben werden sollen, werden vorher auf den DataIn Bus gestellt. Die ausgelesenen Daten befinden sich nach dem Lesezyklus auf dem DataOut Bus. Das Setzen der SelectN-Leitung von Null auf Eins (flankengesteuert) bewirkt das Lesen oder Schreiben des Registers.

Die Tabelle 4.4 beinhaltet alle Register und deren Adresse, die sich nun über diese Schnittstelle auslesen bzw. setzen lassen.

Tabelle 4.4: Übersicht der Registeradressen

Adresse	Registername	Kommentar
0x000	OpticsControl	SFP Ansteuerung
0x004	OpticsStatus	SFP Status
0x008	ModDefAddress	Adresse für SFP Register
0x00C	ModDefData	Daten aus SFP Register
0x010	LinkControl	Unter anderem Loopback für GTP einstellbar
0x014	LinkStatus	Status über LOS (Lost of Synchronization)
0x020	PSMControl	Ansteuerung der Port Zustandsmaschine
0x024	PSMStatus	Status der Zustandsmaschine
0x028	HostPrimData	
0x02C	TOV	
0x030	B2BCredit	
0x034	OutStdB2BCredit	
0x038	OutStdRRDYCredit	
0x03C	BB_SC_N	
0x040	BB_SCs	
0x044	BB_SCr	

Um den Fibre Channel Core zu aktivieren muss, nachdem alle Takte erfolgreich generiert sind und der Reset durchgeführt ist, eine Startsequenz ablaufen. Diese Startsequenz muss nacheinander bestimmte Register des Fibre Channel Cores setzen.

Als erstes muss im OpticsControl Register der Sender des SFPs aktiviert werden. Dies geschieht durch Setzen des Bit 0 auf Null. Als nächstes wird die PSM gestartet. Dazu muss im PSMControl Register das Bit 0 auf Eins gesetzt werden. Nun beginnt der Fibre Channel Core die Initialisierung der Verbindung. Der Status der Initialisierung und mögliche Fehler können in dem PSMStatus Register überprüft werden. Nach einer erfolgreichen Initialisierung besitzen die Bits 16 bis 19 im PSMStatus Register den Wert 0.

Diese Prozedur wird in VHDL mit einer Zustandsmaschine (Abbildung 4.17) realisiert. Die Zustandsmaschine wird mit dem letzten Reset Impuls gestartet. Eine erfolgreiche Initialisierung wird durch das Leuchten der LED D2 (Abbildung 4.18) signalisiert. Ist jedoch ein Fehler aufgetreten wird dies durch das Leuchten der LED D1 angezeigt. Durch das Drücken der Taste SW1 kann ein manueller Reset durchgeführt werden. Durch das Betätigen der Taste SW2 lassen sich zu jeder Zeit alle Register des Fibre Channel Cores auslesen und in ChipScope anzeigen. ChipScope ist ein Entwicklungswerkzeug von Xilinx mit dessen Hilfe Signale im FPGA sichtbar gemacht werden können. Um nun die Informationen zu erhalten muss mindestens der DataIn Bus aufgezeichnet und auf die SelectN Leitung getriggert werden.

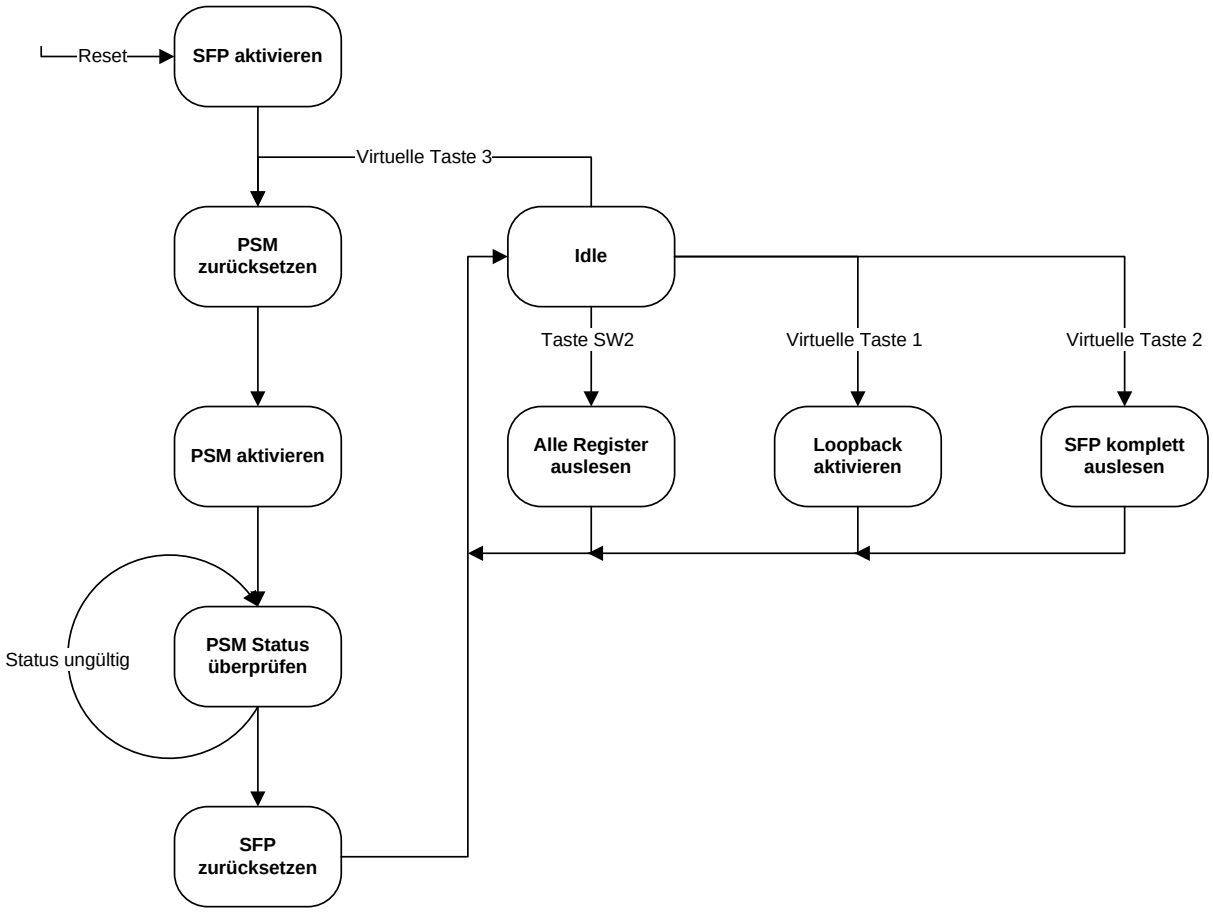

Abbildung 4.17: Strukturgramm der Management Interface Ansteuerung

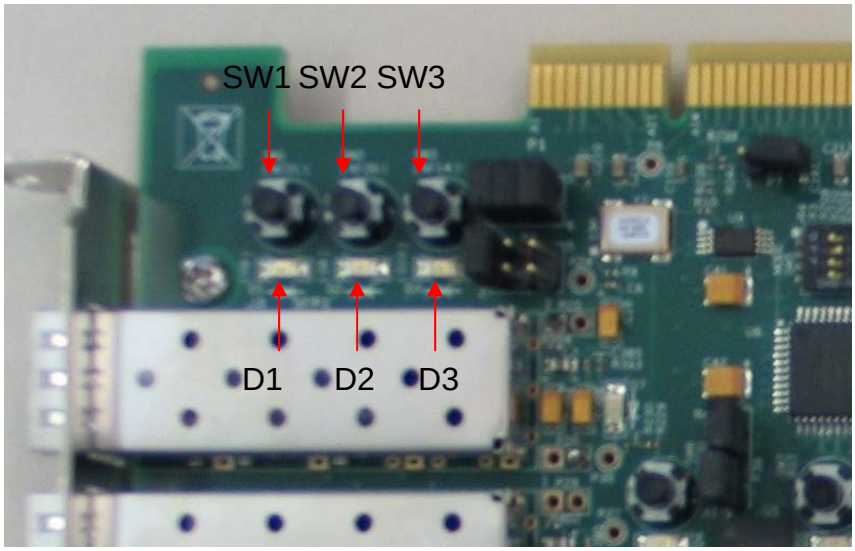

Abbildung 4.18: Positionen der Schalter und LEDs

ChipScope bietet eine Möglichkeit virtuelle IOs zu simulieren. So ist es möglich virtuelle Schalter in der ChipScope Umgebung zu integrieren. Über drei solche Schalter lassen sich noch weitere Funktionen über das Management Interface abrufen. Der erste virtuelle Schalter aktiviert die Loopback Funktion der GTP Transceiver. Mit dem zweiten Schalter lässt sich unabhängig vom Resetsignal die PSM neu starten, was für Testzwecke sehr hilfreich ist. Der letzte virtuelle Schalter startet eine Routine, die den kompletten EEPROM Speicher des SFPs ausliest.
Diese Funktionalitäten sind vorrangig für Testzwecke implementiert worden und auf die Ergebnisse kann nur im ChipScope zugegriffen werden. Dafür müssen in der obersten Hierarchieebene die entsprechenden ChipScope Komponenten (fibre_channel.vhd) implementiert sein. Die Komponenten werden, wie bei dem Fibre Channel Core und den Rocket IOs beschrieben, über den Xilinx Core Generator in das VHDL Programm integriert. Über Signalzuweisungen werden die gewünschten Signale aufgezeichnet. Die Komponenten sind zur Zeit auskommentiert, jedoch kann durch das Aufheben der Kommentierung ChipScope wieder schnell und einfach an der dortigen Stelle genutzt werden. Um im ChipScope Analyzer nicht alles neu einstellen zu müssen, sollte die entsprechende Datei (seqcheck.cpj oder sfpcheck.cpj) dafür geladen werden.

4.5 Loopback Tests

Um den bisherigen Aufbau stufenweise zu überprüfen bieten sich dafür verschiedene Loopback Tests an. Die GTP Transceiver besitzen alleine vier verschiedene Loopback Type, die in Abbildung 4.19 dargestellt sind. Die Testauswahl erfolgt über die 3 Bit breite Loopback Leitung. In Tabelle 4.5 ist gezeigt, mit welcher Bitkombination der entsprechende Loopback Typ ausgewählt wird.

Bei dem ersten Test werden dabei die Daten vor dem Parallel/Seriell-Wandler auf die Empfängerseite umgeleitet. Mit diesem Loopback kann überprüft werden, ob der PCS-Teil der GTPs funktioniert. Der zweite Test funktioniert in der gleichen Weise, nur mit dem kleinen Unterschied, dass der Loopback erst an der Treiberstufe erzeugt wird. Die Daten durchlaufen den kompletten GTP (PCS und PCM), inklusive der Parallel/Seriell- und anschließenden Seriell/Parallel-Wandlung.

Als Datengenerator dient der Fibre Channel Core. Mit dem Aktiveren der Port State Machine beginnt dieser mit dem Senden der Link Initialisierung und es wird die erste Grundsequenz (NOS) gesendet. Arbeitet der GTP ordnungsgemäß, wird der Core nach einen kleinen Verzögerung das eigene NOS Signal empfangen und mit der Link Initialisierung fortsetzen. Durch das Aufzeichnen der Busleitungen RxData und TxData kann der Prozess im ChipScope Analyzer nachvollzogen werden.

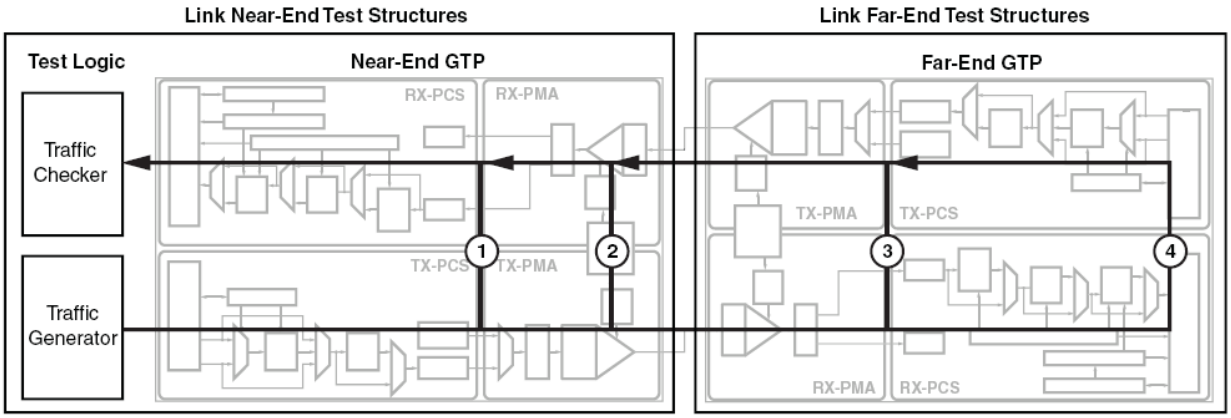

Abbildung 4.19: Übersicht der vier GTP Loopbacktests [8]

Tabelle 4.5: Bitkombinationen mit dazugehörigem Loopback Typ

Nummer	Loopback[2:0] (Binär)	Loopback Typ
-	000	Normaler Betrieb
1	001	Near-End PCS Loopback
2	010	Near-End PMA Loopback
3	100	Far-End PMA Loopback
4	110	Far-End PCS Loopback

Im nächsten Schritt wird der Loopback im GTP entfernt und ein externer optischer Loopback Adapter an das SFP angeschlossen (Abbildung 4.20). Somit kann das ganze System, Fibre Channel Core, GTP Transceiver und SFP Modul getestet werden. Auf Grund dieses Tests konnte eine fehlerhafte Terminierung zwischen SFP Modul und ML 555 festgestellt werden. Der Fehler kann durch eine einfach Brücke auf dem ML 555, wie in Abschnitt 4.1 beschrieben, verhindert werden. Nach der Änderung konnte auch dieser Test erfolgreich abgeschlossen werden.

Abbildung 4.20: Testaufbau mit Loopback Adapter

Da nun alle drei Hauptkomponenten soweit funktionieren, kann ein Far End PMA Loopback Test durchgeführt werden. Dieser leitet alle extern ankommenden Daten auf den Sender um. Somit bietet dieser Test zum ersten Mal die Möglichkeit eine Taktsynchronisierung mit dem Fibre Channel HBA zu überprüfen. Dazu wird das optische Kabel an das SFP Modul und an die Kaufkarte angeschlossen. Die mitgelieferte Software des HBAs besitzt einen Loopback Test. Dieser Test erwartet normalerweise einen Loopback Adapter am HBA. Durch das direkte Durchschleifen im GTP Transceiver wird dieser Effekt ebenfalls erreicht. Führt das Programm den Loopback Test erfolgreich durch, arbeitet die Synchronisierung im GTP fehlerfrei. Der letzte Loopback Test ist wiederum ähnlich dem vorherigen Test, jedoch mit der Möglichkeit, die Daten durch den kompletten GTP zu führen. Des Weiteren können die empfangenen Daten am RX Interface des GTP gelesen werden. Das ermöglicht nun, dass die gesendeten Loopback Daten vom HBA am ChipScope aufgezeichnet werden können.

Um letztendlich auch noch den Fibre Channel Core vollständig zu testen, sollen mit einem Pseudo-Zufallsgenerator kontinuierlich Daten erzeugt und in einem Fibre Channel Frame gesendet werden. Mit Hilfe des Loopback Adapters werden nun diese Daten wieder empfangen und mit den gesendeten Daten verglichen. Somit kann eine Aussage über die Bitfehlerrate getroffen werden.

Zur Realisierung dieser Aufgabe wird ein 36kByte großer Dual Port Block RAM verwendet. Da die Wortbreite im Fibre Channel 32 Bit beträgt, wird daher ebenfalls ein RAM mit 32 Bit Wortlänge verwendet. Durch die beiden unabhängigen RAM Ansteuerungen kann der erste Port dazu benutzt werden, um die gesendeten Daten fortlaufend zu speichern. Werden nun Daten empfangen, muss der zweite Port die abgespeicherten Daten der Reihe nach wieder lesen. Auf Grund der Signallaufzeit durch den Core, den GTP und wieder zurück, wird der Adresszählerstand des ersten Ports immer größer sein als der Adresszählerstand des zweiten Ports.

Ein einfaches Schieberegister wird als Pseudo-Zufallszahlengenerator verwendet. Die dabei entstehenden Daten werden in einen Fibre Channel Frame gepackt und über das Clientinterface vom Fibre Channel Core gesendet. Die ankommenden Daten werden mit den abgespeicherten Daten verglichen und im Fehlerfall wird ein Zähler inkrementiert. Das Messergebnis der Loopbacktests, das mit ChipScope aufgezeichnet wurde, ist in Abbildung 4.21 dargestellt.

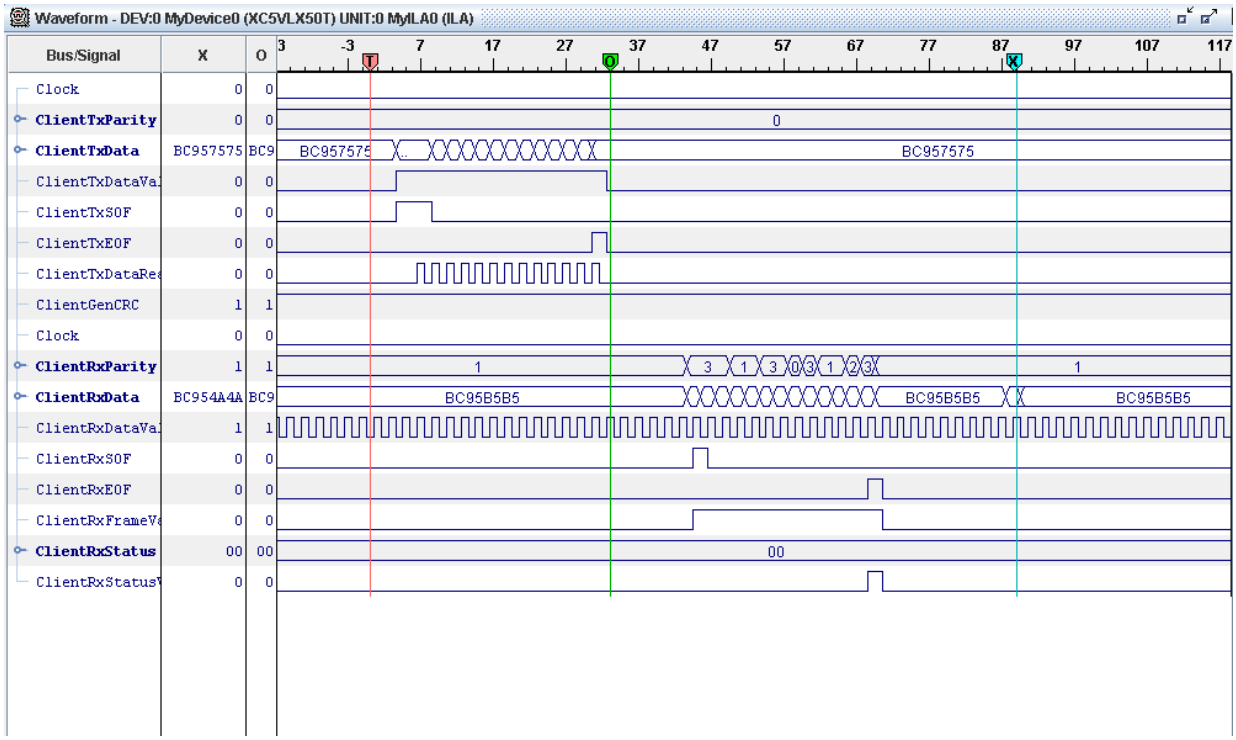

Abbildung 4.21: Ergebnis des Loopbacktests in ChipScope

Die Datei (loopback.vhd) mit dem Clientloopback Test befindet sich zwar nicht mehr im aktuellen Projekt, aber sie ist immer noch im Projektordner verfügbar und kann durch einfaches Ersetzen der aktuellen Clientdatei (vorher eine Sicherungskopie anfertigen) wieder integriert werden. Um den Test im ChipScope nachzuvollziehen, sollten auch dort die passenden Voreinstellungen (error.cpj) geladen werden.

Nachdem nun alle Komponenten fehlerfrei arbeiten und alle Tests positiv verlaufen sind, kann einen Schritt weiter gegangen werden. Die optische Schnittstelle kann nun mit der Kaufkarte verbunden werden. In ChipScope kann nun beobachtet werden, wie die Verbindung zwischen den beiden Teilnehmern initialisiert wird. Am Ende der Initialisierung senden beide Teilnehmer fortlaufend die Idle Sequenz. Dies signalisiert die Bereitschaft beider Teilnehmer für den weiteren Datenaustausch. In einem bestimmten Rhythmus sendet nun die Kaufkarte einen Fabric Login Frame. Die optische Schnittstelle ist nun bereit für die Implementierung eines höheren Protokolls, wie es in Kapitel 5 beschrieben wird.

5 Clientinterface

5.1 Struktur und Konzept

Nun soll die Verbindung zwischen dem HBA und dem ML 555 protokolltechnisch so programmiert werden, dass sie vom Betriebssystem des HBAs erkannt wird. Die einfachste Art schien zu sein dem Betriebssystem mitzuteilen, dass eine Festplatte angeschlossen ist. Für diese Lösung muss jedoch ein SCSI Protokoll verwendet werden. Um nun auf die Login Mechanismen im Fibre Channel Protokoll reagieren zu können und die Login Authentifizierung über SCSI an das Betriebssystem zu senden, muss ein entsprechendes Interface programmiert werden. Diese soll über die Clientschnittstelle des Fibre Channel Cores arbeiten. Diese Schnittstelle am Fibre Channel Core bietet dem Anwender eine einfache Möglichkeit Daten zu senden und zu empfangen.

In Abbildung 5.1 ist der zeitliche Ablauf eines empfangenen Rahmens dargestellt. Wird nun ein Rahmen empfangen, wird das Signal ClientRxFrameValid vom Core auf Eins gesetzt und bleibt solange auf Eins bis der Rahmen zu Ende ist. Die Signale ClientRxSOF und ClientRxEOF markieren den Anfang und das Ende des Rahmens und sind nur zu dieser Zeit auf dem Wert Eins. Mit einer fallenden Flanke an ClientRxDataValid wird dem Anwender nun mitgeteilt, dass die Daten an ClientRxData gültig sind. Es findet keine Rückmeldung statt, ob der Anwender die Daten übernommen hat.

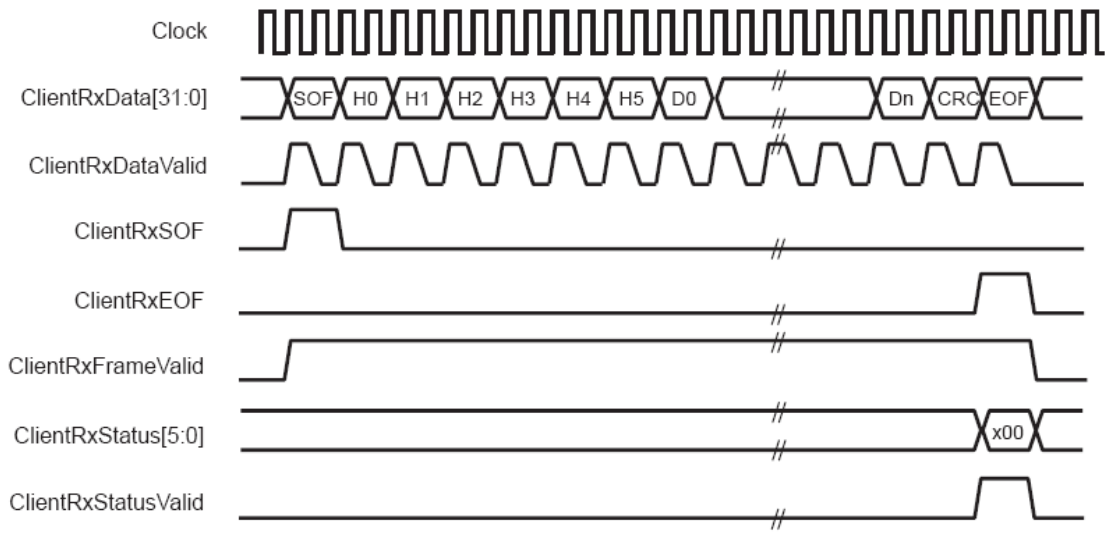

Abbildung 5.1: Zeitdiagramm beim Empfangen von Daten [9]

Die Abbildung 5.2 zeigt hier die einzuhaltende zeitliche Abfolge, wenn Daten zu senden sind. Beim Senden müssen alle Signale, mit Ausnahme von ClientTxDataRead, vom Anwender gesetzt werden. ClientTxDataValid wird über die komplette Dauer auf Eins gesetzt. Die Signale ClientTxSOF und ClientTxEOF markieren wieder den Anfang bzw. das Ende des Rahmens. Mit einer Eins auf der Leitung ClientTxDataRead wird nun dem Benutzer mitgeteilt, dass der Core die Daten an ClientTxData übernommen hat, danach wechselt der Pegel wieder auf Null und es können neue Daten auf den Bus gestellt werden. Das Signal ClientGenCRC sollte immer auf Null gesetzt werden. Somit übernimmt der Core die Aufgabe den CRC Wert für den gesendeten Frame zu bestimmen und an deren Stelle einzufügen.

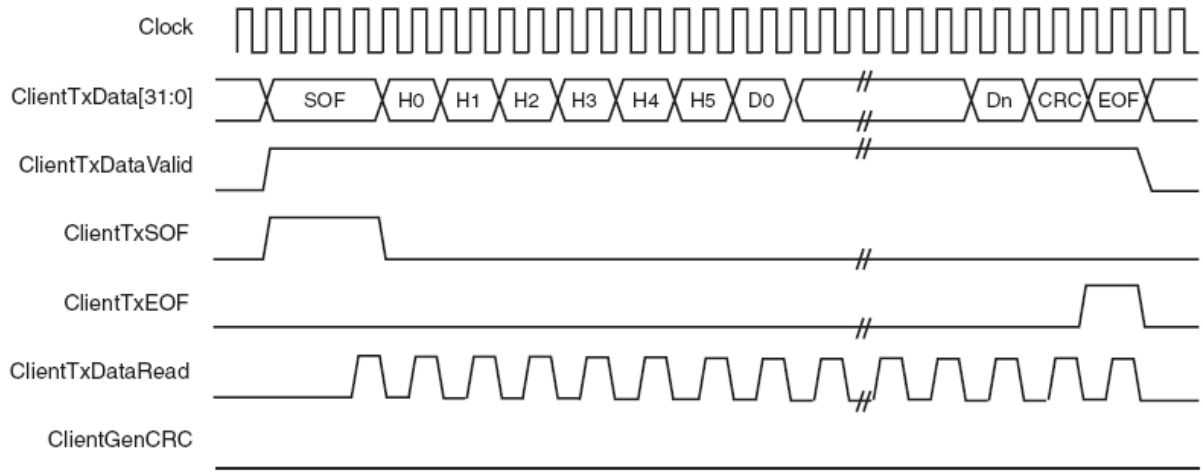

Abbildung 5.2: Zeitdiagramm beim Senden von Daten [9]

Die zu programmierende Software soll den empfangenen Befehl erkennen und einen passenden Frame als Antwort senden. Der Host Bus Adapter soll also der Initiator sein und immer befehlen, was zu tun ist. Daraufhin soll das ML 555 mit der richtigen Antwort reagieren. Mit dieser Strategie soll nun nach und nach der komplette Ablauf realisiert werden. Dazu muss die Anwendung so konzipiert sein, dass neue Befehle schnell und einfach integrierbar sind und bestehende Frames sich für Versuchszwecke leicht ändern lassen.
Aus diesen Vorgaben ist eine flexible und einfach erweiterbare Anwendung entstanden, die in drei Hauptteile gegliedert ist und in den folgenden Abschnitten vorgestellt wird.

5.2 Steuerung

Die Steuerung hat die Aufgabe auf empfangene Befehle zu reagieren und die passende Antwort zu senden. In Abbildung 5.3 ist eine grobe Übersicht über die Abfolge der Rahmen dargestellt.

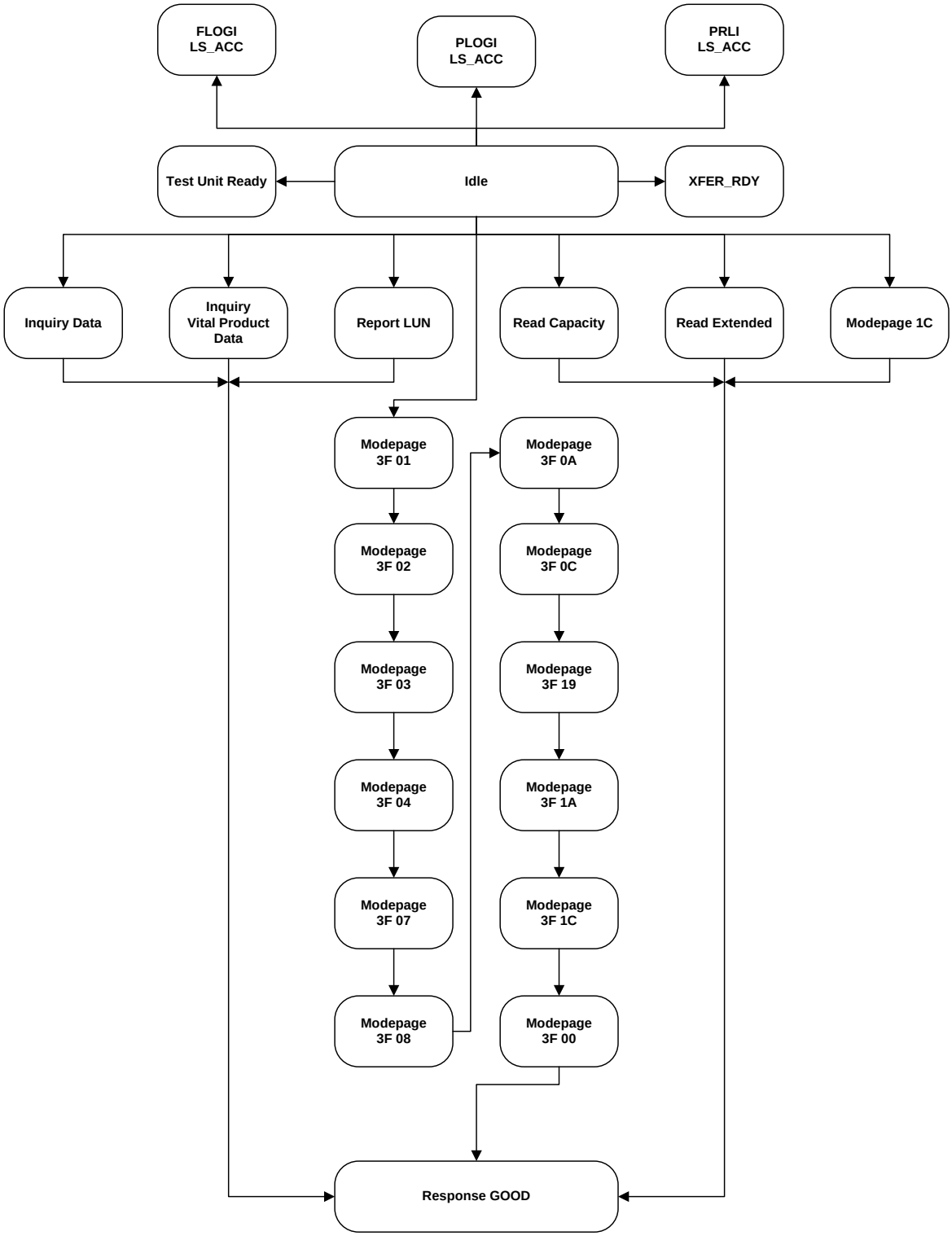

Abbildung 5.3 Übersicht der Rahmenabfolge in der Steuerung

Wird ein bekannter Befehl detektiert, wechselt die Steuerung von ihrem Idle Zustand zu der passenden Antwort des Befehls. Als erstes wird der Header des Rahmens zusammengebaut. Dieser setzt sich aus Angaben des empfangen Rahmens und fest vorgeschriebenen Angaben zusammen. Diese Vorgehensweise ist bei allen Rahmen gleich. Als nächstes wird der Nutzdatenblock bestimmt. Um die Nutzdaten für jeden Rahmen übersichtlich zu gestalten, befinden sie sich in einem ROM. Somit kann auf einfache Art und Weise jedem Rahmen durch die Angabe einer Speicheradresse, die passenden Nutzdaten zugewiesen werden. Ist der Rahmen fertig gestellt, wird dem Sender die Freigabe erteilt. Müssen keine weiteren Rahmen konstruiert werden, wechselt die Steuerung nach dem Erhalt der Sendebestätigung wieder in den Idle Zustand. Soll ein weiterer Rahmen gesendet werden, wartet die Steuerung zuerst auf eine Empfangsbestätigung vom HBA in Form eines R_RDY Signals. Erst danach kann ein weiterer Rahmen gesendet werden.

5.3 Sender

Der Sender hat die Aufgabe das TxClientinterface des Fibre Channel Cores anzusteuern. Dabei müssen die Vorgaben des Cores berücksichtigt werden. Ein Sendezyklus eines 32 Bit Wortes ist in zwei Schritte unterteilt, die sich immer wiederholen. Im ersten Schritt werden die Daten auf den ClientTxData Bus gestellt und die Signalleitungen entsprechend gesetzt. Im zweiten Schritt wird gewartet bis der Core die Daten übernommen hat. Dies wird durch eine steigende Flanke an ClientTxDataRead gekennzeichnet.

Die Abbildung 5.4 zeigt die Sendeeinheit als Strukturdiagramm. Der Sender wird durch ein Freigabesignal von der Steuerung gestartet. Nun wird der komplette Rahmen wortweise gesendet. Beim Erreichen der Nutzdaten werden die Daten aus dem ROM gelesen und auf den ClientTxData Bus gestellt. Der Adresszähler des ROMs wird dazu mit der Nutzdatenadresse initialisiert und von nun an nach jedem gesendeten Datenwort inkrementiert. Diese Schleife wird mit dem Auslesen eines speziellen Datenworts beendet. Das Datenwort ist F0 F0 F0 F0h und soll das Ende der Nutzdaten im ROM markieren. Um den kontinuierlichen Datenfluss an dieser Stelle nicht zu unterbrechen, wird vor dem Verlassen der Schleife dieses Datenwort ebenfalls noch gesendet.

Im nächsten Schritt wird vom Fibre Channel Core ein „leeres“ Datenwort erwartet, um diese Stelle mit dem ermittelten CRC Wert zu überschreiben. Daher kann das Datenwort F0 F0 F0 F0h bedenkenlos gesendet werden. Damit wird auch ein Schritt eingespart und es kann sofort zum End of Frame übergegangen werden. Hat der Sender einmal die komplette Prozedur durchlaufen, wird der Steuerung dies per Sendebestätigung mitgeteilt. Nun befindet sich der Sender wieder im Idle Zustand und ist bereit, einen neuen Rahmen zu senden.

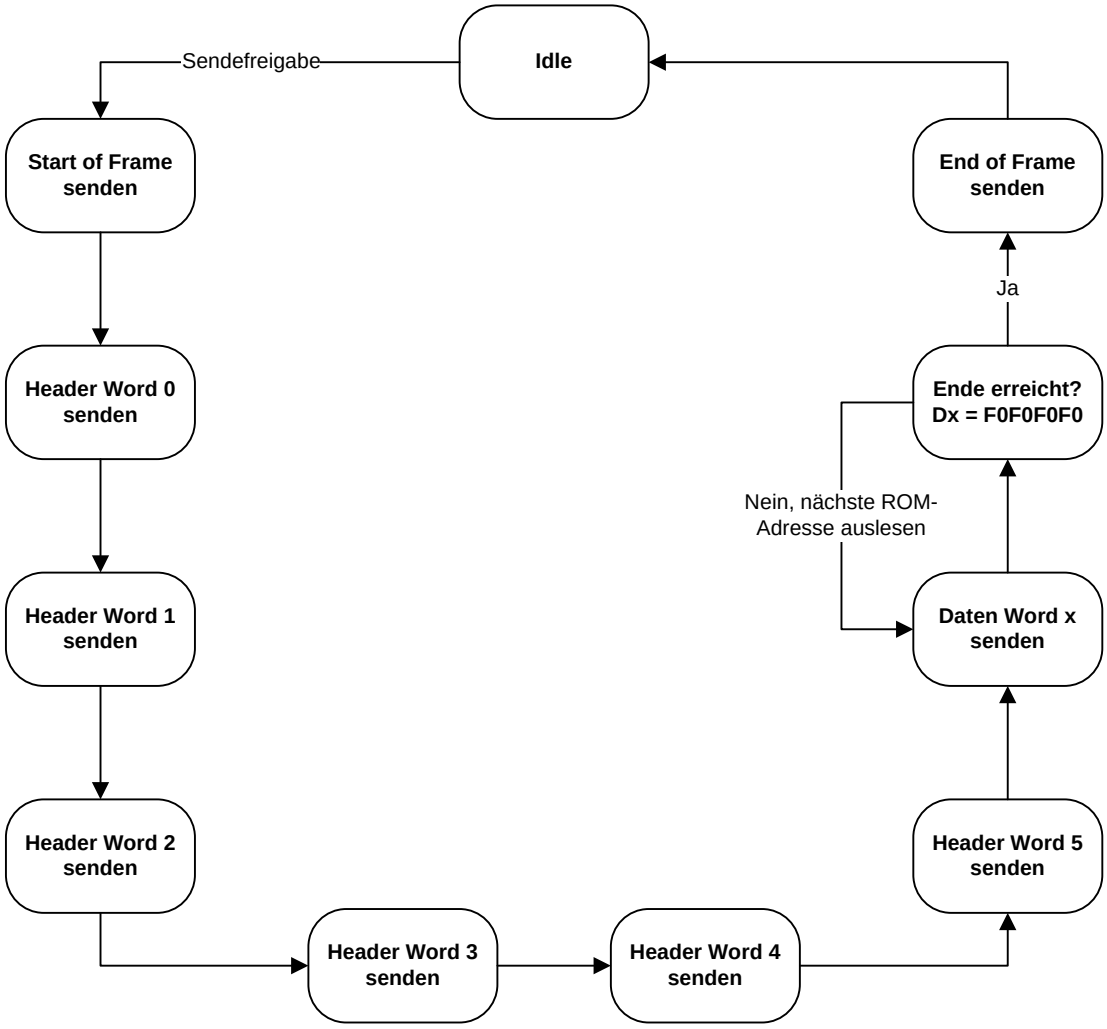

Abbildung 5.4 Strukturdiagramm der Sendeeinheit

5.4 Empfänger

Der Empfänger muss ebenfalls die Vorgaben des Fibre Channel Cores erfüllen, die im Kapitel 5.1 beschrieben sind, um die Daten vollständig zu erhalten. Die Abbildung 5.5 zeigt das Strukturgramm der Empfangseinheit. Diese nimmt erst ihre Arbeit auf, wenn die Verbindung initialisiert und durch das Drücken der Taste SW3 (Abbildung 4.18) freigegeben ist. Nun wechselt der Empfänger in den Idle Zustand und wartet darauf, dass der Fibre Channel Core Daten liefert.
Wird das Primitive Signal R_RDY empfangen, ändert der Empfänger den Zustand um der Steuerung mitzuteilen, dass der gesendete Frame beim HBA angekommen ist. Danach wechselt er wieder in Idle.

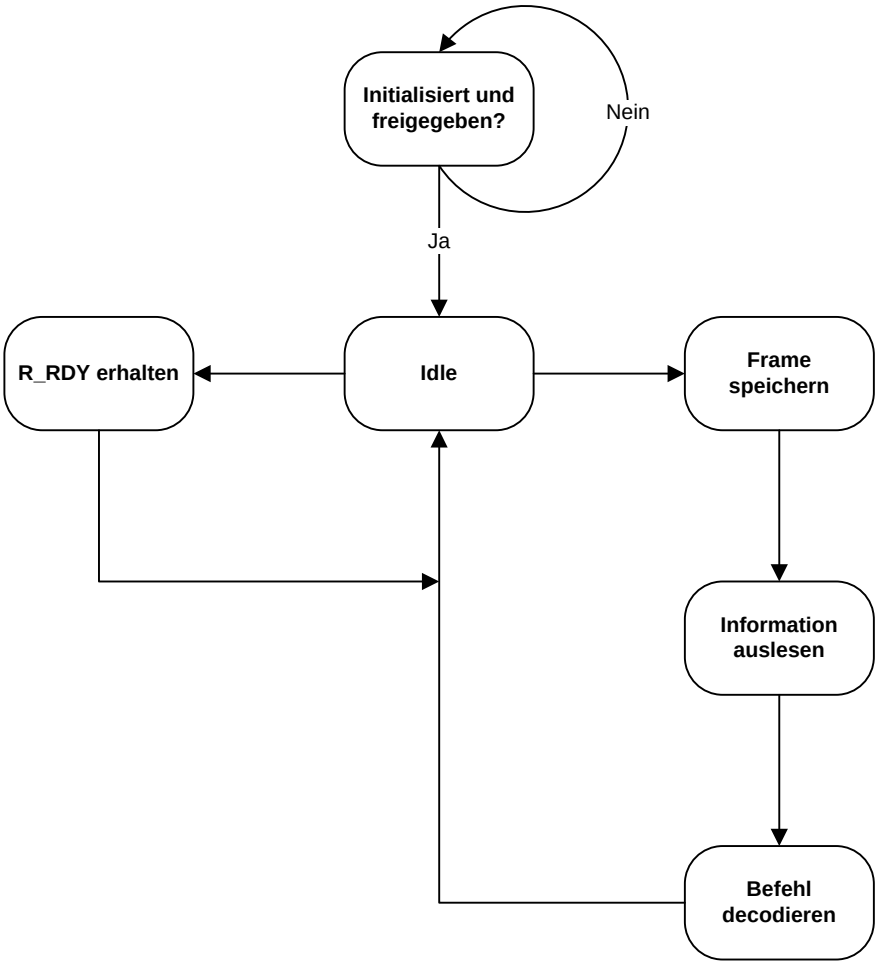

Abbildung 5.5 Strukturdiagramm der Empfangseinheit

Wird durch das Signal ClientRxFrameValid angezeigt, dass Daten vorhanden sind, wechselt der Zustand in Rahmen speichern. Dort wird der Rahmen wieder wortweise zwischengespeichert. Da die Daten kontinuierlich vom Fibre Channel Core geliefert werden, ist dieser Schritt notwendig, ansonsten kann es zum Datenverlust kommen. Ist der komplette Rahmen übertragen, können alle wichtigen Daten aus dem RAM ohne Zeitdruck ausgelesen werden. Der Adresszähler beginnt immer mit der Speicheradresse 0x000h. Damit ist gewährleistet, dass die gewünschten Informationen immer an der gleichen Stelle im RAM stehen. Aus verschiedenen Informationen im Header und im Nutzdatenblock kann nun der Befehl entschlüsselt werden. Im Anschluss wird der Steuerung mitgeteilt, dass ein neuer Befehl vorliegt. Nun wechselt der Empfänger wieder in den Idle Zustand und ist bereit, neue Daten entgegenzunehmen.

6 Ergebnis

Es ist gelungen eine funktionsfähige optische Schnittstelle, die auf der Basis des Fibre Channel Standards arbeitet, in eine FPGA Hardware zu integrieren. Dazu wurde das ML 555 entsprechend konfiguriert. Die Rocket IOs im FPGA mussten so eingestellt werden, dass der Datenaustausch zum SFP Modul und zum Fibre Channel Core ordnungsgemäß funktioniert. Der Fibre Channel Core wurde erfolgreich in das FPGA implementiert und es wurden alle benötigten Randbedingungen, die der Fibre Channel Core dafür benötigt, erfüllt. Mittels einem Loopback Test kann nachgewiesen werden, dass der Fibre Channel Core sich richtig Initialisiert und gesendete Rahmen wieder fehlerfrei empfangen werden (Abbildung 6.1).

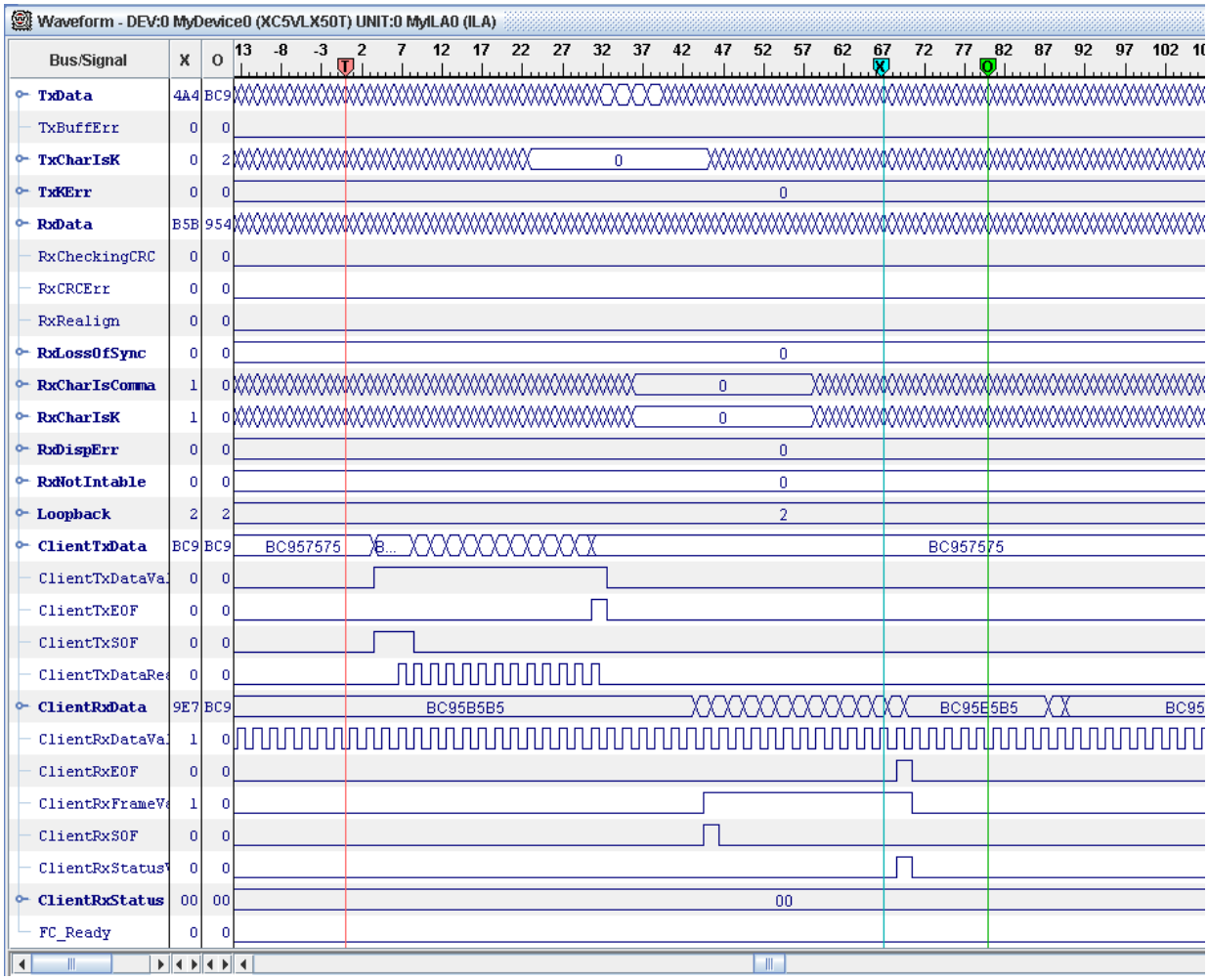

Abbildung 6.1: Messergebnis eines gesendeten und empfangenen Rahmens

Als Folge der erfolgreichen Arbeit konnte nun eine Initialisierung zu dem HBA hergestellt werden. Mit Hilfe von ChipScope konnte nun die Login Prozedur, auf Fibre Channel Protokoll Ebene, nachvollzogen werden. Dazu sendete die Karte Fabric, Port und Prozess Login Rahmen. Diese Rahmen werden vom FPGA sinngemäß beantwortet. Nach erfolgreicher Login Prozedur leuchtet nun an der HBA Karte eine LED auf. Diese signalisiert, dass eine 1 GB/s Verbindung zu einem anderen Teilnehmer hergestellt ist. In der mitgelieferten Software des HBAs wird die programmierte Hardware erkannt und erste Daten, wie die Port- und Node Name sind vorhanden (Abbildung 6.2).

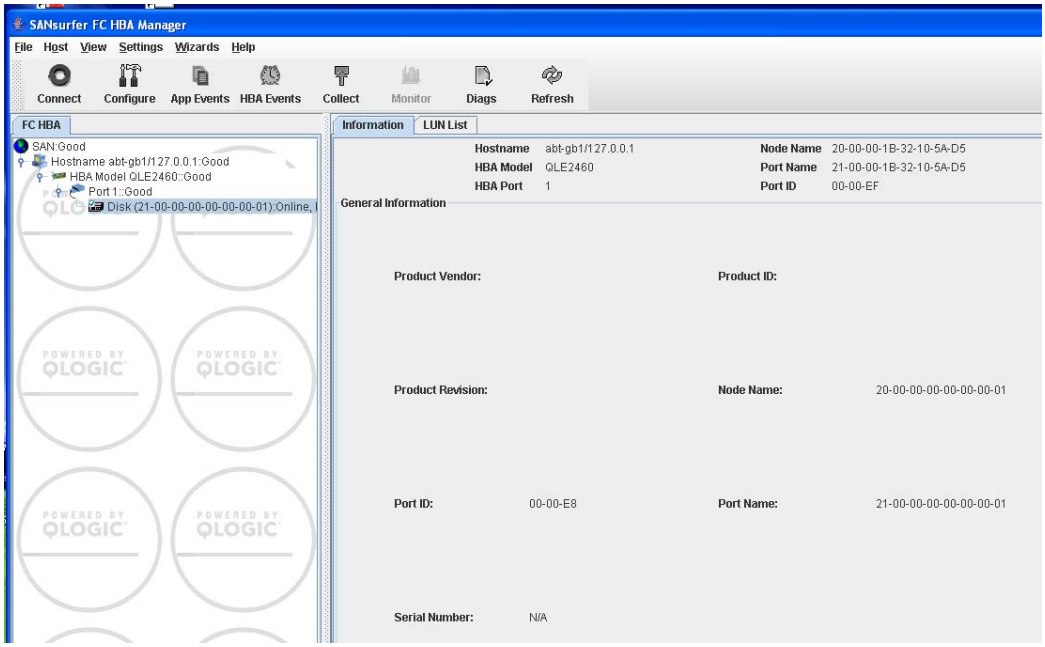

Abbildung 6.2 Das Ergebnis in SAN Surfer

Ab dieser Stelle wird es nun problematisch. Leider konnte keine Möglichkeit gefunden werden direkt auf die Fibre Channel Daten zuzugreifen. Eine Variante zur Problemlösung ist eine Authentifizierung der FPGA Hardware als SCSI Festplatte. Damit sollte Windows in der Lage sein, das ML 555 als Festplatte zu erkennen. Dies bietet programmiertechnisch einen einfachen Zugriff.
Hierfür ist es notwendig, die gewünschten Informationen mit Hilfe des SCSI Befehlssatzes zu versenden. Das erforderte nun tiefer Kenntnisse im SCSI Protokoll und verlangte diesen Standard im Zusammenhang der Fibre Channel Standards in das FPGA zu integrieren.

Die Authentifizierung funktioniert leider nur im Ansatz, wie die Abbildung 6.3 zeigt. Es darf nur eine Festplatte erkannt werden, die anderen werden im SCSI Protokoll deaktiviert. Des Weiteren hat es den Anschein, dass inhaltliche Änderungen im SCSI Rahmen keine Auswirkungen auf den Login Prozess haben.

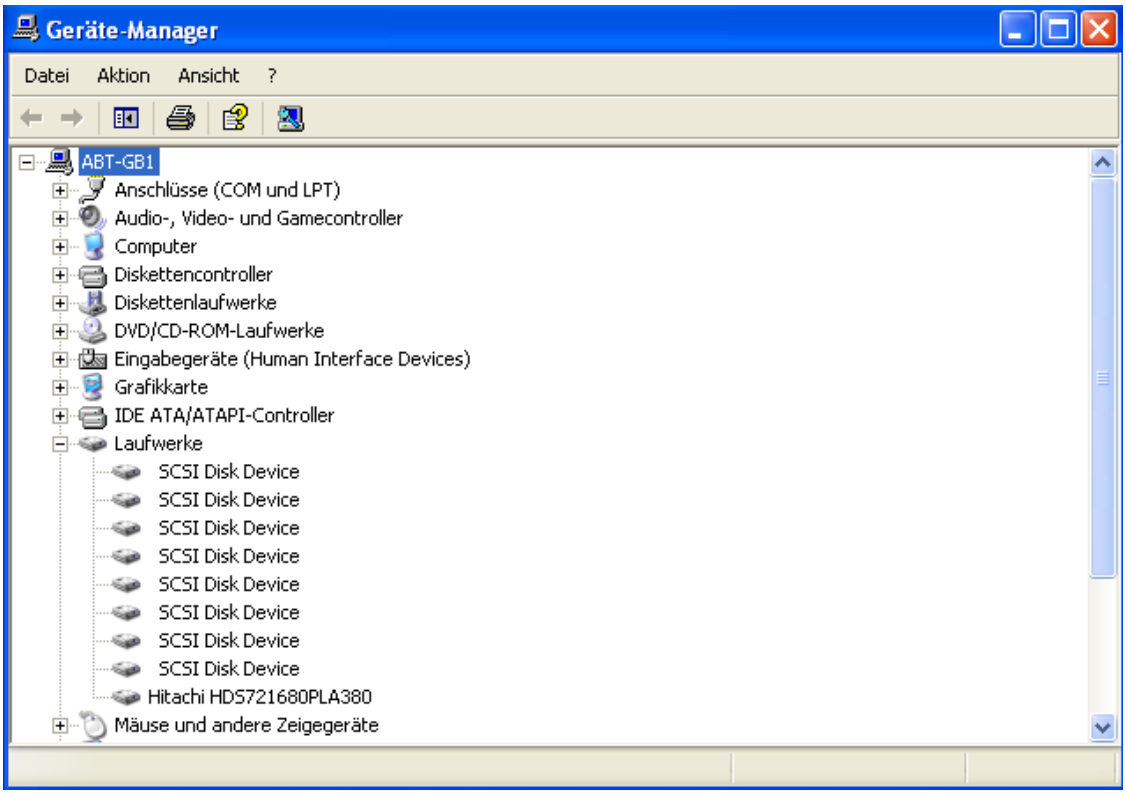

Abbildung 6.3 Windows erkennt acht SCSI Disk Device

Mit Hilfe der mitgelieferten SAN Software kann ein Read/Write Buffer Test durchgeführt werden. Dabei wird ein einstellbares Testmuster gesendet. Der Datenrahmen kann mit Hilfe von ChipScope aufgezeichnet werden (Tabelle 6.1).
Auffällig an dem Datenrahmen ist, dass nach dem Testmuster AAh, 24 Bytes gesendet werden, die durch keinen Standard erklärbar sind.

Tabelle 6.1 Empfangene und gesendete Read/Write Buffer Rahmen

Read/Write Buffer Test

Anfrage	Antwort	Daten
BCB55656	BCB55656	BCB55656
060000E8	050000EF	010000E8
000000EF	000000E8	000000EF
08290000	08890000	08090008
XX000000	FF000000	XX+1000000
YYYYFFFF	YYYYFFFF	YYYYFFFF
00000000	00000000	00000000
00000000	00000000	AAAAAAAA
00000000	00000020	AAAAAAAA
00000001	00000000	25D6917C
3B020000	ZZZZZZZZ	CFEA917C
00000000	BC957575	30FD8B03
20000000		01000000
00000000		17000100
00000020		00000000
ZZZZZZZZ		ZZZZZZZZ
BC957575		BC957575

Dies könnte die Fehlerursache sein, weshalb die vom FPGA gesendeten Datenrahmen nicht vom HBA erkannt werden. Leider konnte diese Frage in dieser Diplomarbeit nicht mehr beantwortet werden.

7 Zusammenfassung

Die Diplomarbeit wurde am Fraunhofer Institut für zerstörungsfreie Prüfverfahren in der Abteilung „Systementwicklung und Prototypenbau“ in Saarbrücken durchgeführt. Das Fraunhofer Institut entwickelt Prüfgeräte zur Charakterisierung und Analyse von Werkstoffen. Dafür müssen die Messdaten von einem Messsystem zu einem Computer gesendet werden. Zur Zeit wird dafür eine 100 Mbit/s Ethernetverbindung verwendet, die sich jedoch an der Kapazitätsgrenze befindet. Um nun weiterhin die Messdaten schnell und zuverlässig zu senden, sollte eine optische Verbindung verwendet werden, die auf einem zukunftssicheren Standard basiert.

Aus dieser Motivation heraus entstand die Aufgabenstellung für zwei Diplomarbeiten. Es sollte eine optische Datenübertragungsstrecke mit bis zu 4 GBit/s unter Verwendung des Fibre Channel Protokolls realisiert werden. Die optische Verbindung sollte zwischen einem Host Bus Adapter und einem FPGA-Entwicklungsboard hergestellt werden. Diese Diplomarbeit behandelt dabei die Integrierung der optischen Schnittstelle in das Entwicklungsboard und die protokollspezifischen Mechanismen, die zur Initialisierung der Verbindung benötigt werden.

Um diese optische Verbindung zu realisieren, sollte ein Fibre Channel Core von Xilinx verwendet werden. Dieser steuert über sogenannte Rocket IO GTP Transceiver ein SFP Modul an, das die Umsetzung elektrisch in optisch bzw. optisch in elektrisch durchführt.

Im ersten Schritt musste man sich mit der Entwicklungsumgebung von Xilinx vertraut machen und sich in die Entwicklungswerkzeuge ISE 10.1, Core Generator und ChipScope einarbeiten. Der nächste Schritt verlangte nun, sich mit dem Fibre Channel Protokoll vertraut zumachen und die Vorgehensweise bei der Integrierung eines Cores und der Rocket IOs kennenzulernen. Danach musste die FPGA Hardware so konfiguriert werden, dass alle Rahmenbedingungen für die verwendeten Komponenten erfüllt werden. Als nächstes wurden die Rocket IOs, nach der Vorgabe des SFPs und Fibre Channel Core, eingestellt. Dabei musste auch die Verschaltung auf dem Entwicklungsboard beachtet werden. Nun konnte der Fibre Channel Core eingefügt werden. Um das komplette System nun Schrittweise zu testen, wurden verschiedene Loopback Tests durchgeführt und gleichzeitig die Statusregister des Fibre Channel Cores ausgelesen.

Das Ergebnis des Loopbacktests war eine funktionstüchtige, optische Schnittstelle, die bereits eine Link Initialisierung, dank des Fibre Channel Cores, zu dem Host Bus Adapters durchführt. Um über einen Computer Zugriff auf die Nutzdaten des Fibre Channel Rahmens zu bekommen, wurde nun eine Authentifizierung an das Betriebssystem als SCSI Festplatte gesendet. Das verlangte im ersten Schritt den SCSI Standard zu studieren. Im nächsten Schritt musste ein VHDL Programm entwickelt werden, dass den SCSI Befehlssatz über das Fibre Channel Protokoll versendet. Jedoch konnte im Rahmen dieser Diplomarbeit nur eine teilweise funktionierende SCSI Authentifizierung realisiert werden, da es schwierig war, an entsprechende Informationen, die das SCSI Protokoll betreffen, zu kommen.
Da dieses Projekt sehr interessant für weitere Entwicklungen am Fraunhofer Institut ist, wird sich nun jemand aus der Softwareabteilung mit dem SCSI Protokoll befassen. Diese Diplomarbeit bietet dafür eine gute Grundlage und eine funktionierende, optische Schnittstelle, die alle Fibre Channel Spezifikationen erfüllt.

8 Quellenverzeichnis

[1] Reichardt, J.; Schwarz B.: VHDL-Synthese. Oldenburger Wissenschaftsverlag, München, Wien, Oldenburg, 2. überarbeitete Auflage, 2001

[2] TECChannel: Fibre Channel: Funktion und Technologie. 02. Oktober 2007 http://www.tecchannel.de/storage/grundlagen/465690/

[3] Technical Committee T11: Fibre Channel Framing and Signaling Interface (FC-FS). 09. April 2003, http://www.t11.org/ftp/t11/pub/fc/fs/03-173v1.pdf

[4] Technical Committee T10: Fibre Channel Protocol (FCP-3). 13. September 2005, http://www.t10.org/ftp/t10/drafts/fcp3/fcp3r04.pdf

[5] Technical Committee T10: SCSI Primary Commands (SPC-4). 29. Juli 2008, http://www.t10.org/ftp/t10/drafts/spc4/spc4r16.pdf

[6] Technical Committee T10: SCSI Block Commands (SBC-3). 25. August 2008, http://www.t10.org/ftp/t10/drafts/sbc3/sbc3r16.pdf

[7] Xilinx, Virtex-5 ML555 Development Kit User Guide. 18. Juni 2007 http://www.xilinx.com/support/documentation/boards_and_kits/ug201.pdf

[8] Xilinx, Virtex-5 FPGA RocketIO GTP Transceiver User Guide. 11. Februar 2008 http://www.xilinx.com/support/documentation/user_guides/ug196.pdf

[9] Xilinx, Logic CORE™ IP Fibre Channel v3.3 User Guide. 24. März 2008

[10] Seagate, Fibre Channel Interface Product Manual.01. August 2000 http://www.seagate.com/support/disc/manuals/fc/67496b.pdf

[11] HITACHI, Hard Disk Specification Version 1.4. 01. November 2006

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Anhang

A1: Schaltplan ML 555 mit Brücke für SFP1 Steckplatz

A2: CD-ROM mit folgendem Inhalt:

- komplettes VHDL Projekt
- FC-FS Standard
- FCP-3 Standard
- SPC-4 Standard
- SBC-3 Standard
- ML 555 Development User Guide
- Rocket IO GTP Transceiver User Guide
- Fibre Channel User Guide
- Fibre Channel Interface Product Manual
- Hard Disk Specification
- Datenblatt: TXN31115D2_DS (SFP)
- Datenblatt: Virtex 5
- Schaltplan: ML 555
- Bestückungsplan ML 555

A1: Schaltplan ML 555 mit Brücke für SFP1 Steckplatz