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	080125-TW	Textseiten	79
Dipl.-Ing. C.Weingard	165956	Vertraulichkeitsgrad		Anlagen	CD-ROM

Verfasser	**Abteilung**	**Telefon**	**Unterschrift**
Weber,Christoph	Gerätebau		

Freigabe für Inhalt		**Freigabe für externe Verteilung**	
Dipl.-Ing. C.Weingard		Dipl.-Ing. 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 behandelt im ersten Teil die PC- seitige Anbindung an die Übertragungsstrecke. Ziel war es, den Zugriff auf die Interfacekarte mit Hilfe von C oder C++ Programmierung und der Fibre Channel HBA API Shared Library für Windows herzustellen und anschließend eine Testsoftware zu entwickeln, um ein weiterverwendbares Softwaremodul zur Einbindung in andere Projekte zu erhalten. Mit zunehmenden Fortschritt der Arbeit wurden die Grenzen des ersten Ansatzes immer deutlicher. Daher wurde nach weiteren Möglichkeiten zur Problemlösung gesucht. Das Sinnvollste schien hier, das FPGA so zu programmieren, dass es sich als SCSI- Festplatte im Windows Betriebssystem anmeldet. Mit einfachen File I/O- Befehlen , sollte so ein erster Datenaustausch realisiert werden. Um die FPGA- Programmierung zu beschleunigen, beschäftigt sich der zweite Teil dieser Diplomarbeit mit dem SCSI- Protokoll.

Verteiler[1]

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

Fraunhofer Institut
für zerstörungsfreie
Prüfverfahren

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

Christoph Weber
geb. in D 66113 Saarbrücken

Matrikelnummer:
3419606

Betreuer: Dipl.- Ing. (FH) Christoph Weingard (IzfP)

Dipl.- Ing. (FH) Hendrik Theado (IzfP)

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

Christoph Weber | Diplomarbeit | Matrikelnummer: 3419606

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 ……

Christoph Weber

Inhaltsverzeichnis:

1 Einleitung und Aufgabenstellung

Computer mit höchsten Rechenleistungen, Messgeräte mit den genauesten Messergebnissen sind nur zwei von unzähligen technischen Errungenschaften, die ständig verbessert und weiterentwickelt werden. Die dazu analog ansteigende Menge von Daten muss immer schneller übertragen werden, um drohende Datenstaus und daraus resultierende Informationsverluste zu vermeiden. Zur Zeit wird am Fraunhofer Institut für zerstörungsfreie Prüfverfahren zur schnellen Datenübertragung Ethernet, mit einer Übertragungsgeschwindigkeit von 100Mbit/s, eingesetzt. Die Probleme mit diesem System entstehen nicht nur durch die steigenden Datenmengen, welche übertragen werden sollen, sondern auch durch die Störanfälligkeit für elektromagnetische Einflüsse des Übertragungsmediums . Aus diesem Grund wird nach einer Möglichkeit gesucht, das bestehende Ethernet- System durch eine bessere Technologie zu ersetzen.

Resultierend aus der oben genannten Problematik ergab sich für mich die Möglichkeit, eine Diplomarbeit am IZFP zu schreiben. Im Rahmen dieser Arbeit sollte eine optische Datenübertragungsstrecke mit bis zu 4 Gigabit / s zur schnellen Datenübertragung zwischen Frontend mit Prüfhardware und einem Bedienrechner unter Nutzung des Fibre Channel Protokolls aufgebaut, in Betrieb genommen und verifiziert werden. Dazu sollte zunächst eine optische Verbindung zwischen einem FPGA-Entwicklungsboard (FPGA = Field Programmable Gate Array) der Firma Xilinx und einem Fibre Channel fähigen PCI- Express Host Bus Adapter (HBA) etabliert und im Folgenden eine Maximierung der Nutzdatenrate an der laufenden Strecke vorgenommen werden, mit dem Ziel, die Ergebnisse in zukünftigen Projekten zu verwenden. Die Motivation der Arbeit lag im Erreichen einer möglich hohen Datenübertragungsrate zwischen Frontendelektronik und Bedienrechner wie zum Beispiel bei der Übertragung von HF- Rohdaten aus Mehrkanal- Prüfsystemen gefordert wird. Dabei soll es später möglich sein, unter Verwendung des Fibre

Channel Standards, PC- seitig verschiedene Interfacekarten einzusetzen, um so auch für zukünftige Anwendungen gerüstet zu sein.
Auf Grund des Umfangs wurde das Thema in zwei Diplomarbeiten unterteilt. Eine Diplomarbeit beschäftigt sich mit dem Design und der Implementierung der optischen Schnittstelle in der FPGA- Hardware , sowie dem Erstellen von Automatismen, die anschließenden Tests dienen sollen.
Diese Diplomarbeit behandelt im ersten Teil die PC- seitige Anbindung an die Übertragungsstrecke. Ziel war es, den Zugriff auf die Interfacekarte mit Hilfe von C oder C++ Programmierung und der Fibre Channel HBA API Shared Library für Windows herzustellen und anschließend eine Testsoftware zu entwickeln, um ein weiterverwendbares Softwaremodul zur Einbindung in andere Projekte zu erhalten. Mit zunehmenden Fortschritt der Arbeit wurden die Grenzen des ersten Ansatzes immer deutlicher. Daher wurde nach weiteren Möglichkeiten zur Problemlösung gesucht. Das Sinnvollste schien hier, das FPGA so zu programmieren, dass es sich als SCSI-Festplatte im Windows Betriebssystem anmeldet. Mit einfachen File I/O- Befehlen , sollte so ein erster Datenaustausch realisiert werden. Um die FPGA- Programmierung zu beschleunigen, beschäftigt sich der zweite Teil dieser Diplomarbeit mit dem SCSI-Protokoll.

2 Fibre Channel Grundlagen[7, 8]

2.1 Überblick

Fibre Channel ist ein Standardprotokoll aus dem Bereich der Speichernetzwerke und ist für serielle, kontinuierliche Hochgeschwindigkeitsübertragungen großer Datenmengen konzipiert worden. Viele Storage Area Networks (SANs) basieren heute auf der Implementierung des Fibre Channel Standards. Die erreichten Datentransferraten betragen heute bis zu 8 Gbits /s . Als Übertragungsmedium wird sowohl Kupferkabel als auch Glasfaserkabel verwendet. Letztere sind dagegen störunanfällig gegenüber elektromagnetischen Einflüssen.
Ähnlich wie bei klassischen Netzwerken, bei denen jede Netzwerkkarte eine MAC-Adresse hat, hat bei Fibre Channel jedes Gerät einen WWNN (World Wide Node Name) sowie jeder Port pro Gerät einen WWPN (World Wide Port Name). Es handelt sich dabei um einen 64-Bit-Wert (meist hexadezimal dargestellt), der jedes Fibre Channel Gerät eindeutig identifiziert. FC- Geräte können über mehr als nur einen Port verfügen, in diesem Fall hat das Gerät weiterhin nur eine WWNN, aber es besitzt WWPNs in der gleichen Anzahl wie es Ports besitzt. Die WWNN und die WWPN sind sich in der Regel sehr ähnlich.
Die erhältlichen Erweiterungskarten, die es Servern ermöglicht, über Fibre Channel zu kommunizieren, nennt man Hostbus Adapter, kurz HBA.

2.2 Topologien

Die Fibre Channel Technologie ist durch seine drei möglichen Topologien sehr flexibel anwendbar. Die Abbildungen 2.2.1 bis 2.2.3 sollen einen kleinen Überblick verschaffen. In dieser Diplomarbeit soll lediglich eine Point-to-Point Struktur realiesiert werden.

2.2.1 Point-to- Point Topologie

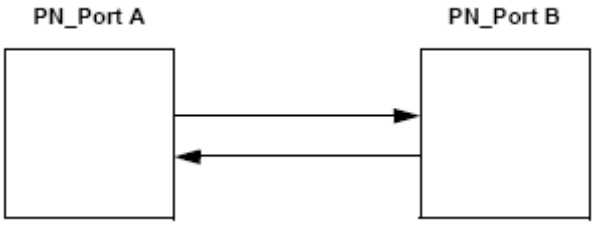

Abbildung 2.2.1: Point-to- Point Topologie [7]

2.2.2 Fabric Topologie

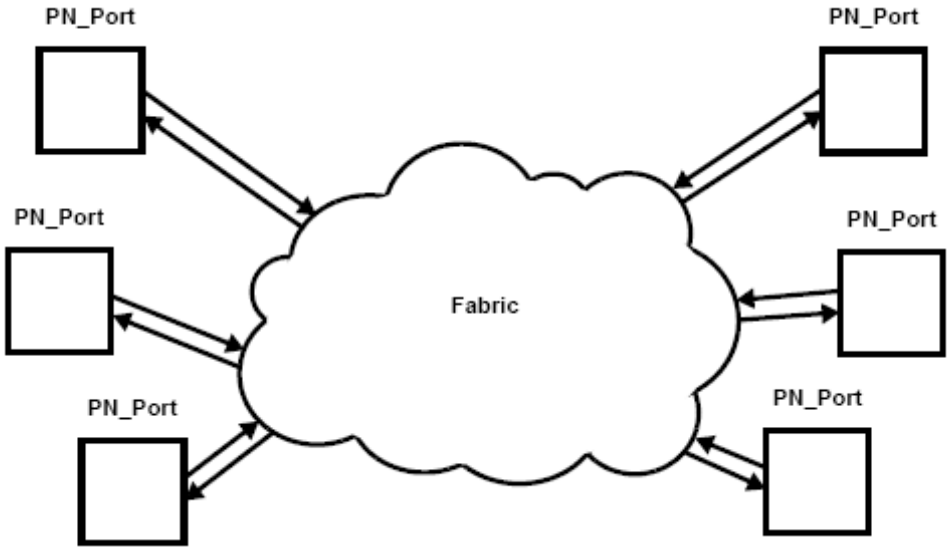

Abbildung 2.2.2: Fabric Topologie [7]

2.2.3 Arbitratet Loop Topologie

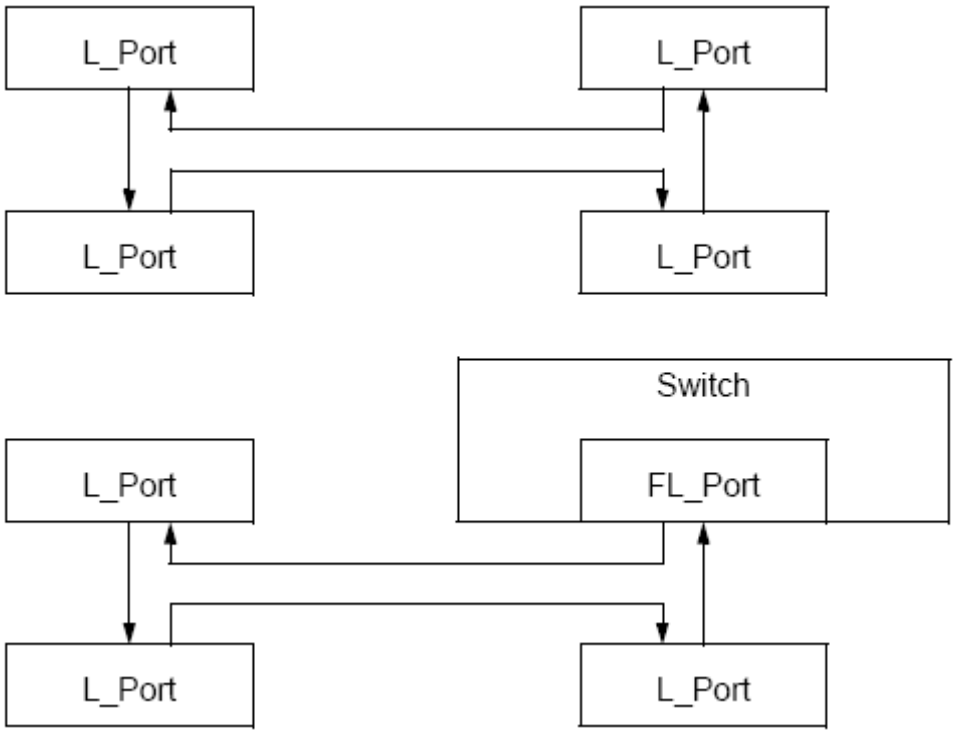

Abbildung 2.2.3: Arbitratet Loop Topologie [7]

Bild 2.2.3 zeigt zwei voneinander unabhängige Arbitrated Loop Konfigurationen. Jede Verbindung zwischen den L-Ports stellt eine Glasfaser Leitung dar. Die obere Abbildung stellt eine Konfiguration dar, die nur aus L-Ports zusammengestellt ist. Die zweite Darstellung zeigt eine Arbitrated Loop bestehend aus einem FL_Port und drei L-Ports. In dieser Zusammenstellung können noch weitere Ports mit einer Switch verbunden sein.

2.3 Struktur

Wie in Bild 2.3.1 zu erkennen ist, sind die Fibre Channel Level hierarchisch aufeinander aufgebaut und beschreiben die Funktionalität der einzelnen Schichten.

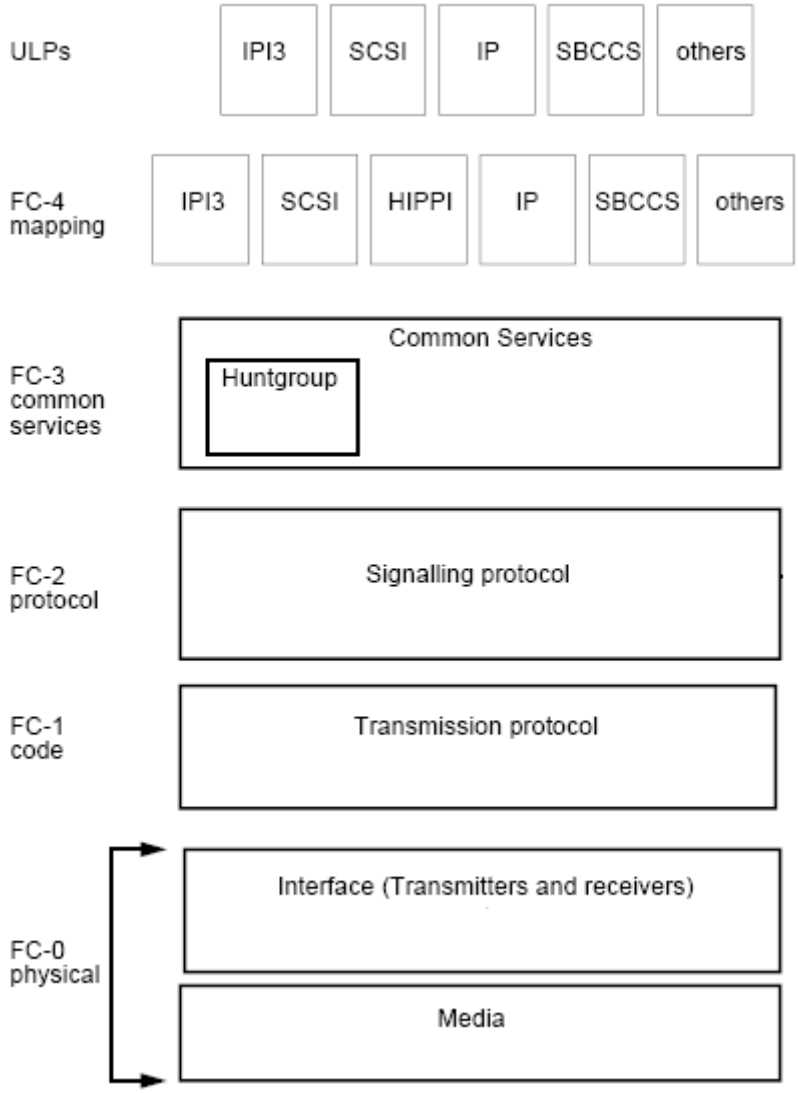

Abbildung 2.3.1: FC Schichtenmodell [7]

FC- 0

Die unterste Schicht FC-0 ist die physikalische Schicht, in der die physikalische Verbindung, die FC-Übertragungsmedien mit ihren Parametern und Anschlüssen sowie die Datenraten spezifiziert sind. Als Übertragungsmedien sind Lichtwellenleiter, Koaxialkabel und STP- Kabel spezifiziert, als Treiber LEDs, Laser und die ECL-Technologie.

FC- 1

Auf der Ebene FC-1 erfolgt die Codierung und Decodierung der zu übertragenden Signale. Fibre- Channel verwendet die 8B/10B-Codierung zur Verbesserung der Übertragungseigenschaften.

FC-2

Auf der Schicht FC-2 findet die Flusskontrolle statt, die mit dem Credit- Verfahren arbeitet und zwei Mechanismen kennt: die Buffer- to- Buffer-Flusskontrolle (BB_Buffer) für die Links und die End-to-End- Flusskontrolle zwischen zwei Endgeräten. Um eine möglichst effiziente Datenübertragung für die unterschiedlichen Verkehrsarten zu gewährleisten, unterscheidet Fibre- Channel mehrere FC- Dienstklassen. Diese werden in der Schicht FC-2 durch die Steuerung der Datensequenzen realisiert. Bei diesen Dienstklassen handelt es sich um bestätigte und unbestätigte verbindungslose Services.

FC- 3

Die Ebene FC-3 unterstützt Services wie Multicast- Sendungen, bei denen eine Information an viele Empfänger gesendet wird, und Striping, bei dem die Bandbreite durch Parallelschaltung von Ports erhöht wird.

FC- 4

FC-4 ist die oberste Protokollschicht und stellt die Schnittstelle den „Upper Level“-Protokollen (ULP), z.B. SCSI, zur Verfügung, um den „Transportservice“ der unteren FC-Schichten zu verwenden. Dabei setzt FC- 4 zwischen den einzelnen ULPs und FC-3 um. Die unteren Schichten müssen „verstehen“, was die Informationseinheiten der unterschiedlichen ULPs zu bedeuten haben und was sie bei dem Empfänger bewirken sollen.

Die am weitesten entwickelte und am häufigsten eingesetzte Schnittstelle zwischen einem ULP und den Transportschichten ist SCSI-3. Für das Host System ist die FC-Funktionalität im FC-Host Bus Adapter (HBA) und in deren Treiber implementiert, so dass das Betriebssystem und der Diskmanager eine FC-Festplatte als ein SCSI adressierbares Speichermedium erkennen. Fibre Channel ist für die Upper Level Protokolle und das Betriebssystem völlig transparent.

2.4 Serviceklassen

Zurzeit werden fünf verschiedene Serviceklassen verwendet bzw. definiert. Serviceklassen sind lediglich unterschiedliche Kommunikationsmethoden, die unter den Teilnehmern verwendet werden.

2.4.1 Klasse 1

Klasse 1 ist wie eine direkte Verbindung zwischen gegenüberliegenden Teilnehmern ohne Unterbrechung. Weitere Eigenschaften einer Klasse 1 Verbindung sind:

- garantierte Lieferung
- Rahmen werden in der gleichen Reihenfolge empfangen, wie sie gesendet wurden

2.4.2 Klasse 2

Klasse 2 ist die verbindungslose Kommunikation zwischen zwei Ports. Mehrere Geräte teilen sich die vorhandene Bandbreite. Daten werden übertragen, sobald Bandbreite vorhanden ist. Der Sender wird sowohl benachrichtigt, wenn die Daten angekommen sind, als auch, wenn sie den Empfänger nicht erreicht haben. Das garantiert eine gewisse Übertragungssicherheit.

2.4.3 Klasse 3

Klasse 3 ist wie Klasse 2, aber ohne Bestätigung des Datentransfers von Ende zu Ende. SCSI wird meist mit dieser Dienstklasse kombiniert. Alle angeschlossenen Geräte teilen sich die Bandbreite und können bei geringem Verkehrsaufkommen mit voller Geschwindigkeit arbeiten. Bei viel Datenverkehr sind allerdings Einschränkungen möglich.

2.4.4 Klasse 4

Klasse 4 ist der Klasse 1 sehr ähnlich. Mit Klasse 4 kann die gesamte Bandbreite nicht genutzt werden, da Teile der Bandbreite anderen angeschlossenen Teilnehmern zugewiesen werden.

2.4.5 Klasse 5

Sie dient der internen Kontrolle und der Koordination der verschiedenen Elemente einer Fabric. Daten können hier nur zwischen Switches ausgetauscht werden. Alle anderen Geräte ignorieren Klasse -4-Daten.

2.5 Exchange und Sequence

Rahmen transportieren alle Informationen zwischen den Ports. Die Konstruktion der Rahmen erfolgt normalerweise beim Sender-Port. Ein Frame stellt die kleinste Informationseinheit dar, die übermittelt wird. Eine Sequenz besteht aus einem oder mehreren Rahmen, wobei ein Exchange wiederum aus einer oder mehreren Sequenzen besteht. Es besteht die Möglichkeit, dass eine Sequenz nur aus einem Rahmen besteht und ein Exchange nur aus einer Sequenz, dies ist aber normalerweise nicht gebräuchlich. In Abbildung 2.5.1 wird der Zusammenhang zwischen Sequenz und Exchange detailliert dargestellt.

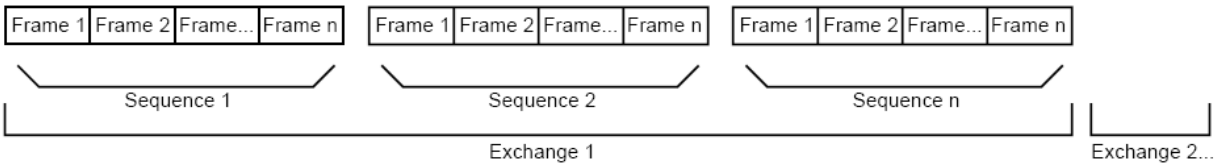

Abbildung 2.5.1: Zusammenhang zwischen Rahmen, Sequenzen und Exchanges [5]

2.6 Rahmenaufbau

Alle FC-2 Rahmen sind wie in Bild 2.6.1 dargestellt aufgebaut. Jeder Rahmen besteht aus einem Start-of-Frame, den Frame-Content oder auch Payload genannt, und dem End-of-Frame. Der Inhalt unterteilt sich wiederum in Extended-Header, Frame-Header, Data-Field und dem CRC.

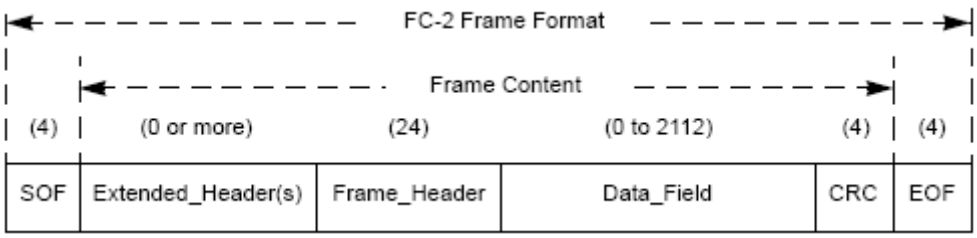

Abbildung 2.6.1: Frameformat [7]

Der SOF- Delimiter zeigt auf den Anfang eines Rahmen und teilt mit, wie viele weitere Rahmen noch folgen. Der Extended-Header ergänzt optional die im Frame-Header angegebene Funktionalität. Dem SOF folgt darauf direkt der Frame-Header, falls kein Extended-Header verwendet wird. Er ist 24 Byte lang und enthält Kontroll- und Adressinformationen über den Frame. Das Datenfeld beinhaltet die zu übertragenen Informationen und hat eine variable Größe von 0 bis 2112 Byte. Dies ist ein Kompromiss zwischen großen Rahmen, die die Multiplexing- Fähigkeit verschlechtern und kleinen Rahmen, die zu viel Overhead generieren. Zudem enthält es einen optionalen Header und 3 Füllbytes. Laut FC-Restriktionen muss der Nutzdatenbereich (Payload) einem Vielfachen von Vier entsprechen. Sollte das nicht der Fall sein, nutzt man die Füllbytes um die Vorschrift zu erfüllen. Der Cyclic Redundancy Check (CRC) testet, ob die Daten des Headers und der Payloads richtig übertragen wurden. Mit dem End-of-Frame Delimiter wird das Ende des Rahmens angegeben. Sollte der Rahmen nicht richtig übertragen worden sein, wird er erneut vom Port angefordert.

2.6.1 SOF

Der Start of Frame Delimiter signalisiert den Anfang eines Rahmens. Tabelle 2.6.1.1 Enthält eine Liste der verschiedenen SOF Delimiter.

Tabelle 2.6.1.1: SOF und EOF Delimiter [5]

Delimiter function	Delimiter	Beginning running disparity	Ordered set (FC-1)	Ordered set (hex)
SOF Connect Class 1	SOFc1	Negative	K28.5 D21.5 D23.0 D23.0	BC B5 17 17
SOF Initiate Class 1	SOFi1	Negative	K28.5 D21.5 D23.2 D23.2	BC B5 57 57
SOF Normal Class 1	SOFn1	Negative	K28.5 D21.5 D23.1 D23.1	BC B5 37 37
SOF Initiate Class 2	SOFi2	Negative	K28.5 D21.5 D21.2 D21.2	BC B5 55 55
SOF Normal Class 2	SOFn2	Negative	K28.5 D21.5 D21.1 D21.1	BC B5 35 35
SOF Initiate Class 3	**SOFi3**	**Negative**	**K28.5 D21.5 D22.2 D22.2**	**BC B5 56 56**
SOF Normal Class 3	**SOFn3**	**Negative**	**K28.5 D21.5 D22.1 D22.1**	**BC B5 36 36**
SOF Initialize Loop	**SOFil**	**Negative**	**K28.5 D21.5 D22.2 D22.2**	**BC B5 56 56**
SOF Activate Class 4	SOFc4	Negative	K28.5 D21.5 D25.0 D25.0	BC B5 19 19
SOF Initiate Class 4	SOFi4	Negative	K28.5 D21.5 D25.2 D25.2	BC B5 59 59
SOF Normal Class 4	SOFn4	Negative	K28.5 D21.5 D25.1 D25.1	BC B5 39 39
SOF Fabric	SOFf	Negative	K28.5 D21.5 D24.2 D24.2	BC B5 58 58
EOF Terminate	**EOFt**	**Negative** **Positive**	**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 Disconnect-Terminate	EOFdt	Negative Positive	K28.5 D21.4 D21.4 D21.4 K28.5 D21.5 D21.4 D21.4	BC 95 95 95 BC B5 95 95
EOF Abort	EOFa	Negative Positive	K28.5 D21.4 D21.7 D21.7 K28.5 D21.5 D21.7 D21.7	BC 95 F5 F5 BC B5 F5 F5
EOF Normal	**EOFn**	**Negative** **Positive**	**K28.5 D21.4 D21.6 D21.6** **K28.5 D21.5 D21.6 D21.6**	BC 95 D5 D5 **BC B5 D5 D5**
EOF Disconnect-Terminate-Invalid	EOFdti	Negative Positive	K28.5 D10.4 D21.4 D21.4 K28.5 D10.5 D21.4 D21.4	BC 8A 95 95 BC AA 95 95
EOF Normal-Invalid	EOFni	Negative Positive	K28.5 D10.4 D21.6 D21.6 K28.5 D10.5 D21.6 D21.6	BC 8A D5 D5 BC AA D5 D5

2.6.2 Header

Die Abbildung 2.6.2.1 zeigt wie sich der Frame-Header unterteilt:

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 2.6.2.1: Header-Frame [7]

R-CTL:

Das R_CTL Feld ist ein Ein Byte großer Bereich im Wort Null, welches Routing Bits und Information Bits enthält, um die Rahmen-Funktion zu bestimmen. Dieses Feld ist daher weiterhin in einen Routingbereich (Bits 31-28) und einen Informationsbereich (Bits 27-24) unterteilt.

Die Tabellen 2.6.2.1 und 2.6.2.2 verdeutlichen die Bedeutungen der beiden Felder.

Tabelle 2.6.2.1und 2.6.2.2: R_CTL Code Bedeutung [7]

R_CTL		Frame type
ROUTING	INFORMATION	
0h	(see table 15)	Device_Data frames (see clause 12)
2h	(see FC-LS)	Extended Link Services (see FC-LS)
3h	(see table 17)	FC-4 Link_Data (see 9.4.3)
4h	(see table 18)	Video_Data (see 9.4.3)
5h	(see table 37)	Extended_Headers (see 10)
8h	(see table 62)	Basic Link Services (see clause 13)
Ch	(see table 52)	Link_Control Frame (see 12.3)
Fh	(see table 19)	Extended Routing (see 9.4.3)
Others	Reserved	Reserved

R-CTL		Description
ROUTING	INFORMATION	
0h	0h	Uncategorized information
	1h	Solicited Data
	2h	Unsolicited Control
	3h	Solicited Control
	4h	Unsolicited Data
	5h	Data Descriptor
	6h	Unsolicited Command
	7h	Command Status
	Others	Reserved

Destination ID

Das D_ID Feld ist drei Byte lang und soll den Adress-Identifier des Ziel-Nx-Ports enthalten.

Source ID

Die S_ID dagegen gibt den Adress-Identifier des Quell-Nx-Ports an

Tabelle 2.6.2.3 zeigt eine Auflistung der bekannten Adress-Identifier

Tabelle 2.6.2.3: Adress-Identifier [7]

Address Value	Description
FF FC 01h to FF FC FEh	Reserved for Domain Controllers
FF FF F0h	N_Port Controller (see FC-LS)
FF FF F1h to FF FF F3h	Reserved
FF FF F4h	Event Service (see FC-GS-6)
FF FF F5h	Multicast Server (see 22.4.4)
FF FF F6h	Clock Synchronization Service (see clause 25)
FF FF F7h	Security Key Distribution Service (see FC-GS-6)
FF FF F8h	Alias Server (see 23.3)
FF FF F9h	Quality of Service Facilitator - Class 4 (QoSF) - Obsolete
FF FF FAh	Management Service (see FC-GS-6)
FF FF FBh	Time Service (see FC-GS-6)
FF FF FCh	Directory Service (see FC-GS-6)
FF FF FDh	Fabric Controller
FF FF FEh	F_Port Controller
FF FF FFh	Broadcast Alias_ID (see 22.5)

CS_CTL

Die Bedeutung dieses Feldes wird durch das CS_CTL / Priority Enable bit (F_CTL, Bit 17) aktiviert oder deaktiviert. Wenn dieses Bit auf Null gesetzt ist werden die Bits 31-24 von Wort Eins als CS_CTL Feld interpretiert. Es enthält Managementinformationen zur Serviceklasse, die im SOF festgelegt wurden. Die Bedeutung des CS_CTL ist abhängig von der Serviceklasse. Wird Bit 17 auf Eins gesetzt, werden die Bits 31-24 in Wort Eins als Priority Informations interpretiert.

Type

Die Datenstruktur Type ist ein Byte groß und im zweiten Wort zu finden.
Sie identifiziert das Protokoll des Datenrahmeninhalts. Sollten die Routing Bits im R_CTL Feld Basic oder Extended Link Service angegeben sein, so werden die Type-Codes wie in Tabelle 2.6.2.4 entschlüsselt.

Tabelle 2.6.2.4 : Type-Codes Link Service [7]

Encoded Value Word 2, bits 31-24	Description
00h	Basic Link Service
01h	Extended Link Service
02h to CFh	Reserved
D0h to FFh	Vendor specific

F_CTL

Das Frame Control Feld ist drei Byte lang und befindet sich im Wort 2, Bits 23 – 0. Es enthält Kontrollinformationen, die sich auf den Rahmeninhalt beziehen. Das Format des Frame Control Feld wird in Tabelle 2.6.2.5 definiert.

Tabelle 2.6.2.5: Exchange / Sequenz Kontrolle [7]

Control Field	Word 2 Bits	Description
Exchange Context	23	0 = Originator of Exchange 1 = Responder of Exchange
Sequence Context	22	0 = Sequence Initiator 1 = Sequence Recipient
First_Sequence	21	0 = Sequence other than first of Exchange 1 = first Sequence of Exchange
Last_Sequence	20	0 = Sequence other than last of Exchange 1 = last Sequence of Exchange
End_Sequence	19	0 = Data frame other than last of Sequence 1 = last Data frame of Sequence
End_Connection (Class 1 or 6)	18	0 = Connection active 1 = End of Connection Pending (Class 1 or 6)
CS_CTL/Priority Enable	17	0 = Word 1, Bits 31-24 = CS_CTL 1 = Word 1, Bits 31-24 = Priority
Sequence Initiative	16	0 = hold Sequence Initiative 1 = transfer Sequence Initiative
X_ID reassigned - Obsolete	15	
Invalidate X_ID - Obsolete	14	
ACK_Form	13-12	00b = No assistance provided 01b = Ack_1 Required 10b = reserved 11b = Ack_0 Required
Data Compression – Obsolete	11	
Data Encryption - Obsolete	10	
Retransmitted Sequence	9	0 = Original Sequence transmission 1 = Sequence retransmission
Unidirectional Transmit (Class 1)	8	0 = Bi-directional transmission (Class 1) 1 = Unidirectional transmission (Class 1)
Continue Sequence Condition	7-6	Last Data frame - Sequence Initiator 00b = No information 01b = Sequence to follow-immediately 10b = Sequence to follow-soon 11b = Sequence to follow-delayed

Control Field	Word 2 Bits	Description
Abort Sequence Condition	5-4	ACK frame - Sequence Recipient 00b = Continue sequence 01b = Abort Sequence, Perform ABTS 10b = Stop Sequence 11b = Immediate Sequence retransmission requested Data frame (1st of Exchange) - Sequence Initiator 00b = Abort, Discard multiple Sequences 01b = Abort, Discard a single Sequence 10b = Process policy with infinite buffers 11b = Discard multiple Sequences with immediate retransmission
Relative offset present	3	0 = Parameter field defined for some frames 1 = Parameter Field = relative offset
Exchange reassembly	2	Reserved for Exchange reassembly
Fill Bytes	1-0	End of Payload - bytes of fill 00b = 0 bytes of fill 01b = 1 byte of fill (first byte following Payload) 10b = 2 bytes of fill (first two bytes following Payload) 11b = 3 bytes of fill (first three bytes following Payload)

Bit 18 ist nur von Bedeutung, wenn Bit 19 auf Eins gesetzt wird.

SEQ_ID

Die Sequence_ID wird vom Sequenz-Initiator vorgegeben und sollte während der geöffneten Sequenz einzigartig für ein bestimmtes D_ID und S_ID Paar sein.

DF_CTL

Das Data Field-Control Feld spezifiziert, ob optionale Header zu Beginn des Data Field vorhanden sind. Dieses Feld befindet sich in Wort Drei von Bit 23 – 16.

Tabelle 2.6.2.6: DF_CTL Bit Definitionen [7]

Word 3, Bit(s)	Optional Header	Applicability
23	Reserved	all frames
22	0 = Neither ESP_Header nor ESP_Trailer 1 = Both ESP_Header and ESP_Trailer	all frames
21	0 = No Network_Header 1 = Network_Header	Device_Data and Video_Data frames
20	0 = No Association_Header 1 = Association_Header	Device_Data and Video_Data frames
19-18	Reserved	all frames
17-16	00b = No Device_Header 01b = 16 Byte Device_Header 10b = 32 Byte Device_Header 11b = 64 Byte Device_Header	Device_Data and Video_Data frames

Die optionalen Header müssen dann im Data Field der Reihe nach platziert sein. Das bedeutet, der Bit 23 Header ist an erster Stelle, gefolgt vom Bit 22 Header usw.

SEQ_CNT

Der Sequence Count gibt die Reihenfolge an, in der die Datenrahmen innerhalb einer einzelnen Sequenz übertragen werden. Dabei sollte der SEQ_CNT des ersten Datenrahmens der ersten Sequenz des Exchanges, der von Transmitter oder Responder gesendet wird, binär Null sein. Jeder weitere SEQ_CNT eines untergeordneten Datenrahmens wird dann um Eins erhöht. Falls der Sequenzzähler 65535 erreicht hat springt er beim nächsten Inkrement wieder auf Null.

OX_ID

Das Originator Exchange ID Feld soll die Exchange_ID identifizieren, die vom Erzeuger des Exchange vergeben wird

RX_ID

Ein vom Absender eröffneter Exchange, wird mit dem 2 Byte großen Responder Exchange Identifier vom Responder gekennzeichnet. Da diese Funktion nicht verwendet wird, steht in diesem Bereich FFFF.

Parameter

In diesem 4 Byte großen Feld kann ein relativer Offset definiert werden, der die Position des ersten Byte in der Payload festlegt. Wenn ein relativer Offset vorhanden ist, ist Bit 3 des F_CTL auf Eins gesetzt, damit er gültig ist.

2.6.3 Data-Field

Das Datenfeld, auch Payload genannt, ist auf die Wortgrenzen hin abgestimmt. Die Länge der Payloads muss ein ganzzahliges Vielfaches von 4 Byte sein und ist auf 2112 Bytes begrenzt. Um diese Auflage zu erfüllen, müssen gültige Füllbytes eingesetzt werden, um immer ein Vielfaches von Vier zu erhalten. F_CTL Bits Null und Eins weisen darauf hin, wie viele Füllbytes verwendet werden. Füllbytes werden immer nur im letzten Rahmen einer Sequenz eingesetzt.

2.6.4 CRC

Für den Cyclic Redundancy Check ist ein vier Byte großes Feld vorgesehen, welches direkt dem Datenfeld folgt. Der CRC wird dazu benutzt, die Vollständigkeit des Header und der Payload zu bestätigen. Dies hilft Fehler im Rahmen aufzuspüren. SOF und EOF sind im CRC nicht mit eingeschlossen.

2.6.5 EOF

Der End-of-Frame Delimiter zeigt das Ende eines Rahmen an. Eine Liste der verschiedenen EOFs befindet sich in Tabelle 2.6.1.1, Kapitel SOF.

3 Der Hostbusadapter [9, 10]

Ein Hostbusadapter, kurz HBA, ist eine Hardwareschnittstelle, die ein Computersystem mit externen oder internen Netzwerk-, Speicher oder anderen Geräten verbindet. Er erweitert die Fähigkeiten um die Hardwarekompatibilität zu einem bestimmten Bussystem. Der HBA ist seinerseits an einem internen Bus des Computers angeschlossen (z. B. dem PCI-Bus) und ermöglicht so den Austausch von Daten mit Geräten oder anderen Computern über mehrere Bus-Systeme hinweg. Einen Host-Bus-Adapter gibt es als optionale Erweiterung (Steckkarte) und auch als Hauptplatinenbestandteil. Der Begriff wird vorrangig im Zusammenhang mit den Bus-Systemen SCSI, ATA (IDE), SATA und Fibre Channel verwendet. Im Prinzip ist auch eine Netzwerkkarte ein HBA.
Zu den bekanntesten Herstellern von HBA- Karten zählen Adaptec, QLogic, JNI, LSI (Engenio), Alacritech und Emulex.

3.1 Fibre Channel HBA der Firma QLogic

Der in dieser Diplomarbeit verwendete Single Port 4-Gbps Fibre Channel to PCI Express HBA ist in Bild 3.1 abgebildet. Seine technischen Daten sind dem Datenblatt im Anhang zu entnehmen.

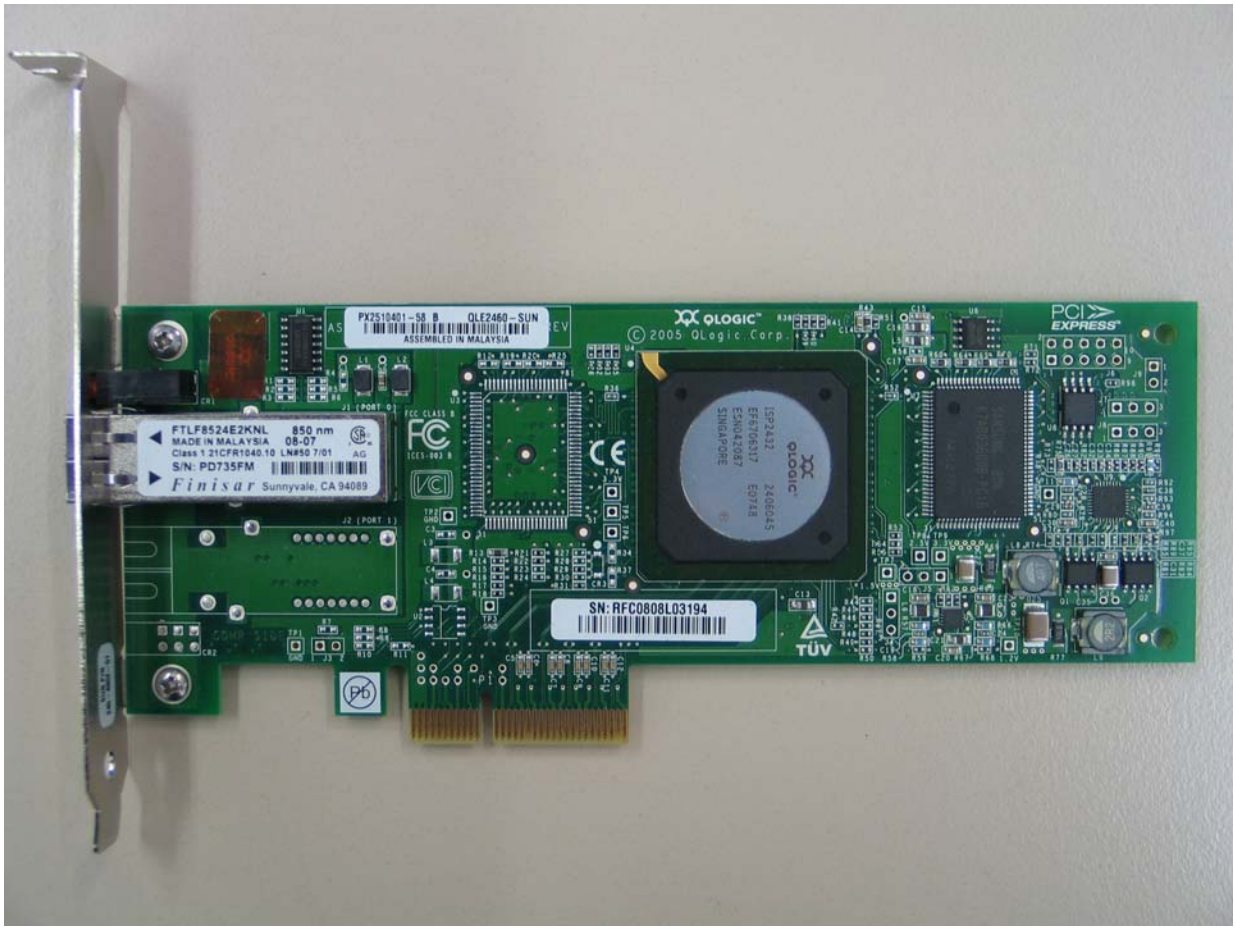

Abbildung 3.1: HBA QLE2460

3.2 HBA LEDs

Auf der Frontplatte des Hostbusadapters sind drei LEDs angebracht , die dem Anwender den aktuellen Betriebszustand der Karte anzeigen. In Tabelle 3.2.1 sind die Bedeutungen der Lampen aufgeführt.

Tabelle 3.2.1: LED Überblick

Gelbe LED	**Grüne LED**	**Orangene LED**	**Aktivität**
○	○	○	Power Off
●	●	●	Power On (vor Firmware Initialisierung)
blinkend	blinkend	blinkend	Power On (nach Firmware Initialisierung)
Abwechselnd blinkend	Abwechselnd blinkend	Abwechselnd blinkend	Firmware Error
○	○	●/ blinkend	Online, 1Gbps link I/O Aktivität
○	●/ blinkend	○	Online, 2Gbps link I/O Aktivität
●/ blinkend	○	○	Online, 4Gbps link I/O Aktivität
blinkend	○	blinkend	Signalisierung

3.3 HBA Manager Tool

Neben der entsprechenden Treibersoftware liefert QLogic auch ein sogenanntes SANsurfer FC HBA Manager Tool. Diese Software stellt eine graphische Benutzeroberfläche dar, mit deren Hilfe man die Karte installieren, konfigurieren und bedienen kann. Das Programm bietet außerdem verschiedene diagnostische Möglichkeiten zur Fehlerbehebung und nützliche Statistiken, um die SAN-Anwendungen zu optimieren. Um einen kleinen Überblick über dieses Werkzeugs zu bekommen werden nun im Folgenden die wichtigsten Anwendungen kurz beschrieben.

3.3.1 Anwendungen

Update management: Um den SANsurfer FC HBA Manager einfach zu aktualisieren, können die Updates sowohl über Internet als auch über CD-ROM vorgenommen werden.

HBA asset management: Mit dem SANsurfer FC HBA Manager ist der Anwender in der Lage sich mit lokalen oder externen Hosts zu verbinden oder von ihnen zu trennen. Man hat außerdem die Möglichkeit Informationen von den angeschlossenen Hosts und deren HBAs und daran angebundene Speichermedien zu erhalten. Zusätzlich wird für einen speziellen Host die momentane SAN Konfigurierung in einer zweidimensionalen Grafik dargestellt.

SAN configuration mangement: Mit dem Managertool hat man die Möglichkeit sowohl lokale als auch Remote Systeme zu konfigurieren.
Des Weiteren ist der Benutzer in der Lage, Hostsysteme zu vergleichen. Das bedeutet, er kann sich die Unterschiede zwischen den aktuellen Einstellungen des Hosts und die Konfigurierungen der gespeicherten Hosts anschauen. Dadurch weis

man, was sich im SAN verändert hat. Zusätzlich erlaubt das Configuration Management die HBA- Parameter und Treiber zu erneuern und abzuspeichern.

3.3.2 Die SANsurfer FC HBA Manager Bedienoberfläche

Wenn das Hauptfenster das erste mal geöffnet wird, wird ein Ampelsymbol angezeigt. Diese Ampel gibt den Status der Hosts an. In Abbildung 3.3.2.1 ist die Bedeutung der Ampel aufgezeigt.

Signal	Indicates
 Good	All hosts in the system are up and running.
 Warning	One or more hosts in the system are in a warning state. For instance, a loop may be down on an HBA in one of the hosts in the system.
 Bad	All hosts in the system are down.

Abbildung 3.3.2.1: Ampelzeichen [10]

Das HBA Manager Fenster zeigt eine HBA Baumstruktur, Tool tabs, eine Titel-, Werkzeug und Menüleiste, sowie Registerkarten und eine Statuszeile.
Abbildung 3.3.2.2 zeigt die Aufteilung des Hauptfensters.

Abbildung 3.3.2.2: Hauptfenster [10]

In der folgenden Tabelle 3.3.2.1 werden die Funktionen der am häufigsten benutzten Werkzeugleistenknöpfe kurz beschrieben.

Tabelle 3.3.2.1: Toolbar Knöpfe

	Durch Drücken der Connect Taste wird der SANsurfer FC HBA Manager mit einem speziellen Host oder einer IP-Adresse verbunden

	Durch Betätigung dieses Knopfes wird die Fibre Channel Port Configuration dialog box geöffnet, wo man HBAs auf dem eigenen Host konfigurieren kann
	Das Drücken dieser Taste hat zur Folge, dass eine Auflistung aller Application Events erscheint. Hier können alle Ereignisse, die auf diesem Host vorkommen, wieder angesehen und die Application Events Informationen verwaltet werden.
	Das Drücken dieser Taste hat zur Folge, dass eine Auflistung aller HBA Ereignisse erscheint. Hier können alle Events, die auf diesem Host vorkommen wieder angesehen und die HBA Events Informationen verwaltet werden.
	Speichert die folgenden Dateien automatisch in eine zip- Datei: - Host in eine Textdatei - Host in eine .qlc Datei - Application Event Liste in eine Textdatei - HBA Event Liste in eine Textdatei
	Öffnet die HBA Statistics dialog box und gibt die verschiedenen Funktionen des HBA auf dem Bildschirm aus.
	Öffnet die HBA Diagnostics dialog box und ermöglicht die Anwendung der zwei Diagnosetests Loopback und Read/Write
	Alle Informationen der an den Host angeschlossenen HBAs werden durch Drücken dieser Taste erneuert.

Der HBA- Baum zeigt dem Benutzer den Hostnamen und den Hoststatus an. Der Status wird mit Live oder Offline gekennzeichnet. Unter dem Namen kann auch die IP-Adresse angegeben sein. Des Weiteren enthält die Baumanzeige Informationen über die angeschlossen HBAs., wie zum Beispiel das HBA Model. In diesem Fall ist das ein QLE2460. HBA alias ist ein benutzerdefinierter Name für die Karte. Loop down bedeutet, dass der Adapter nicht synchronisiert ist. Mögliche Ursachen für diesen Fehler können ein nicht angeschlossenes Kabel oder ein falsch eingestellter Verbindungsmodus sein. HBA Status weist mit bad alarm, loop down, warning alarm oder good auf den Status des HBA hin.
Neben den Host und HBA Informationen werden im HBA Tree aber auch weitere Informationen über den HBA Port angezeigt, wie zum Beispiel Port Aliasnamen, Portnummer und Portstatus.
Sollten verschiedene Speichergeräte wie Disks, Tapes oder weitere am System angeschlossen sein, so werden auch hier dem Benutzer einige Daten in der Baumstruktur übergeben. Die wichtigsten sind hier der Gerätetyp (Disk, Tape), die Geräte Port ID mit Angabe der Port ID des Speichermediums und die Online/ Inaktive Statusanzeige. Weiterhin erkennt man, ob die Diagnoseeigenschaften des angeschlossenen Speichers aktiviert oder ausgeschaltet, bzw. nicht angeboten werden.

3.3.3 Vorgenommene Einstellungen

Im Folgenden werden die benutzerdefinierten Einstellungen aufgezeigt, mit denen die Kommunikationsverbindung aufgebaut wurde.
Durch Doppelklicken auf den Port Eins im HBA Baum wurden unter der Karteikarte Parameter der Verbindungstyp auf Point to Point, die Datenrate zunächst auf 1Gbps und die Rahmengröße auf 512 Byte eingestellt. Alle weiteren Einstellungen auf dieser Seite sind voreingestellt.

3.3.4 Diagnose Tests

Wie bereits weiter vorne in dieser Ausarbeitung erwähnt, bietet der Sansurfer FC HBA Manager zwei Diagnosemöglichkeiten. Zum einen den Loopback- Test und zum anderen den Read/Write Buffer Test. Das entsprechende Diagnosefenster kann über den Diags-Knopf in der Toolbar erreicht werden.

Vorbereitungen für den Loopback-Test:
Bevor dieser Test gestartet werden kann, muss in der Registerkarte Parameters die Verbindungsoption von Point-to-Point in Loop Only geändert werden. Danach muss ein Loopbackadapter mit dem HBA Port verbunden werden und anschließend kann der Test im Diagnosefenster gestartet werden.

Vorbereitungen für den Read/ Write Buffertest:
Diese Diagnoseart wird für das Testen von an das System angebundenen Speichermedien angewendet. Er sendet spezielle Daten mit Hilfe des SCSI Write Buffer Kommandos zu einem Zielgerät. Dieses Gerät liest die gesendeten Daten mit dem SCSI Read Buffer Kommando ein und überprüft die Informationen auf Fehler. Der Test vergleicht außerdem den Linkstatus vor und nach der Datenübertragung. Sollte ein Fehler erkannt werden, weist das Ergebnis auf eine unterbrochene oder unzuverlässige Verbindung zwischen HBA Port und Gerät hin.
Zum Starten der Überprüfung muss mindestens eine Disk oder ähnliches vorhanden sein. Dies lässt sich durch Betrachten des HBA Baums leicht erkennen.
Durch Rechtsklicken auf einen HBA Port eröffnet sich eine Menüleiste, in der man den Punkt Device Diagnostics aktiviert und alle Geräte auswählt.

Zurück im Diagnosefenster werden die Testkonfigurierungen wie folgt vorgenommen und anschließend getestet:

- Data Pattern: z.B. FF
- Data Size: 512 Bytes
- Number of Tests: wird durch fortlaufenden Test auf N/A gesetzt
- Test Increment: 1000
- On Error: Ignore
- Test continuously: aktiviert

Bild 3.3.4.1 zeigt diese Einstellung

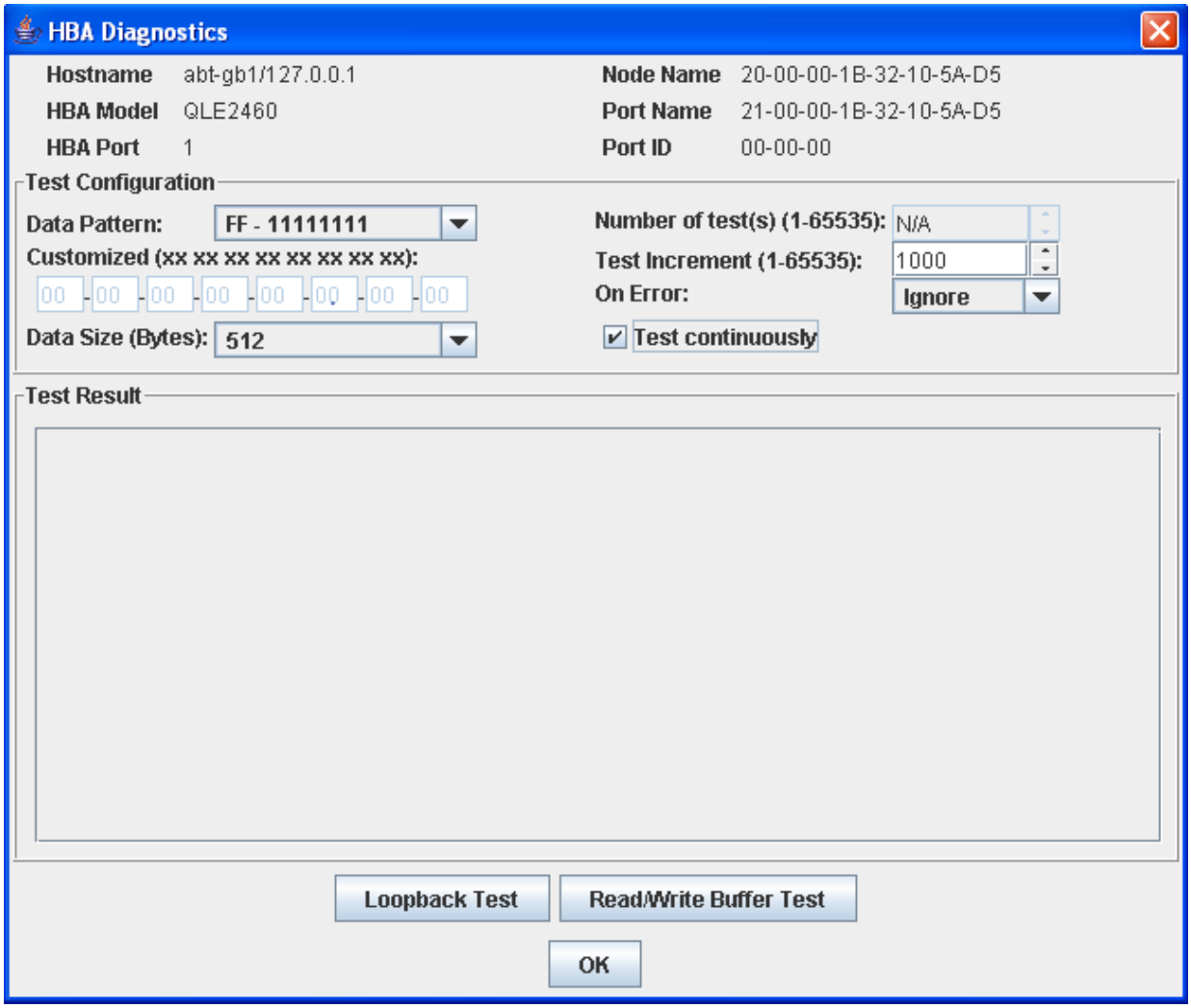

Abbildung 3.3.4.1: Read/Write Buffer Testeinstellungen

4 Testprogramm [1, 11]

Da für den HBA eine entsprechende Testsoftware geschrieben werden sollte, war es nötig, sich zuerst mit der mitgelieferten API zu beschäftigen. API steht dabei für Application Programming Interface und ist eine Programmierschnittstelle, die von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. Neben dem Zugriff auf Datenbanken, der Festplatte oder der Grafikkarte kann eine API auch das Erstellen von Komponenten der grafischen Benutzeroberfläche ermöglichen oder vereinfachen. Im weiteren Sinne wird die Schnittstelle jeder Bibliothek (Library) als API bezeichnet. Auch von QLogic wird eine solche Fibre Channel HBA API Shared Library für Windows angeboten.
Die Tabellen 4.1 bis 4.6 zeigen, welche Funktionalität die Programmbibliothek zur Verfügung stellt.

Tabelle 4.1: HBA API Attribute [11]

Attribute	Unterstützung
HBA_GetVersion	Ja
HBA_LoadLibrary	Ja
HBA_FreeLibrary	Ja
HBA_RegisterLibrary	Ja
HBA_GetNumberOfAdapters	Ja
HBA_GetAdapterName	Ja
HBA_OpenAdapter	Ja
HBA_CloseAdapter	Ja
HBA_GetAdapterAttributes	Ja
HBA_GetAdapterPortAttributes	Ja
HBA_GetPortStatistics	Ja
HBA_GetDiscoveredPortAttributes	Ja
HBA_GetPortAttributesByWWN	Ja
HBA_SendCTPassThru	Ja
HBA_RefreshInformation	Ja
HBA_ResetStatistics	Ja
HBA_GetFcpTargetMapping	Ja
HBA_GetFcpPersistentBinding	Nein

HBA_GetEventBuffer	Ja
HBA_SetRNIDMgmtInfo	Ja
HBA_GetRNIDMgmtInfo	Ja
HBA_SendRNID	Ja
HBA_SendScsiInquiry	Ja
HBA_SendReportLUNs	Ja
HBA_SendReadCapacity	Ja
HBA_RegisterLibraryv2	Ja
HBA_OpenAdapterByWWN	Ja
HBA_GetFcpTargetMappingV2	Ja
HBA_SendCTPassThruV2	Ja
HBA_RefreshAdapterConfiguration	Nein
HBA_GetBindingCapability	Nein
HBA_GetBindingSupport	Nein
HBA_SetBindingSupport	Nein
HBA_SetPersistentBindingV2	Nein
HBA_GetPersistentBindingV2	Nein
HBA_RemovePersistentBinding	Nein
HBA_RemoveAllPersistentBindings	Nein
HBA_SendRNIDV2	Ja
HBA_ScsiInquiryV2	Ja
HBA_ScsiReportLUNsV2	Ja
HBA_ScsiReadCapacityV2	Ja
HBA_GetVendorLibraryAttributes	Ja
HBA_RemoveCallback	Ja
HBA_RegisterForAdapterAddEvents	Nein
HBA_RegisterForAdapterEvents	Nein
HBA_RegisterForAdapterPortEvents	Nein
HBA_RegisterForAdapterPortStatEvents	Nein
HBA_RegisterForTargetEvents	Nein
HBA_RegisterForLinkEvents	Ja
HBA_SendRPL	Ja
HBA_SendRPS	Ja
HBA_SendSRL	Nein
HBA_SendLIRR	Nein
HBA_GetFC4Statistics	Ja
HBA_GetFCPStatistics	Nein

Tabelle 4.2: Adapter Attribute [11]

Manufacturer	Ja
Serial Number	Ja
Model	Ja
Model Description	Ja
Node WWN	Ja
Node Symbolic Name	Nein
Hardware Version	Nein
Driver Version	Ja
Option ROM Version	Ja
Firmware Version	Ja
Vendor Specific ID	Nein
Number Of Ports	Ja
Driver Name	Ja

Tabelle 4.3: Adapter Port Attribute [11]

Node WWN	Ja
Port WWN	Ja
Port FcId	Ja
Port Type	Ja
Port State	Ja
Supported Class of Service	Ja
Supported Fc4Types	Ja
Active Fc4Types	Ja
Port Supported Speed	Ja
Port Speed	Ja
Port Max Frame Size	Ja
Port Symbolic Name	Nein
OS Device Name	Ja
Number of Discovered Ports	Ja
Fabric Name	Nein

Tabelle 4.4: Discovered Port Attribute [11]

Node WWN	Ja
Port WWN	Ja
Port Symbolic Name	Nein
Port FcId	Ja
Port Type	Nein
Port State	Nein
Supported Class of Service	Nein
Port Supported Fc4Types	Nein
Port Active Fc4Types	Nein
Port Supported Speed	Nein
Port Speed	Nein
Port Max Frame Size	Nein
OS Device Name	Ja
Number of Discovered Ports	Nein
Fabric Name	Nein

Tabelle 4.5: Port Statistiken [11]

Seconds Since Statistics Reset	Nein
TxFrames	Nein
RxFrames	Nein
TxWords	Nein
RxWords	Nein
LIPCount	Nein
NOSCount	Nein
ErrorFrames	Nein
DumpedFrames	Nein
LinkFailureCount	Ja
LossOfSyncCount	Ja
LossOfSignalCount	Ja
PrimitiveSeqProtocolErrCount	Ja
InvalidTxWordCount	Ja
InvalidCRCCount	Ja

Tabelle 4.6: FC4 Statistiken [11]

InputRequest	Ja
OutputRequests	Ja
ControlRequests	Ja
InputMegabytes	Ja
OutputMegabytes	Ja

4.1 Hersteller spezifische Funktionen

4.1.1 HBA_GetVersion:

Funktion:	HBA_UINT32 HBA_GetVersion();
Verwendungszweck:	Gibt die Version der SNIA API zurück
Input Parameter:	Keine
Output Parameter:	Keine
Rückgabewert:	Version der SNIA API

4.1.2 HBA_LoadLibrary:

Funktion:	HBA_STATUS HBA_LoadLibrary();
Verwendungszweck:	Lädt alle anbieterspeziefischen Bibliotheken
Input Parameter:	Keine
Output Parameter:	Keine
Rückgabewert:	HBA_STATUS_OK
	HBA_STATUS_ERROR
	HBA_STATUS_ERROR_BUSY

HBA_STATUS_OK wird nach einer erfolgreichen Ausführung des Aufrufs zurückgegeben.

HBA_STATUS_ERROR wird ausgegeben, wenn ein interner Bibliotheks- oder Systemfehler erkannt wurde.

HBA_STATUS_ERROR_BUSY erscheint, wenn die Library gerade beschäftigt ist und der Funktionsaufruf dadurch nicht ausgeführt werden kann.

4.1.3 HBA_FreeLibrary:

Funktion:	HBA_STATUS HBA_FreeLibrary
Verwendungszweck:	Gibt alle zugewiesenen Ressourcen, die für internen Gebrauch während eines HBA_LoadLibrary() Aufrufs reserviert waren, wieder frei.
Input Parameter:	Keine
Output Parameter:	Keine
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY

4.1.4 HBA_GetNumberOfAdapters:

Funktion:	HBA_UINT32 HBA_GetNumberOfAdapters();
Verwendungszweck:	Zeigt die totale Anzahl der am System angeschlossenen Adapter an.
Input Parameter:	Keine
Output Parameter:	Keine
Rückgabewert:	32- Bit großer Unsigned- Integer Wert aller mit dem System verbundenen Fibre Channel Adapter

Bemerkung:

Ein Wert von 0 kann aus folgenden Gründen zurückgegeben werden:

- ein interner Fehler ist aufgetreten
- die Bibliothek ist beschäftigt
- die Anzahl der Adapter ist gleich Null

4.1.5 HBA_GetAdapterName:

Funktion:	HBA_STATUS HBA_GetAdapterName(HBA_UINT32 adapter_index, char *adapter_name);
Verwendungszweck:	Gibt den Namen des Adapters zurück, der durch adapter_index Bestimmt wird
Input Parameter:	adapter_index- Index des entsprechenden Adapters. Der gültige Bereich dieses Werts geht von Null bis Anzahl der Adapter minus Eins
Output Parameter:	adapter_name- Pointer zu einem Buffer, der den Namen des Adapters als String zurückgibt. Die Größe des Buffers muss 256 Byte betragen.
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY HBA_STATUS_ERROR_ARG

Bemerkung:

HBA_STATUS_ERROR_ARG wird zurück gegeben, wenn der Pointer dieser Funktion NULL ist.

4.1.6 HBA_OpenAdapter:

Funktion:	HBA_HANDLE HBA_OpenAdapter(char *adapter_name);
Verwendungszweck:	Öffnet eine Adapter Instanz, während weitere Adapter von der Bibliothek verwaltet werden
Input Parameter:	adapter_name – beliebiger Name der vom HBA_GetAdapterName() Aufruf zurückgegeben wurde
Output Parameter:	keine
Rückgabewert:	Handle zu dieser geöffneten Instanz. Dieser Handle-Wert ist einzigartig für jeden Adapter, der sich im System befindet. Sollte irgendein Fehler auftreten, wird eine Null zurück gegeben.

4.1.7 HBA_CloseAdapter:

Funktion:	void HBA_CloseAdapter (HBA_HANDLE handle);
Verwendungszweck:	Schließt den Adapter, mit dem dazu gehörigen Input-Handle
Input Parameter:	handle – Zugriff auf einen HBA, der durch einen HBA_OpenAdapter () Aufruf bereits hergestellt wurde
Output Parameter:	keine
Rückgabewert:	keine

4.2 Informative Funktionen

4.2.1 HBA_GetAdapterAttributes:

Funktion:	HBA_STATUS HBA_GetAdapterAttributes(HBA_HANDLE handle, HBA_ADAPTERATTRIBUTES *hba_attributes);
Verwendungszweck:	Gibt die Adapterport- Attribute zurück
Input Parameter:	handle – Zugriff auf einen HBA, der durch einen HBA_OpenAdapter () Aufruf bereits hergestellt wurde
Output Parameter:	hba_attributes- Pointer auf eine Struktur, die die Attribute zurück gibt
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY HBA_STATUS_ERROR_ARG HBA_STATUS_ERROR_UNAVAILABLE HBA_STATUS_ERRROR_INVALID_HANDLE

Bemerkung:
HBA_STATUS_ERRROR_INVALID_HANDLE wird zurückgegeben, wenn der übermittelte Handle zu dieser Funktion ungültig ist.

HBA_STATUS_ERROR_UNAVAILABLE wird zurückgegeben, wenn der Adapter zu dem dazugehörigen Handle nicht erreichbar ist.

4.2.2 HBA_GetAdapterPortAttributes:

Funktion:	HBA_STATUS HBA_GetAdapterPortAttributes (HBA_HANDLE handle, HBA_UINT32 port_index, HBA_PORTATTRIBUTES *port_attributes);
Verwendungszweck:	Gibt die Adapterport- Attribute zurück
Input Parameter:	handle – Zugriff auf einen HBA, der durch einen HBA_OpenAdapter () Aufruf bereits hergestellt wurde port_index – dazugehöriger Portindex
Output Parameter:	port_attributes- Pointer auf eine Struktur, die die Portattribute zurück gibt
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY HBA_STATUS_ERROR_ARG HBA_STATUS_ERROR_UNAVAILABLE HBA_STATUS_ERRROR_INVALID_HANDLE HBA_STATUS_ERROR_ILLEGAL_INDEX

Bemerkung:

HBA_STATUS_ERROR_ILLEGAL_INDEX wird zurückgegeben, wenn der Portindex ungültig ist.

4.2.3 HBA_GetPortStatistics:

Funktion:	HBA_STATUS HBA_GetPortStatistics (HBA_HANDLE handle, HBA_UINT32 port_index, HBA_PORTSTATISTICS *port_statistics);
Verwendungszweck:	Erfasst Statistiken für einen zugewiesenen Port
Input Parameter:	handle – Zugriff auf einen HBA, der durch einen HBA_OpenAdapter () Aufruf bereits hergestellt wurde port_index – dazugehöriger Portindex
Output Parameter:	port_statistics – Pointer auf eine Struktur, die die Portstatistiken zurück gibt
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY HBA_STATUS_ERROR_ARG HBA_STATUS_ERROR_UNAVAILABLE HBA_STATUS_ERRROR_INVALID_HANDLE HBA_STATUS_ERROR_ILLEGAL_INDEX

4.2.4 HBA_GetDiscoveredPortAttributes

Funktion:	HBA_STATUS HBA_GetDiscoveredPortAttributes(HBA_HANDLE handle, HBA_UINT32 port_index, HBA_UINT32 discovered_port_index, HBA_PORTATTRIBUTES *port_attributes);
Verwendungszweck:	Gibt die Attribute eines Remote Ports zurück
Input Parameter:	handle – Zugriff auf einen HBA, der durch einen HBA_OpenAdapter () Aufruf bereits hergestellt wurde port_index – dazugehöriger Portindex discovered_port_index – Index des erkannten Ports, dessen Attribute abgerufen werden sollen
Output Parameter:	port_attributes- Pointer auf eine Struktur, die die Portattribute zurück gibt
Rückgabewert:	HBA_STATUS_OK HBA_STATUS_ERROR HBA_STATUS_ERROR_BUSY HBA_STATUS_ERROR_ARG HBA_STATUS_ERROR_UNAVAILABLE HBA_STATUS_ERRROR_INVALID_HANDLE HBA_STATUS_ERROR_ILLEGAL_INDEX

4.3 Programmablauf

Um die in Kapitel 4.2 beschriebenen Funktionen im Testprogramm einbinden zu können, musste zuerst der SCSI/STOR Miniport Treiber Version 8.2.0 oder höher installiert sein. In diesem Fall wurde die derzeit aktuelle Treiberversion 9.1.4.10 installiert und die hbaapi.h Datei in der C- Anwendung implementiert. Die Programmausgabe lässt sich dann im Debugmodus anschauen. Zunächst wird mit der Funktion HBA_GetVersion() die Versionsnummer der aktuellen Programmbibliothek erfasst und am Bildschirm ausgegeben. Der weitere Ablauf des Testprogramms muss dann wie im Folgenden beschrieben, aufgebaut sein. Mit dem Befehl HBA_LoadLibrary() wird die im System installierte HBA API Bibliothek geladen und initialisiert. Ist dabei kein Fehler aufgetreten, gibt die Funktion HBA_STATUS_OK zurück und der nächste Schritt kann eingeleitet werden. Das bedeutet, dass die Anzahl aller an das Netzwerk angeschlossenen HBAs mit Hilfe von HBA_GetNumberOfAdapters() ermittelt und angezeigt wird. Dabei muss mindestens ein Adapter vorhanden sein, damit keine Fehlermeldung auftritt. Nachdem dieser Funktionsaufruf abgearbeitet ist, muss diesem ein HBA_GetAdaptername() Befehl folgen. Damit werden den Geräten fortlaufende Indizes , bei Index Null beginnend, vergeben und den Adapternamen zugeordnet. Die einzelnen Namen werden dann zwischengespeichert. Erst wenn auch dieser Befehl fehlerfrei ausgeführt ist, kann unter Verwendung der Adapternamen jeder beliebige HBA mit HBA_OpenAdapter() für weitere Einstellungen angesprochen werden. Für diese Diplomarbeit wurde nur ein HBA verwendet, daher wird auch nur ein OpenAdapter- Aufruf benötigt. Anschließend daran können beliebige, natürlich nur von der Programmierschnittstelle unterstütze, Funktionen aufgerufen werden. Eine Ausnahme bilden hier allerdings die beiden Befehle HBA_FreeLibrary() und HBA_LoadLibrary(). FreeLibrary darf nur zum Programmende verwendet werden, denn damit werden sämtliche Programmfunktionalitäten der Bibliothek wieder freigegeben. LoadLibrary hingegen darf nur zum Programmaufruf eingesetzt werden. Das Testprogramm gibt hier die Ergebnisse der Funktionen HBA_GetAdapterAttributes(),

HBA_GetAdapterPortAttributes() und HBA_GetPortStatistics() am Bildschirm aus. Nachdem diese Befehle bearbeitet sind, wird der HBA mit einem HBA_CloseAdapter() wieder geschlossen. Wichtig ist, dass jeder geöffnete Adapter auch wieder mit einem CloseAdapter- Befehl geschlossen wird. Das Ende des Testprogramms wird mit dem HBA_FreeLibrary Befehl abgeschlossen. Die Abbildung 4.3.1 zeigt den Programmablauf des Testprogramms und Bild 4.3.2 zeigt die Bildschirmausgabe des laufenden Programms.

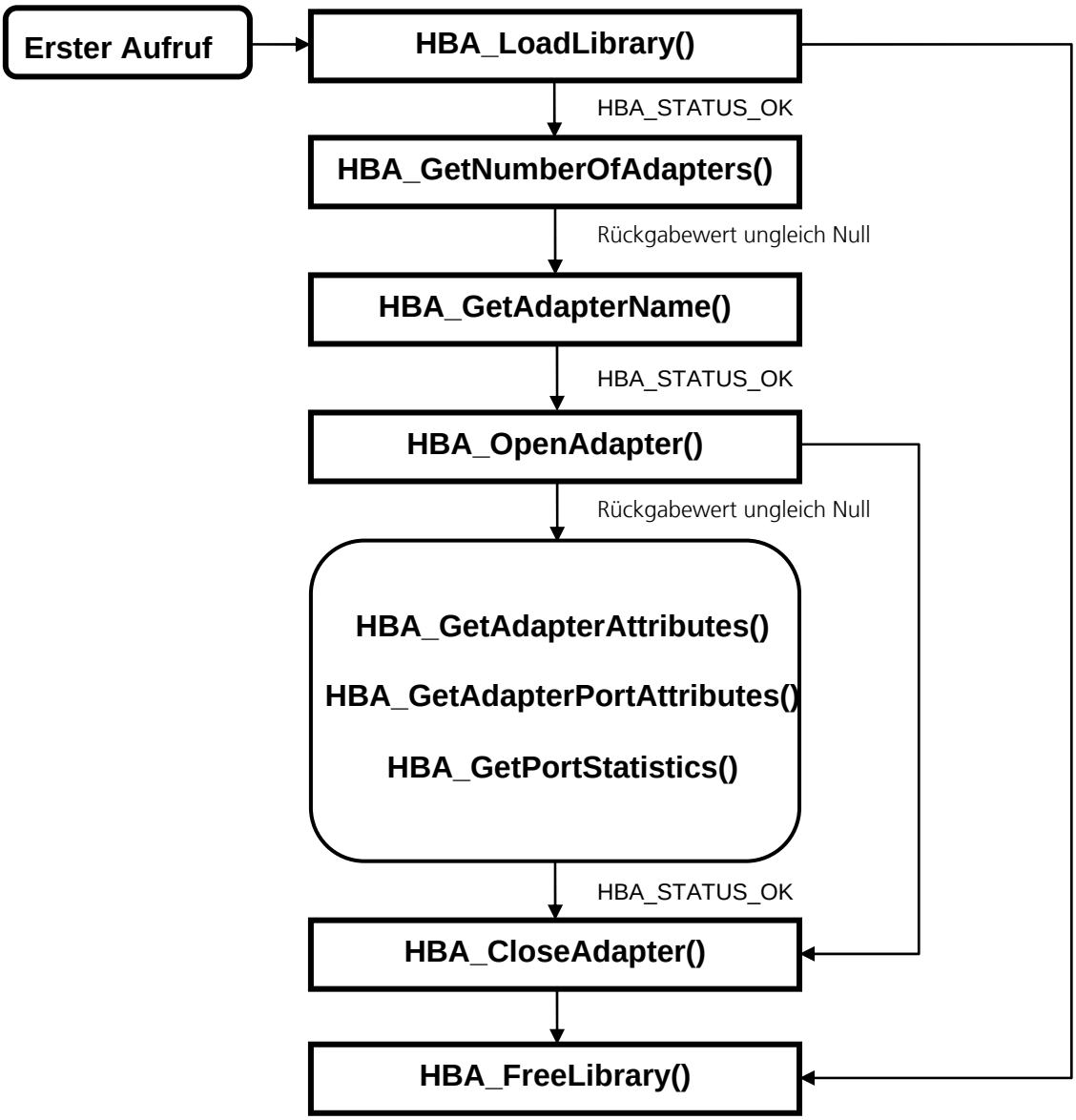

Abbildung 4.3.1: Programmablaufplan

Abbildung 4.3.2: Bildschirmausgabe

5 SCSI Protokoll [3, 5]

5.1 Überblick

Das **S**mall **C**omputer **S**ystem **I**nterface ist eine standardisierte parallele Schnittstelle für die Verbindung und Datenübertragung zwischen Peripheriegeräten und dem Computer-Bus. Im Vergleich zu ATA/ATAPI ist ein wesentliches Merkmal von SCSI die Möglichkeit, mehr als zwei Geräte anschließen zu können. Um an einen Computer SCSI-Geräte anschließen zu können, wird ein SCSI-Host-Bus-Adapter (kurz HBA) benötigt, der den Datentransfer auf dem SCSI-Bus kontrolliert. Das anzuschließende SCSI-Gerät besitzt einen SCSI-Controller, um die Daten über den Bus zu übertragen und mit dem Host Bus Adapter zu kommunizieren. Der SCSI-Host-Bus-Adapter kann auf der Hauptplatine integriert sein, wird aber in der Regel als Steckkarte nachgerüstet. SCSI wird meist zur Anbindung von Festplatten und Bandlaufwerken genutzt, wird jedoch auch mit einer Reihe von weiteren Geräten verwendet, wie zum Beispiel Scannern und optischen Laufwerken. Der SCSI-Standard ist geräteunabhängig ausgelegt, so dass theoretisch jedes Peripheriegerät SCSI benutzen kann. Vereinzelt wird SCSI auch in der industriellen Steuerungstechnik eingesetzt. SCSI wurde über die Jahre weiterentwickelt.
Folgende Standards (in chronologischer Reihenfolge) sind definiert:

- SCSI-1 (1986)
- SCSI-2 (1989)
- Ultra-SCSI (1992)
- SCSI-3 (1993)
- Ultra-2 SCSI (1997)
- Ultra-160 (1999)
- Ultra-320 (2002)
- SCA

Generell sind SCSI-Geräte abwärtskompatibel, das heißt, es ist möglich, eine Ultra-3-Festplatte an einen Ultra-2-Host-Bus-Adapter anzuschließen und zu benutzen (allerdings mit reduzierter Geschwindigkeit und ohne spezifische Ultra-3-Befehle). Jedes SCSI-Gerät (einschließlich des Host Bus Adapters) muss mit einer eindeutigen ID- Nummer konfiguriert werden. Dem Host Bus Adapter bzw. Controller wurde die ID=7 generell zugeordnet. So werden die einzelnen Geräte auf dem SCSI-Bus eindeutig identifiziert und die Priorität der Geräte festgelegt. Die Priorität der IDs lautet in absteigender Reihenfolge 6 bis 0 und dann 15 bis 8. Eventuell bestehen Einschränkungen seitens BIOS oder Betriebssystem bei der Vergabe der ID-Nummern. Jedes Gerät mit einer ID hat darunter zusätzlich mindestens eine LUN (Logical Unit Number) konfiguriert. In der Vergangenheit war SCSI auf allen Arten von Computern weit verbreitet. Für Hochleistungs-Workstations, Server und High-End-Peripherie gilt dies auch heute noch. Desktop-Computer und Notebooks nutzen allerdings überwiegend die langsamere ATA- bzw. (seit etwa 2004) Serial ATA-Schnittstelle für ihre Laufwerke und USB (USB nutzt SCSI-ähnliche Kommandos für einige Operationen) für andere Geräte, da diese Schnittstellen, obwohl sie weniger allgemein verwendbar sind, in der Implementierung weniger kosten. Die ursprünglichen SCSI-Standards spezifizierten die physikalischen Eigenschaften der Busse und die elektrische Signalisierung sowie einen Befehlssatz, der die unterschiedlichen Kommandos definierte, die die SCSI-Geräte ausführen konnten. Dieser Befehlssatz ist auch unabhängig vom SCSI-Bus sehr nützlich, da er ausgereift ist und es eine große Zahl von damit vertrauten Benutzern und Entwicklern gibt. Daher tauchen Teile des SCSI-Befehlssatzes auch in anderen Standards wie ATAPI, Fibre Channel, Serial Storage Architecture, InfiniBand, iSCSI, USB, IEEE 1394 und Serial Attached SCSI auf.

SCSI Informationen werden in Fibre Channel Containern (Rahmen) transportiert. Alle SCSI Fibre Channel Protokoll (SCSI-FCP) Vorgänge beginnen mit einem FCP CMND (Command) und enden mit einem FCP RSP (Response).Vorgänge, die durch FCP CMND eingeleitet werden, können zum Beispiel SCSI Kommandos wie read data oder write data sein.
SCSI-FCP ist ein FC-4 mapping Protokoll, welches beschreibt, wie die SCSI Kommandos im Fibre Channel Protokoll eingebunden werden. Dieses Protokoll behält während jeder I/O Operation die halb duplex Struktur von parallelem SCSI bei. Zum Beispiel arbeitet eine Single Operation wie Read Command über ein einfaches Portpaar zwischen Ziel und Initiator.

Grundsätzlich gibt es im SCSI- Protokoll vier Rahmenformate, die im Folgenden näher beschrieben werden.

5.2 FCP CMND

Alle SCSI FC Protokoll Vorgänge beginnen mit einem FCP_CMND Kommandorahmen. Read data und write data können zum Beispiel solche SCSI Kommandos sein. Zu erwähnen ist an dieser Stelle das R_CTL Feld im Header, welches durch die 06h immer einen solchen Rahmen kennzeichnet. Des Weiteren muss das F_CTL Feld auf 290000h eingestellt werden, was bedeutet, dass der Rahmen die erste Sequenz des Exchange ist und der letzte Frame der Sequenz.
Der Inhalt des Kommandorahmens setzt sich wie in Abbildung 5.2.1und 5.2.2 gezeigt zusammen:

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 5.2.1: FCP CMND frame header [5]

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								
30	DL							
31								(LSB)

Abbildung 5.2.2: FCP_CMND Payload [5]

An dieser Stelle soll nur auf die wichtigsten Felder des Payloads eingegangen werden. Die Logical Unit Number (LUN) ist die physikalische oder virtuelle Adresse eines Zielgerätes. Das Read Data bzw. Write Data Feld gibt an, ob ein Datenaustausch zum Initiator hin oder vom Initiator weg gesendet werden soll. Durch eine Eins im entsprechenden Feld wird die Funktion aktiviert.

Der CDB (Command Descriptor Block) ist immer 16 Byte groß und sein aktueller Inhalt hängt vom Kommandotyp ab. Ungenutzte Bytes werden von der Festplatte nicht überprüft. Das erste Byte des CDB beinhaltet den Operation Code des Kommandos, zum Beispiel weist die 12h auf ein Inquiry – Kommando hin. Der Aufbau verschiedener Kommandos wird in Kapitel 5.6 näher betrachtet. Byte 28 – 31 (Data Length) geben an, welche Datenmenge der Initiator erwartet.

5.3 FCP XFER RDY

Dieser Rahmen wird zum Beispiel von einer Festplatte gesendet, wenn diese bereit zum Empfangen von Daten ist. Jeder FCP_XFER_RDY Frame wird mit einer 05h im R_CTL Feld gekennzeichnet. Des weiteren enthält der Bereich F_CTL den Wert 89 00 00h. Dies bedeutet, dass der Rahmen vom Empfänger des Exchange gesendet wird und nicht vom Absender. Außerdem gibt er das Ende der Fibre Channel Sequenz an. Die Sequenzinitiative wird dem Initiator, also dem Absender übergeben, so dass die angeforderten Daten übermittelt werde können. Abbildung 5.3.1 und Abbildung 5.3.2 zeigen den Aufbau von FCP_XFER_RDY

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 5.3.1: FCP XFER RDY Header [5]

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 5.3.2: FCP XFER RDY Payload [5]

Wie im Bild 5.3.2 ersichtlich, wird im XFER RDY Datenblock ein relativer Offset und eine Burst Lenght übertragen. Anhand der Burst Lenght wird die Anzahl der Daten im nächsten Rahmen übermittelt. Damit werden auf der Empfängerseite immer genügend Kapazitäten zur Datenverarbeitung reserviert.

5.4 FCP DATA

Der FCP_DATA Rahmen hat die Aufgabe Anwenderdaten zu übermitteln. Daher stehen im Nutzdatenbereich die benutzerdefinierten Daten. Der Header kennzeichnet sich durch eine 01h im R_CTL Feld aus. Abbildung 5.4.1 zeigt die Zusammensetzung des FCP_DATA Headers.

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 5.4.1: FCP DATA format [5]

5.5 FCP RSP

Ein FCP Response Frame wird auf jeden FCP_CMND Vorgang zurückgesendet. Er zeichnet sich im R_CTL durch eine 07h aus und der F_CTL Bereich wird vom Empfänger mit 990000h gefüllt.

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 5.5.1: FCP_RSP Header [5]

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 5.5.2: FCP_RSP Payload [5]

Das Length of Response Information Feld enthält die Bytelänge der FCP Response Informationen. Gültige Längen für ein Response sind Null, Vier und Acht Byte. Zusätzliche SCSI Informationen können in Byte 32 bis 51 stehen. Die Menge dieser Informationen steht im Lenght of Sense Information Feld.
Der Residual Count wird nur verwendet, wenn die Felder Resid Overrun oder Underrun aktiviert sind. Dabei gibt das gesetzte Resid Overrun oder auch Residual Overrun Bit an, dass die Anzahl an Bytes im Residual Count Field die Menge von Daten ist, die nicht gesendet wurde. Das resultiert aus der Tatsache, das durch die angegebene Datenlänge im CDB des FCP CMND Rahmens weniger Daten erwartet

werden. Durch Verwenden des Resid Underrun oder Residual Underun wird gekennzeichnet, dass weniger Daten gesendet wurden als im DL Feld des CDB verlangt.
Sense Lenght Valid wird gesetzt, wenn zusätzliche SCSI-Informationen in der Payload enthalten sind und RSP Lenght Valid falls in der Payload noch weitere Response Informationen stehen.
Der SCSI Status wird an den Sender nach einem FCP Kommando zurück geschickt. Eine 00h (Status good) signalisiert zum Beispiel, das der Empfänger das Kommando erfolgreich bearbeitet hat. Weiter Statusanzeigen können zum Beispiel 02h (Check Condition) oder 08h (Busy) sein. Das Response Information Feld ist wie in Bild 5.5.3 aufgebaut.

Bit **Byte** (*)	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0 (24) : 2 (26)	(MSB)			Reserved				(LSB)
3 (27)	Response Information Code							
4 (28) : 7 (31)	(MSB)			Reserved (optional)				(LSB)

Abbildung 5.5.3: Response Information Format [5]

Liegt kein Fehler vor, wird dies mit dem Response Information Code 00h signalisiert. Sollten FCP CMND Felder ungültig sein muss der Code 02h eingestellt sein. Insgesamt gibt es sechs verschiedene Codes, auf die aber nicht näher eingegangen wird.

5.6 Wichtige SCSI Kommandos

Mit Hilfe von ChipScope, einem Entwicklungswerkzeug von Xilinx, war es möglich die empfangenen Informationen des QLogic HBAs wie auf einem Oszilloskop darzustellen und zu entschlüsseln. Dies war die Grundlage für alle gesendeten Antworten. Im weiteren Verlauf dieser Arbeit werden die wichtigsten Kommandos nach dem erfolgreichen Prozess-Login erklärt und an einem verwendeten Beispiel veranschaulicht.

5.6.1 Standard Inquiry

Das Inquiry Kommando wird im CDB Feld des FCP CMND Rahmens übermittelt. Mit ihm fordert der Initiator Parameterinformationen von der Festplatte an. Signalisiert wird dieses Kommando durch den Operation Code 12h im ersten Byte. Abbildung 5.6.1.1 zeigt den Aufbau des sechs Byte großen Kommandos

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 5.6.1.1: Inquiry Command (12h) [5]

Das EVPD Feld (Enable Vital Product Data) erlaubt dem Sender zusätzliche Informationen von der Festplatte zu erhalten. Wenn das EVPD Feld eine Null enthält, werden von der angesprochenen Festplatte die Standard Inquiry Informationen erwartet. Beinhaltet dieses Feld jedoch eine Eins, so werden die Vtial Product Daten von der Festplatte übermittelt, die durch das Page Code Feld spezifiziert sind. Auf das Page Code Feld soll an dieser Stelle nicht weiter eingegangen werden, da sich der Abschnitt nur mit dem Standard Inquiry beschäftigt. Antwortet der Empfänger mit einem FCP Data Frame, so stehen die Festplatten-Standardinformationen in den ersten 36 Bytes der Payload und geben Auskunft über beispielsweise Vendor Identification, Product Identification und Product Revision Level. Diese Angaben müssen im ASCII Format eingetragen werden Eine Reihe von weiteren, festplattenabhängigen Informationen stehen direkt im Anschluss an die ersten 36 Byte. Bild 5.6.1.2 zeigt das Format der Inquiry Daten für eine Festplatte.

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 5.6.1.2 Festplatten Inquiry Datenformat [5]

Auf ein Inquiry vom Initiator wird vom Empfänger immer mit einem FCP Data und einem FCP RSP Rahmen geantwortet. Die Abbildung 5.6.1.3 zeigt einen Standard Inquiry Ablauf mit den Einstellungen , wie er in dieser Diplomarbeit vorkam.

Standard Inquiry

Empfangen	Gesendet	Gesendet
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	

Abbildung 5.6.1.3: Standard Inquiry

5.6.2 Test Unit Ready

Das Test Unit Ready Kommando ist ein Werkzeug zum Festzustellen, ob eine logical Unit, zum Beispiel eine Festplatte, betriebsbereit ist und kein Mittel zur Selbstüberprüfung. Der sechs Byte große Befehl wird, wie jeder andere Befehl, im Command Description Block übertragen und ist, wie in Abbildung 5.6.2.1 ersichtlich, durch den Command Code 00h im ersten Byte gekennzeichnet.

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 5.6.2.1: Test Unit Ready Command (00h) [5]

Auf ein solches Kommando wird mit einem Response geantwortet. Ist die Festplatte betriebsbereit und kann einen Zugriff zulassen, so wird in der Antwort ein Status Good übertragen. Sollte irgend ein Fehler auftreten, wird dem Initiator ein Check Condition gesendet. Die Abbildung 5.6.2.2 zeigt einen Test Unit Ready Command, der von dem HBA gesendet wurde und die darauf entsprechende Antwort. Die Response Payload enthält die Information Good Status in Byte 39 von Beginn an gezählt. Die Länge der Response Informationen ist auf 08h festgelegt und die Response Information gibt an, dass kein Fehler vorliegt

Test Unit Ready

Empfangen	Gesendet
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	00000008
00000000	00000000
00000000	00000000

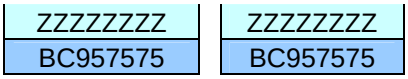

Abbildung 5.6.2.2: Test Unit Ready

5.6.3 Report LUN

Nachdem ein Initiator ein Inquiry Kommando an eine logical Unit verschickt hat, erhält er über sie festplattenspeziefische Informationen, die in Abbildung 5.6.1.2 bereits näher erläutert wurden. Der Report LUN Befehl fragt eine Auflistung aller Logical Unit Nummern von Geräten an, deren Peripheral Qualifier Feld den Wert 000b enthält. Das 12 Byte Kommando steht im CDB des CMND Rahmen und wird durch den Operation Code A0h kenntlich gemacht. Abbildung 5.6.3.1 zeigt das typische Format eines Report LUN Kommandos.

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3 – 5	Reserved							
6	(MSB)							
9	ALLOCATION LENGTH							(LSB)
10	Reserved							
11	CONTROL							

Abbildung 5.6.3.1: Report LUN (A0) [3]

Falls ein Report LUN Befehl nicht richtig ausgeführt werden kann, wird er mit Check Condition abgebrochen. Die angeforderten Parameter werden im FCP Data Block übertragen und richten sich nach dem Format in Abbildung 5.6.3.2.

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				LUN LIST LENGTH (n-7)				(LSB)
4								
7				Reserved				
				LUN list				
8								
15				LUN [first]				
				:				
n-7								
n				LUN [last]				

Abbildung 5.6.3.2: Report LUN Parameter Format [3]

Das LUN LIST Lenght Feld enthält die Länge der LUN Liste in Byte, die übertragen werden kann. In der Abbildung 5.6.3.3 erkennt man einen vollständigen Report LUN Vorgang. Auf das Kommando folgt immer ein FCP Data Frame mit dem Inhalt der LUN Liste und ein Response Frame.

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

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

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

Abbildung 5.6.3.3: Report LUN

Da nur eine Festplatte simuliert werden soll, muss die Länge der LUN Liste auf Acht gesetzt werden und in den Bytes 8-15 muss 00h eingetragen werden. Auffällig im Response Frame ist, dass das Resid Underrun Bit in Byte 10 aktiviert ist. Das resultiert aus der Tatsache, das 2048 Byte maximal vom Initiator erwartet werden, das Datenpaket aber nur 16 Byte an Informationen enthält. Daher steht im Residual Count Feld mit 07F0h oder 2032b die Differenz zwischen maximal erwarteten Daten und tatsächlich gesendeten Daten.

5.6.4 Read Capacity

Der Read Capacity Command bietet dem Initiator die Möglichkeit, die Kapazität einer Festplatte zu erfragen und nimmt im CDB des FCP CMND Frame 10 Byte ein .Sein spezieller Operation Code lautet 25h.

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 5.6.4.1: Read Capacity Command (25h) [5]

Die Funktion, die mit dem RelAdr (Relative Address) Feld verbunden ist, wird hier nicht verwendet und daher mit einer Null deaktiviert. Auch das PMI (Partial Medium Indicator) wird mit einer Null besetzt. Das bedeutet, dass die Information, die im Read Capacity Datenblock zurückgeschickt wird, aus der logischen Blockadresse und der Blocklänge des letzten logischen Blocks der letzten Logical Unit der Festplatte besteht. Das Read Capacity Daten Format sieht wie in Bild 5.6.4.2 dargestellt aus.

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 5.6.4.2: Read Capacity Datenblock [5]

Nachdem ein Initiator, in diesem Fall der HBA von QLogic, einen Read Capacity Command verschickt hat, wird vom Empfänger (FPGA) mit einem Daten-Frame geantwortet. Danach wird dieser Ablauf mit einem Response beendet. Abbildung 5.6.4.3 zeigt den gesamten Vorgang. Wie zu erkennen ist, enthält der Command Frame das Read Capacity Kommando (25h) und erwartet ein Datenpaket mit maximal Acht Byte. Der darauf folgende FCP Data Frame enthält Informationen über die Blockadresse 200000 h und die Blocklänge von 512 Byte (200h) Daraus resultiert eine angegebene Kapazität von 200000h X 200h = 40.000.000h oder dezimal

ausgedrückt, von 1073741824 Byte (1GB). Abgeschlossen wird dieser Vorgang, wie schon erwähnt, mit einem normalen Response.

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

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

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

Abbildung 5.6.4.3: Read Capacity

6 Ergebnisse

Die Idee, mit Hilfe von C oder C++ Programmierung und der FC HBA API auf das HBA von Qlogic zuzugreifen und Datenpakete zu verschicken, konnte nur teilweise realisiert werden, da der Umfang dieses Themas, mit zunehmenden Fortschritt, immer deutlicher wurde. Mit dem in C geschriebenen Testprogramm können die verschieden Adapter-Parameter abgefragt und am Bildschirm ausgegeben werden. Weitere Funktionalitäten konnten aber nicht mit eingebunden werden, da eine entsprechend antwortende Gegenstelle, zum Beispiel eine Fibre Channel Festplatte fehlte. Aufgrund dessen, wurde das Ziel verfolgt, über die Glasfaserverbindung das FPGA als SCSI Festplatte im Windows Betriebssystem anzumelden. So könnte vielleicht der Datentransfer zwischen den beiden Punkten über einfache File I/O Befehle

vorgenommen werden. Resultat dieser Möglichkeit ist, dass sich das FPGA als SCSI Festplatte nicht nur im Managertool anmeldet (Bild 6.1), sondern auch vom Gerätemanager des PC als Laufwerk erkannt wird. Dieser Punkt ist noch nicht einwandfrei gelöst, da das FPGA im Windows Gerätemanager unter sieben statt nur einem SCSI Device erkannt wird (siehe Abbildung 6.2). Die Ursache dafür wurde im Rahmen dieser Diplomarbeit nicht mehr herausgefunden. Ein Ansatz zu dieser Problematik könnte aber in der weiter unten beschriebenen Auffälligkeit zu finden sein. Mit dem SANsurfer FC HBA Manager konnte nach der erfolgreichen Anmeldung des SCSI Device ein Read/Write Buffer Test gestartet werden. Abbildung 6.3 zeigt den vom HBA gesendeten Write/Buffer Command (3Bh). Merkwürdig dabei sind die 24 Byte nach dem gesendeten Testmuster AAh. Der Zweck dieser Daten konnte wie bereits erwähnt, nicht herausgefunden werden.

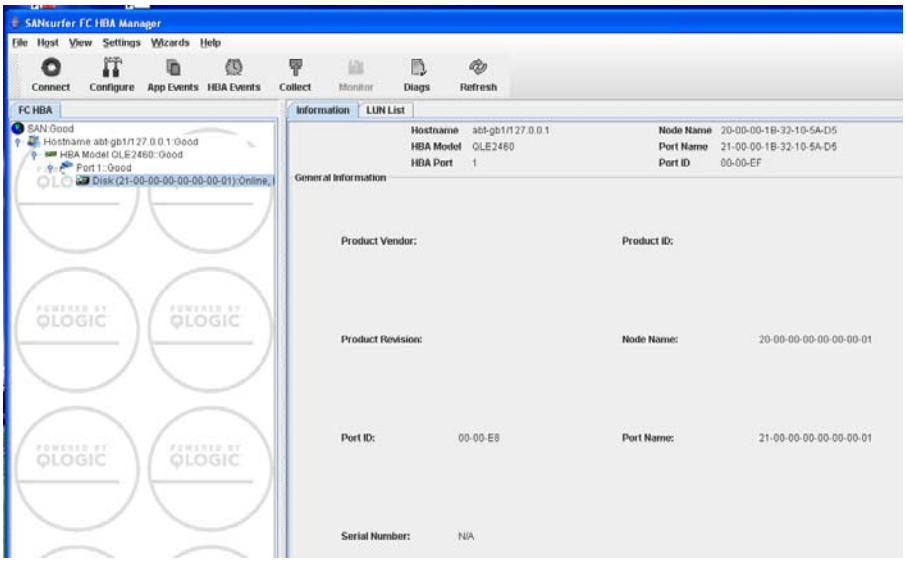

Abbildung 6.1 Angemeldete Disk im FC HBA Manager

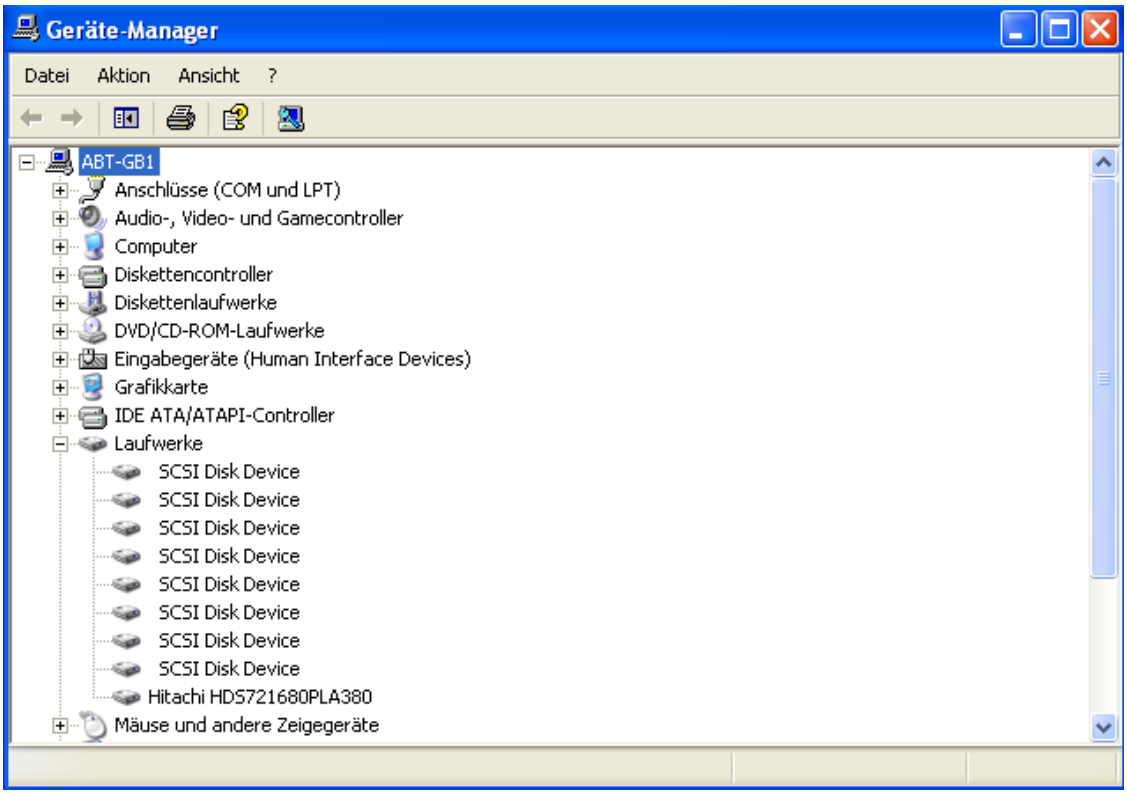

Abbildung 6.2 Windows Gerätemanager

Empfangen	Gesendet	Empfangen
BCB55656	BCB55656	BCB55656
060000E8	050000EF	010000E8
000000EF	000000E8	000000EF
08290000	08890000	08090008
YY000000	FF000000	YY+1000000
XXXXFFFF	XXXXFFFF	XXXXFFFF
00000000	00000000	00000000
00000000	00000000	AAAAAAAA
00000000	00000020	AAAAAAAA
00000001	00000000	25D6917C
3B020000	XXXXXXXX	CFEA917C
00000000	BC957575	30FD8B03
20000000		01000000
00000000		17000100
00000020		00000000
XXXXXXXX		XXXXXXXX
BC957575		BC957575

Abbildung 6.3 Write Buffer Command

7 Zusammenfassung

Die hier vorliegende Diplomarbeit entstand am Institut für zerstörungsfreie Prüfverfahren in Saarbrücken in der Abteilung Gerätebau und Systementwicklung. Diese Abteilung befasst sich mit der Entwicklung und dem Bau von Prüfgeräten und -anlagen zur zerstörungsfreien Prüfung verschiedenster Materialien. Diese Überprüfungen bringen einen immer größer werdenden Datenaufwand mit sich. Daher werden neue Lösungen und Technologien, die schnelle und sichere Datenübertragungen verbessern, immer aufmerksam verfolgt und bei Bedarf eingesetzt. Der zur Zeit verwendete Ethernet Standart zur Übertragung von Daten, gelangt zunehmend an seine Grenzen. Die Fibre Channel Technologie könnte an dieser Stelle durch ihre Übertragungsgeschwindigkeit und Übertragungssicherheit der Daten in Zukunft zum Einsatz kommen. Daraus entstand die Idee in dieser Diplomarbeit, erste Erfahrungen mit dem Fibre Channel Standard zu machen und Ergebnisse zu erreichen.

Grundlegendes Vorhaben war dabei zunächst eine Datenübertragungsstrecke zwischen einem FPGA –Entwicklungsboard und einer Fibre Channel Kaufkarte herzustellen. Im Folgenden sollte dann diese Verbindung, die Geschwindigkeit betreffend, optimiert werden. Zu Beginn dieser Diplomarbeit war es daher nötig, sich in den Fibre Channel Standard einzulesen und die mitgelieferte Programmierschnittstelle in einer C- Anwendung zu implementieren. Nachdem sich mehr und mehr herausstellte, dass ein Datentransfer mit der HBA API – Funktionalität nicht hergestellt werden konnte, wurde das Ziel verfolgt, die FPGA Karte als SCSI Festplatte am PC anzumelden. Daraus resultierte zwangsläufig das Studieren und Durchsuchen der entsprechenden Standards, mit dem Ergebnis eines erkannten SCSI Disk Devices im Gerätemanager des PCs. Da das Fibre Channel Thema eine interessante Grundlage für weitere Entwicklungen und Projekte am Fraunhofer Institut ist, wird man sich auch in Zukunft damit beschäftigen. Die Ergebnisse und Erkenntnisse dieser Diplomarbeit sind dafür eine gute Grundlage.

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10 Quellenverzeichnis

[1] Fibre Channel SNIA HBA APIProgrammer's Guide
http://docs.hp.com/en/J2635-90015/J2635-90015.pdf

[2] Technical Commitee T11: Fibre Channel HBA API (FC-HBA)
http://www.t11.org/ftp/t11/pub/fc/hba/04-137v0.pdf

[3] Technical Commitee T10: SCSI Primary Commands (SPC 4)
http://www.t10.org/ftp/t10/drafts/spc4/spc4r16.pdf

[4] Technical Commitee T10: Fibre Channel Protocol for SCSI, Version 4 (FCP-3)
http://www.t10.org/ftp/t10/drafts/fcp4/fcp4r01.pdf

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

[6] HITACHI, Hard Disk Specification Version 1.4.

[7] Technical Commitee T11: FC Framing and Signaling-2 (FC-FS-2)
http://www.t11.org/ftp/t11/pub/fc/fs-2/06-085v3.pdf

[8] TEC Channel
www.tecchannel.de/storage/grundlagen/465690/#

[9] QLogic, Fibre Channel HBA Installation Quick Start Guide

[10] QLogic, SANsurfer FC HBA Manager User's Guide

[11] QLogic, Fibre Channel HBA API Shared Library for Windows

11 Anhang

A1: CD- ROM mit folgendem Inhalt:

- HBA QLE2460 Datenblatt (PDF)
- Vollständiges Testprogramm in C-Programmierung
- Fibre Channel SNIA HBA API Programmer’s Guide (PDF)
- SANsurfer FC HBA Manager User’s Guide (PDF)
- Fibre Channel HBA Installation Quick Start Guide (PDF)
- FC-HBA Standard (PDF)
- SPC-4 Standard (PDF)
- FCP-3 Standard (PDF)
- Fibre Channel Interface Product Manual von Seagate (PDF)
- HARD Disk Specification Version 1.4 von HITACHI (PDF)