Examensarbete

High-Speed Storage Encryption over Fibre Channel

Examensarbete utfört i Elektroteknik
vid Tekniska högskolan vid Linköpings universitet
av

Christian Svensson

LiTH-ISY-EX--13/4713--SE

Linköping 2013

Master thesis

High-Speed Storage Encryption over Fibre Channel

by

Christian Svensson

LiTH-ISY-EX--13/4713--SE

2013-09-04

Supervisors: Johan Uppman, Andreas Ehliar

Examiner: Olle Seger

Avdelning, Institution Division, Department Division of Computer Engineering Department of Electrical Engineering SE-581 83 Linköping	**Datum** Date 2013-09-04

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ________________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ________________	**ISBN** — **ISRN** LiTH-ISY-EX--13/4713--SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-97043

Titel
Title High-Speed Storage Encryption over Fibre Channel

Författare
Author Christian Svensson

Sammanfattning
Abstract

This thesis focused on testing whether persistent encryption of Fibre Channel is doable and what kind of security it provides. It has been shown that intercepting, analysing and modifying Fibre Channel traffic is possible without any noticeable performance loss as long as latency is kept within certain boundaries. If latency are outside those boundaries extreme performance loss are to be expected. This latency demand puts further restrictions on the cryptography to be used.

Two platforms were simulated, implemented and explained. One for intercepting and modifying Fibre Channel and one for analysing Fibre Channel traffic using Linux and Wireshark.

Nyckelord
Keywords fibre channel, cryptography, fpga, network, communication

Abstract

This thesis focused on testing whether persistent encryption of Fibre Channel is doable and what kind of security it provides. It has been shown that intercepting, analysing and modifying Fibre Channel traffic is possible without any noticeable performance loss as long as latency is kept within certain boundaries. If latency are outside those boundaries extreme performance loss are to be expected. This latency demand puts further restrictions on the cryptography to be used.

Two platforms were simulated, implemented and explained. One for intercepting and modifying Fibre Channel and one for analysing Fibre Channel traffic using Linux and Wireshark.

Acknowledgements

I would like to thank a few people who have helped me throughout the thesis.

Thanks to my supervisors Johan Uppman and Andreas Ehliar for very valuable ideas and feedback. It would have been incredibly hard to do this without you.

Thanks to my examiner Olle Seger for taking his time supervising the thesis.

A specially big thanks to SECTRA Communication for hosting this thesis and providing me with all the equipment I needed as well as all the wonderful co-workers I have had the pleasure of working with.

Finally, thanks to Markus Häll and Erik Carlsson for information of how Fibre Channel is used in the real world and thanks to Michael Richardson, Guy Harris and Martin Kaiser for merging my patches in libpcap and Wireshark.

Contents

1 Introduction **4**
- 1.1 Thesis Formulation . . . 4
  - 1.1.1 Open/Secure Cut . . . 5
- 1.2 Approach . . . 5
  - 1.2.1 Primary Scenario: External Hardware Encryption after HBA . . . 5
  - 1.2.2 Secondary Scenario: Internal Hardware Encryption on HBA . . . 5
  - 1.2.3 Tertiary Scenario: Encrypted Transport . . . 6
  - 1.2.4 Latency . . . 6
  - 1.2.5 Performance and Backwards Compatibility . . . 6
  - 1.2.6 Fibre Channel Topology . . . 7
  - 1.2.7 Cut-through Forwarding versus Store and Forward . . . 7

2 Fibre Channel, SCSI and Cryptography **8**
- 2.1 Fibre Channel . . . 8
  - 2.1.1 FC-0: Physical Layer . . . 8
  - 2.1.2 FC-1: Code Layer . . . 8
  - 2.1.3 FC-2: Protocol Layer . . . 9
  - 2.1.4 FC-3: Services Layer . . . 9
  - 2.1.5 FC-4: Mapping Layer . . . 10
- 2.2 SCSI . . . 10
- 2.3 Cryptography . . . 11
  - 2.3.1 Open, Secure, Red and Black . . . 11

3 Cut-through Forwarding **13**
- 3.1 Hardware . . . 13
- 3.2 Simulation Model . . . 13
- 3.3 Implementation . . . 14
  - 3.3.1 Synthesis Statistics . . . 15
- 3.4 Link Snooper Connection Ports . . . 15

4 Link Snooper **17**
- 4.1 Hardware . . . 17
- 4.2 Implementation . . . 18
  - 4.2.1 On Board Logic . . . 18
  - 4.2.2 Linux Kernel . . . 18
  - 4.2.3 Linux User Space . . . 19
- 4.3 Usage . . . 19

5 Fibre Channel Encryptor **21**

5.1 Simulation Model . . . 21
5.2 Implementation . . . 21
5.2.1 Data Flow . . . 21
5.2.2 Encryption . . . 22
5.2.3 Synthesis Statistics . . . 22

6 Results **24**
6.1 Measurement Setup . . . 24
6.1.1 Computer A . . . 24
6.1.2 Computer B . . . 24
6.2 Baseline Performance . . . 24
6.3 Fibre Channel Performance . . . 25
6.4 Latency and Performance Impact . . . 27
6.4.1 Basic Latency . . . 27
6.4.2 Frame Sizes and Latency . . . 28
6.4.3 Asymmetrical versus Symmetrical Latency . . . 29
6.4.4 Summary . . . 29

7 Discussion **31**
7.1 Security Implications . . . 31
7.1.1 Open/Secure Cut . . . 31
7.1.2 Encryption Exchange Record . . . 31
7.1.3 Unknown Data . . . 31
7.2 Encrypted Transport . . . 31
7.3 Conclusion . . . 32

8 Future Work **33**

A Source Code **34**
A.1 Measurement . . . 34
A.1.1 blk_scan.sh . . . 34

Bibliography **35**

Nomenclature

CDC	Clock domain crossing.
CRC	Cyclic redundancy check. An error-detection code.
EOF	End of frame.
F-port	Fabric port.
FCoE	Fibre Channel over Ethernet.
FDE	Full disk encryption.
FPGA	Field programmable gate array.
FSM	Finite state machine.
HBA	Host bus adapter. Used to connect a Fibre Channel device to the host.
IP	Internet protocol.
LBA	Logical block address.
LUN	Logical unit number.
N-port	Node port.
PCAP	Packet capture file format. Used to store network packets.
SCSI	Small computer system interface. A storage standard.
SerDes	Serializer/Deserializer.
SMA	Subminiature version A. An RF connector.
SOF	Start of frame.
VHDL	VHSIC hardware description language.
VHSIC	Very high speed integrated circuits.
xxGFC	Gigabit Fibre Channel. Speed grade of Fibre Channel.

List of Figures

1.1.1 External hardware encryption after HBA 4
1.1.2 Internal hardware encryption on HBA 5
1.1.3 Internal software encryption before HBA 5
1.1.4 Encrypted transport 5
1.1.5 Fibre Channel layers [T11 Technical Committee(2011), FC-PI-5, page 16] 6

2.1.1 Fibre Channel frame format [T11 Technical Committee(2010), page 130] 9
2.1.2 Fibre Channel frame headers [T11 Technical Committee(2010), page 142] 9
2.1.3 Wireshark showing FCP and SCSI CDB 10
2.2.1 A Fibre Channel read exchange 12

3.1.1 The forwarding platform 14
3.2.1 Simulation - Test bench 14
3.2.2 Simulation - Terminator block 14
3.2.3 Simulation - Place holder encryptor 15
3.3.1 Implementation of the platform 15

4.1.1 Xilinx ML505 development kit 17
4.2.1 Block diagram of the link snooper 18
4.2.2 Block diagram of the PCIe access block 19
4.3.1 Link Snooper connected to the forwarding board 19

5.2.1 Partial block diagram of the encryptor platform 21
5.2.2 Block diagram of the encryptor module 22

6.2.1 Reading data from RAM 25
6.3.1 Reading data from RAM over Fibre Channel 26
6.3.2 Reading data from RAM over Fibre Channel reversed 26
6.3.3 Reading data from RAM (Summary) 27
6.4.1 Latency block diagram 27
6.4.2 Reading data from RAM over Fibre Channel with latency (Summary) 28
6.4.3 Reading data over Fibre Channel with varying latency and frame sizes 29
6.4.4 Comparison of symmetric and asymmetric latency 29

7.2.1 Encrypted transport implementation 32

List of Tables

3.3.1 Resource usage for the forwarding platform 16

5.2.1 Resource usage for the forwarding platform with encryption 23

Chapter 1

Introduction

This report begins with a thesis formulation of what problems will be analysed and what is included in the scope of the thesis. Following that is a brief introductions to Fibre Channel, SCSI and cryptography to make the reader familiar with the technologies discussed.

The chapters 3 to 5 describes the implementation of the hardware used in the thesis while chapter 6 discusses the measured results. Finally chapter 7 and 8 discusses limitations and possible future work.

1.1 Thesis Formulation

The problem in focus is allowing storage of sensitive information using Fibre Channel. The fundamental scenario is that a host wants to transmit sensitive information (secure) to be stored on storage connected via Fibre Channel where the link and storage is considered to be untrusted (open).

This may be achieved in a number of ways:

- External hardware encryption after HBA (figure 1.1.1)
- Internal hardware encryption on HBA (figure 1.1.2)
- Internal software encryption before HBA (figure 1.1.3)
- Encrypted transport (figure 1.1.4)

Internal software encryption using a standard FDE software works by encrypting the data before it leaves the operating system. This is considered out of scope and will not be researched in the thesis.

Encrypted transport assumes that only the link is untrusted, not the storage. See 1.2.3.

A non-trivial problem is where to place the cut between the trusted and untrusted environments. This problem is discussed in section 1.1.1.

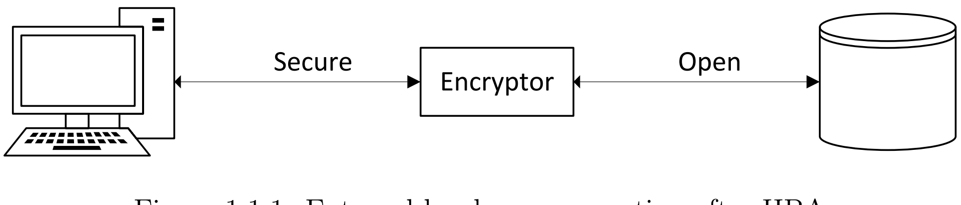


Figure 1.1.1: External hardware encryption after HBA

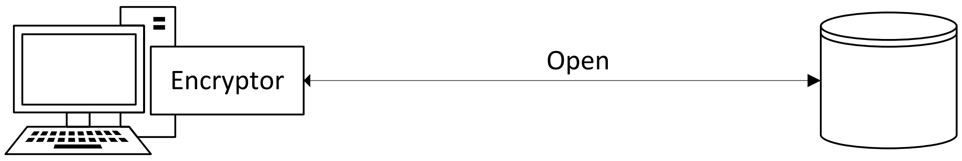


Figure 1.1.2: Internal hardware encryption on HBA

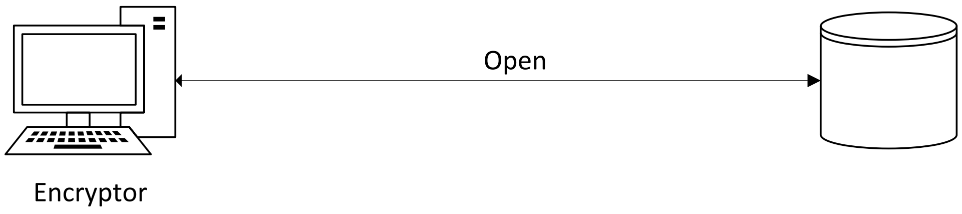


Figure 1.1.3: Internal software encryption before HBA

1.1.1 Open/Secure Cut

Fibre Channel is a layer based technology (figure 1.1.5) which makes it possible to place the open/secure cut at a number of different places, all with different implications. E.g. placing the cut between FC-0 and FC-1 would result in a transport based encryption. A suitable place for the cut will be as high up as possible, e.g. inside the SCSI read/write commands. Placing the cut inside the SCSI layer have the benefit of a potentially simple implementation but may reveal too much information.

Another possibility is to place the cut between FC-3 and FC-4. In this case the encryptor would need to terminate the SCSI commands and generate new (secured) SCSI commands to the back end.

1.2 Approach

The approach has been divided into three different scenarios.

1.2.1 Primary Scenario: External Hardware Encryption after HBA

Using an commercially available HBA the idea is to emulate the storage interface (F-port or N-port) to the HBA. The encryptor should encrypt and decrypt sensitive data passing through it to the open or secure side respectively (illustrated in figure 1.1.1).

Since the other scenarios are mutations of this scenario, this will be the scenario to be implemented.

1.2.2 Secondary Scenario: Internal Hardware Encryption on HBA

Instead of having a standalone HBA send data to the encryptor, this scenario works by having the encryptor act as an HBA to the host computer as well as handle the Fibre Channel

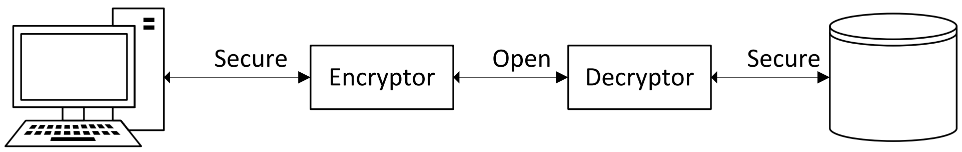


Figure 1.1.4: Encrypted transport

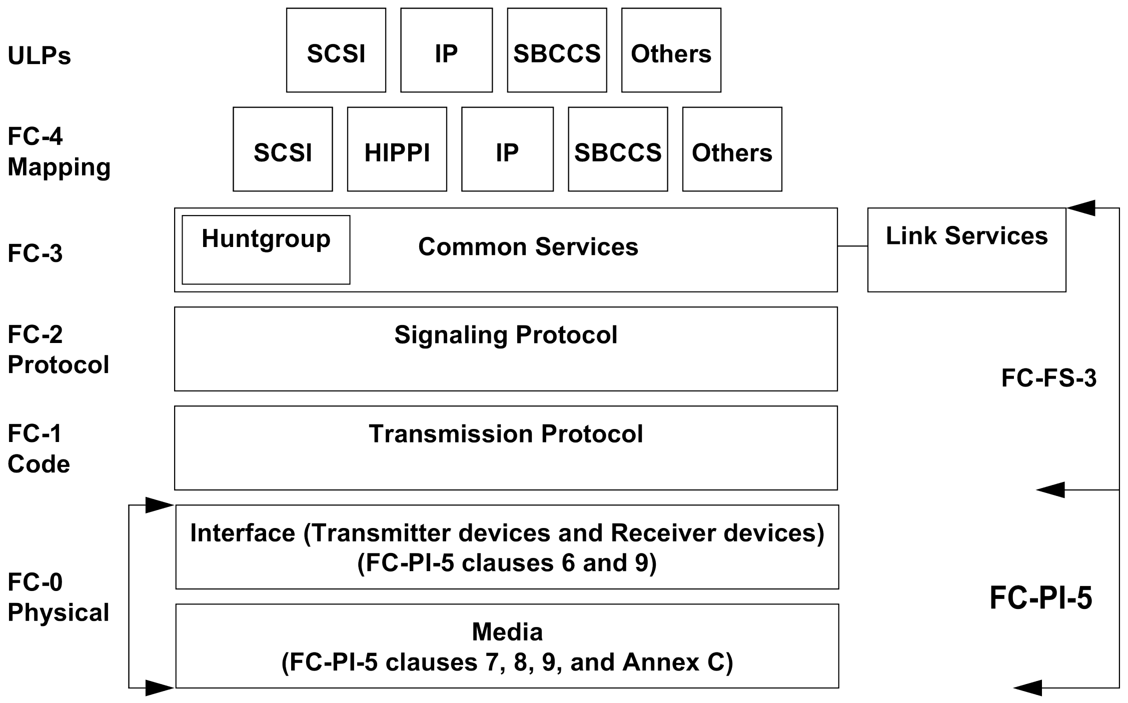


Figure 1.1.5: Fibre Channel layers [T11 Technical Committee(2011), FC-PI-5, page 16]

communication in the same manner as in section 1.2.1 (illustrated in figure 1.1.2).

An intuitive implementation of this is implementing an HBA with corresponding driver. It is also possible to emulate a network interface and send frames encapsulated in FCoE packets to the kernel. This scenario is deemed to be too complex for this thesis and will not be implemented.

1.2.3 Tertiary Scenario: Encrypted Transport

There is also the possible application of simply transporting the Fibre Channel communication encrypted and decrypt it just before it reaches the storage (illustrated in figure 1.1.4). This puts other demands on the storage environment and will not be the focus of the thesis, but it will be explored as an option.

1.2.4 Latency

Latency is the measurement of time delay that the data experiences while flowing through the system. For this thesis it is the time delay added by the platform that is inserted between the endpoints.

Measurements of latency and research on what effect it has on throughput are presented in chapter 6.

1.2.5 Performance and Backwards Compatibility

The back end storage is assumed to be any modern storage capable of running at the speeds 1-, 2- or 4GFC. Given the hardware available the speed will be fixed at 2GFC and no speed negotiation will be supported.

1.2.6 Fibre Channel Topology

Focus will be on point-to-point. Other topologies will not be examined as to limit the scope of the thesis.

1.2.7 Cut-through Forwarding versus Store and Forward

The two methods for network forwarding are cut-through forwarding and store-and-forward forwarding. The main differences between these two methods are in latency [Juniper Networks(2012)] and space complexity.

The cut-through forwarding method works by streaming the data through the forwarding engine as quickly as possible. Transmission is done when the first processed data is available which means that the latency increase is almost constant [Juniper Networks(2012)] and the memory requirements are limited to the amount of data currently inside the forwarding engine. The constant latency property is derived from the fact that frame transmission must normally be continuous i.e. when the frame has begun transmission it is required to maintain the data stream without pauses.

The store and forward method works by storing the whole frame in some kind of memory before it is forwarded [Yang Yang(2012)]. This inherently means that the forwarding engine needs to wait for the whole frame to be received before even starting to transmit thus increasing latency as well as making it dependent on the frame's size. Since the packets need to be stored it requires memories to both read outgoing frames and write new incoming ones. Given that the frame is stored before transmission it is potentially easy to extensively modify the frame such as changing its size.

The method of choice in this thesis is cut-through forwarding due to it being easier to implement.

Chapter 2

Fibre Channel, SCSI and Cryptography

This chapter exists to make the reader familiar with the concepts of Fibre Channel and SCSI that is referenced in this thesis. The different areas are explained in a bottom-up approach meaning that it starts with the electrical layer and moves up towards more logical layers.

2.1 Fibre Channel

Fibre Channel is split into a number of abstraction layers (figure 1.1.5) called FC-0 through FC-4, FC-0 being the physical (e.g. electrical or optical) layer and FC-4 being the service layer. The reader might be familiar with the Open Systems Interconnection (OSI) model at which case it should be noted that the layers in Fibre Channel does not match the OSI layers one to one but the same concept applies.

The layers described herein are defined in the [T11 Technical Committee(2010), FC-FS-3] and [T11 Technical Committee(2011), FC-PI-5] standards.

2.1.1 FC-0: Physical Layer

This thesis uses SFP modules and built in transceivers which makes the physical layer largely hidden. To understand FC-0 it is important to know that the upper layer (FC-1) always makes sure the data is encoded with 8b/10b. This encoding allows an 8-bit byte and a control bit to be transformed into 10-bits depending on the DC bias at the time as well as guaranteeing that it is possible to align the data clock to the data stream (clock recovery) [Ulf Troppens(2009), 3.3.3]. 8b/10b uses the DC bias to calculate the next character using what is called a running disparity. Balancing the DC bias around zero will produce a DC balanced stream - a stream where the mean current flow is equal to zero.

Consider a data stream encoded for +5 V for logical high and 0V for logical low. Sending data with at least one logical high bit on this bus will cause a current to flow from the sender to the receiver creating a biased system.

Now consider the case where a logical high is +2.5V and the logical low is -2.5V. Sending data on this bus may send current flowing in either direction depending on the data transmitted. However it is trivial to see that the slightest logical imbalance in the bits would cause the bus to have a DC bias. This problem is solved with the running disparity in 8b/10b.

2.1.2 FC-1: Code Layer

FC-1 introduces something called transmission character, transmission word and ordered sets.

A transmission character is an 8b/10b encoded byte plus control bit. It's written as Xyy.z where X is “K” if control is high and “D” if control is low, yy is the 5 least significant bits and z is the 3 most significant. Example: K28.5 is 10111100 with control bit high.

A transmission word is a collection of four transmission characters. An ordered set is a special transmission word where the first character is K28.5.

Ordered sets are used to send control information such as when there is no data to be transmitted, during link negotiations and marking the start and end of data frames. For example, the idle marker is K28.5, D21.4, D21.5, D21.5

2.1.3 FC-2: Protocol Layer

Special frame markers (SOF and EOF) delimits the Fibre Channel frame. The frame format is show in figure 2.1.1 and encompasses headers, data and CRC. The frame header (figure 2.1.2) and the extended header contains properties such as addresses and control flags used to manage what is called an exchange[Ulf Troppens(2009), 3.3.4]. An exchange is a conversation between two endpoints and is referenced by an unique number called OX_ID.

The field R_CTL is called routing control and is used to classify the frame function such as sending data or commands. Frame functions can be both solicited and unsolicited and are given different routing classes depending on which is the case. For example a read or write command always produces solicited data.

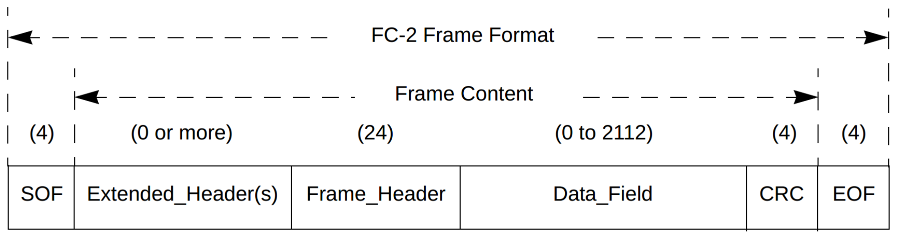
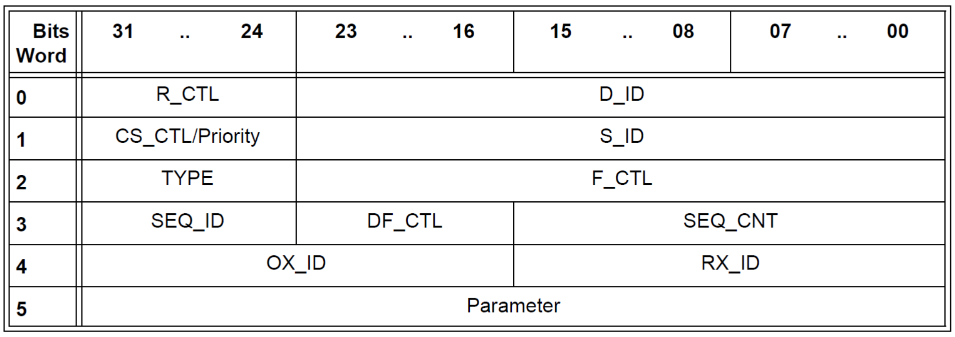


Figure 2.1.1: Fibre Channel frame format [T11 Technical Committee(2010), page 130]
The size is given in bytes.



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			

Figure 2.1.2: Fibre Channel frame headers [T11 Technical Committee(2010), page 142]

2.1.4 FC-3: Services Layer

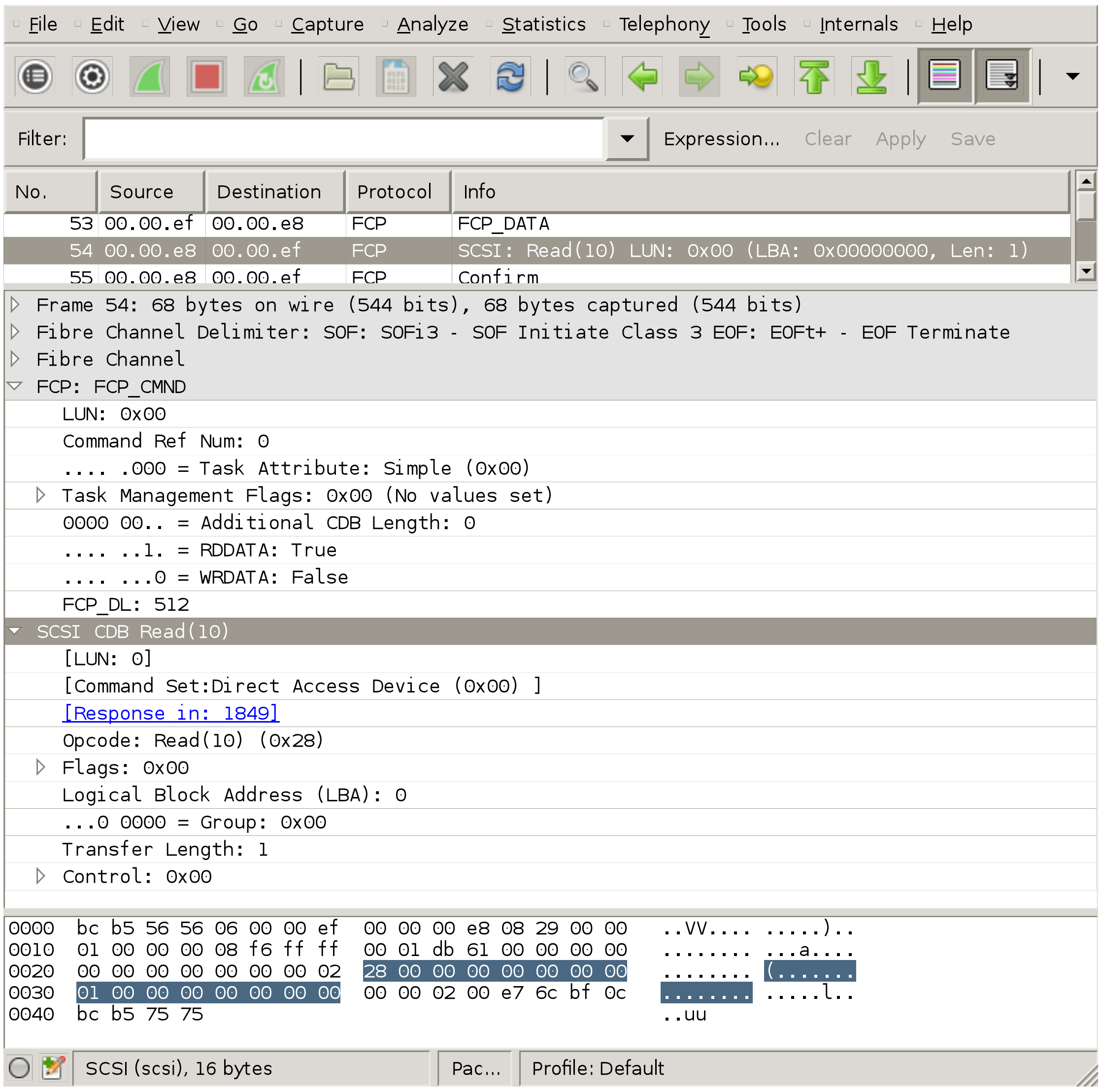
The FC-3 services layer is used to facilitate communication between the Fibre Channel device and the fabric it is attached to. Understanding of this layer is not needed for this thesis and will be omitted.

2.1.5 FC-4: Mapping Layer

The mapping layer is the final layer in the Fibre Channel layer model. While this layer is used to carry many protocols such as IP or SCSI, it is only the later that is relevant for this thesis.

Carrying SCSI over Fibre Channel is done by the Fibre Channel Protocol (FCP). FCP contains information needed to route the SCSI payload to the correct device as well as the SCSI Command Descriptor Block (CDB). SCSI CDB corresponds exactly to the SCSI standard and is the point at which Fibre Channel no longer cares about the contents and passes it through as it is.

An example of FCP and CDB captured from a SCSI exchange is shown in figure 2.1.3.



```
No.  Source     Destination  Protocol  Info
 53  00.00.ef   00.00.e8     FCP       FCP_DATA
 54  00.00.e8   00.00.ef     FCP       SCSI: Read(10) LUN: 0x00 (LBA: 0x00000000, Len: 1)
 55  00.00.e8   00.00.ef     FCP       Confirm
```

```
▷ Frame 54: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
▷ Fibre Channel Delimiter: SOF: SOFi3 - SOF Initiate Class 3 EOF: EOFt+ - EOF Terminate
▷ Fibre Channel
▽ FCP: FCP_CMND
    LUN: 0x00
    Command Ref Num: 0
    .... .000 = Task Attribute: Simple (0x00)
  ▷ Task Management Flags: 0x00 (No values set)
    0000 00.. = Additional CDB Length: 0
    .... ..1. = RDDATA: True
    .... ...0 = WRDATA: False
    FCP_DL: 512
▾ SCSI CDB Read(10)
    [LUN: 0]
    [Command Set:Direct Access Device (0x00) ]
    [Response in: 1849]
    Opcode: Read(10) (0x28)
  ▷ Flags: 0x00
    Logical Block Address (LBA): 0
    ...0 0000 = Group: 0x00
    Transfer Length: 1
  ▷ Control: 0x00
```

```
0000  bc b5 56 56 06 00 00 ef  00 00 00 e8 08 29 00 00   ..VV.... .....)..
0010  01 00 00 00 08 f6 ff ff  00 01 db 61 00 00 00 00   ........ ...a....
0020  00 00 00 00 00 00 00 02  28 00 00 00 00 00 00 00   ........ (.......
0030  01 00 00 00 00 00 00 00  00 00 02 00 e7 6c bf 0c   ........ .....l..
0040  bc b5 75 75                                        ..uu
```

Figure 2.1.3: Wireshark showing FCP and SCSI CDB

2.2 SCSI

SCSI is a set of standards for connecting and transferring data between devices. While this thesis uses the term SCSI it is only a very small subset of the SCSI standards that are referred to.

Commands for SCSI works by sending CDB frames to a device addressed by a Logical Unit Number (LUN) where the CDB contains an opcode indicating which function the device is requested to perform. Depending on the opcode the rest of the CDB will vary.

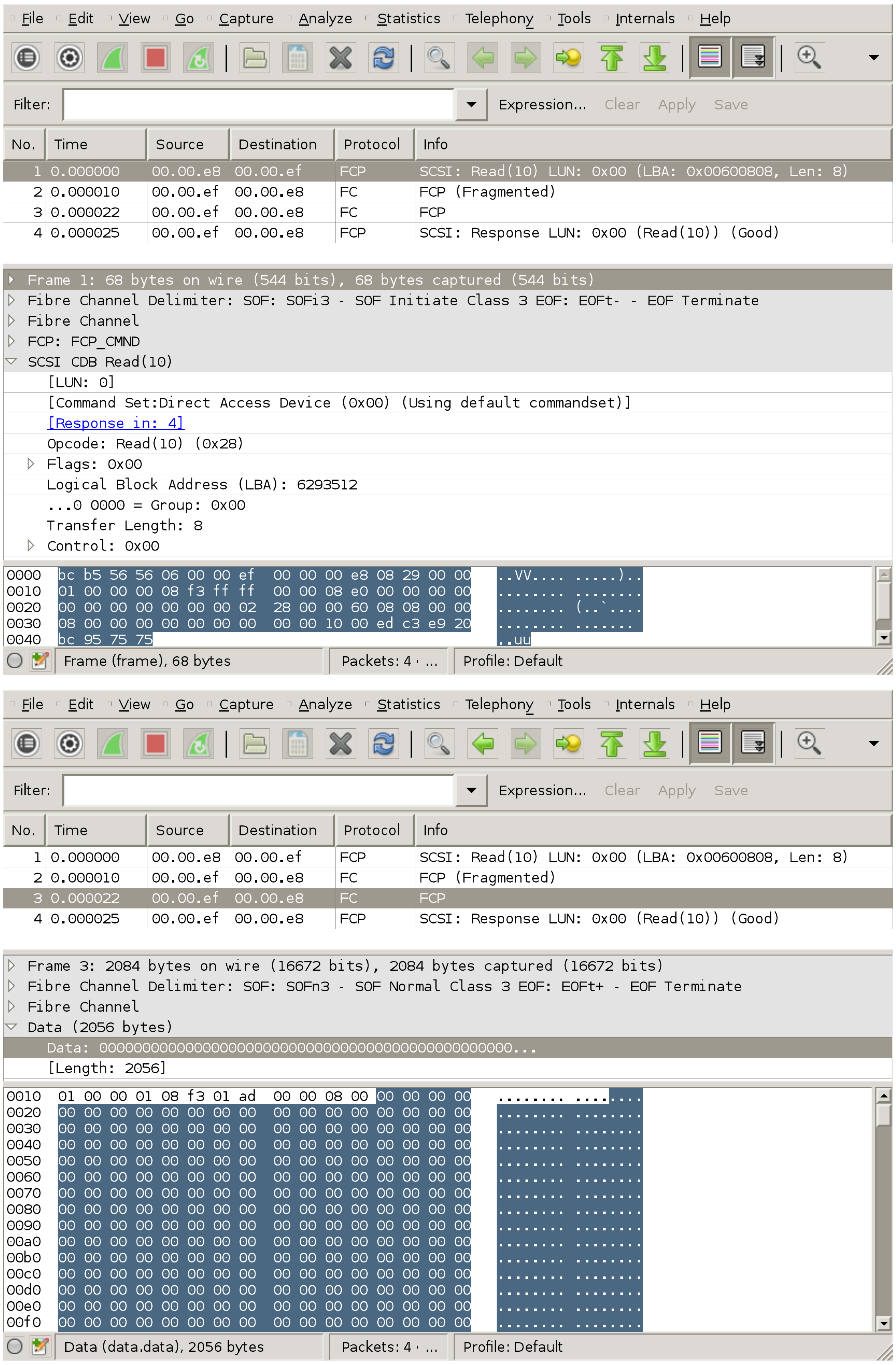
Two such commands are the Read(10) and Write(10) command. The number after the command is used to separate the different versions of read and write commands. A SCSI over Fibre Channel exchange can be seen in figure 2.2.1. Refer to [Seagate(2006)] for more information regarding SCSI commands and what data they contain.

2.3 Cryptography

The concept of encryption is to transform data so that only authorised parties can recover it. This thesis will not focus on the algorithm itself but rather what requirements are to be satisfied if the algorithm is to be used for encryption in these scenarios.

2.3.1 Open, Secure, Red and Black

In the first chapter the terms open and secure were used. Readers familiar with cryptography may recognise “red” and “black” instead, where red is the side that carries the classified data (called “secure” in the previous figures) and black carries the encrypted data (called “open” in previous figures). Red and black will be used from here on.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No. Time      Source    Destination Protocol Info
  1 0.000000  00.00.e8  00.00.ef    FCP      SCSI: Read(10) LUN: 0x00 (LBA: 0x00600808, Len: 8)
  2 0.000010  00.00.ef  00.00.e8    FC       FCP (Fragmented)
  3 0.000022  00.00.ef  00.00.e8    FC       FCP
  4 0.000025  00.00.ef  00.00.e8    FCP      SCSI: Response LUN: 0x00 (Read(10)) (Good)

Frame 1: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
Fibre Channel Delimiter: SOF: SOFi3 - SOF Initiate Class 3 EOF: EOFt- - EOF Terminate
Fibre Channel
FCP: FCP_CMND
SCSI CDB Read(10)
    [LUN: 0]
    [Command Set:Direct Access Device (0x00) (Using default commandset)]
    [Response in: 4]
    Opcode: Read(10) (0x28)
    Flags: 0x00
    Logical Block Address (LBA): 6293512
    ...0 0000 = Group: 0x00
    Transfer Length: 8
    Control: 0x00

0000 bc b5 56 56 06 00 00 ef  00 00 00 e8 08 29 00 00   ..VV.... .....)..
0010 01 00 00 00 08 f3 ff ff  00 00 08 e0 00 00 00 00   ........ ........
0020 00 00 00 00 00 00 00 02  28 00 00 60 08 08 00 00   ........ (..`....
0030 08 00 00 00 00 00 00 00  00 00 10 00 ed c3 e9 20   ........ .......
0040 bc 95 75 75                                        ..uu

Frame (frame), 68 bytes    Packets: 4 · ...    Profile: Default
```

```
No. Time      Source    Destination Protocol Info
  1 0.000000  00.00.e8  00.00.ef    FCP      SCSI: Read(10) LUN: 0x00 (LBA: 0x00600808, Len: 8)
  2 0.000010  00.00.ef  00.00.e8    FC       FCP (Fragmented)
  3 0.000022  00.00.ef  00.00.e8    FC       FCP
  4 0.000025  00.00.ef  00.00.e8    FCP      SCSI: Response LUN: 0x00 (Read(10)) (Good)

Frame 3: 2084 bytes on wire (16672 bits), 2084 bytes captured (16672 bits)
Fibre Channel Delimiter: SOF: SOFn3 - SOF Normal Class 3 EOF: EOFt+ - EOF Terminate
Fibre Channel
Data (2056 bytes)
    Data: 000000000000000000000000000000000000000000000000...
    [Length: 2056]

0010 01 00 00 01 08 f3 01 ad  00 00 08 00 00 00 00 00   ........ ........
0020 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0030 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

Data (data.data), 2056 bytes    Packets: 4 · ...    Profile: Default
```

Figure 2.2.1: A Fibre Channel read exchange

Chapter 3

Cut-through Forwarding

In order to access and modify the Fibre Channel frame contents it is necessary for the platform to be able to receive and transmit according to the FC-PI-5 and FC-FS-3 standards. These standards define how communication should be conducted between Fibre Channel equipments, what kind of coding is used, time-out values and other things the developer needs to know in order to model the correct behaviour. The goal with the cut-through forwarding scenario is to implement and verify the VHDL models based on the described behaviour.

3.1 Hardware

The hardware available for this scenario is:

- Two QLogic QLE2460 Fibre Channel HBA
- One Linux PC
- One Windows PC
- Hardware platform with one XC5VLX50 FPGA, two SFP cages and two SMA connector pairs.

The Linux PC hosts two SCSI devices using SCST[1] to be mounted by the Windows PC over Fibre Channel. The board is placed between these two PCs resulting in the connection shown in figure 3.1.1.

3.2 Simulation Model

Since cut-through forwarding is a matter of simply connecting the two transceivers together in a pass through configuration, a simulation model offers little to no benefit. The challenge was instead to find a good configuration for the transceivers themselves that would work with the existing Fibre Channel hardware.

Even with this in mind, a simulation model was produced in the early days of this thesis to test different ways of implementing and handling framing and encapsulation of Fibre Channel frames as well as to better understand how Fibre Channel works. Note that this simulation model does not match the blocks of the final implementation as much of the work is later done inside the hard transceiver blocks.

[1]Generic SCSI target subsystem for Linux. http://scst.sourceforge.net/

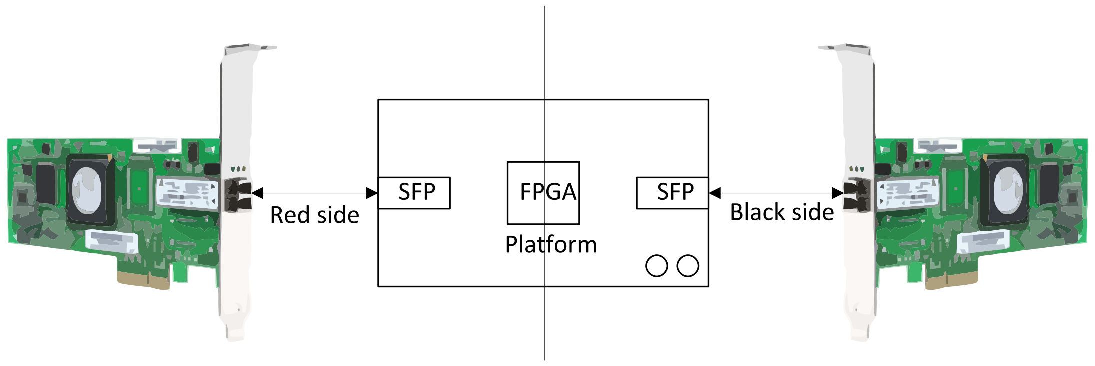


Figure 3.1.1: The forwarding platform
The black line illustrates the cut between red and black sides.

The test bench (figure 3.2.1) works by reading and writing PCAP files containing Fibre Channel frames. Using Wireshark one can easily compare the results and quickly determine if the output was correct. The data is loaded by a terminator block (figure 3.2.2), passed through the encryptor block and then logged as it reaches the other terminator block. The place holder encryptor module (figure 3.2.3) was used to compare cut-through forwarding to store and forward and did not perform any encryption or decryption of the data.

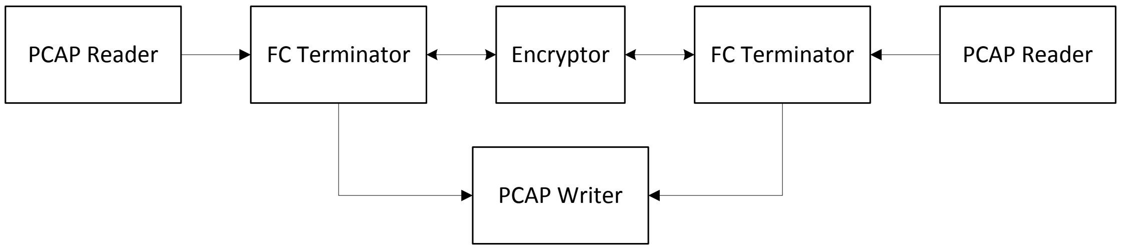


Figure 3.2.1: Simulation - Test bench

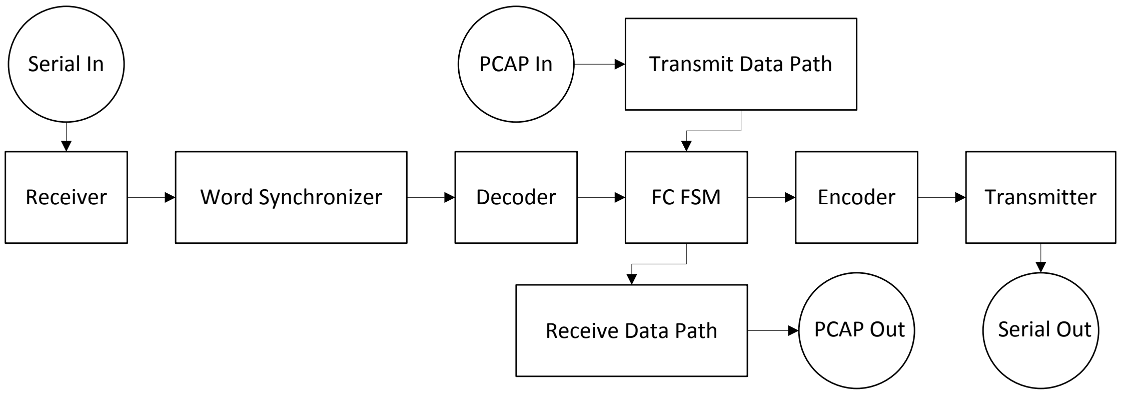


Figure 3.2.2: Simulation - Terminator block

3.3 Implementation

Implementation of the forwarding platform is not complicated although it must be precisely tuned to Fibre Channel to work satisfactory. The physical aspect of the platform is illustrated

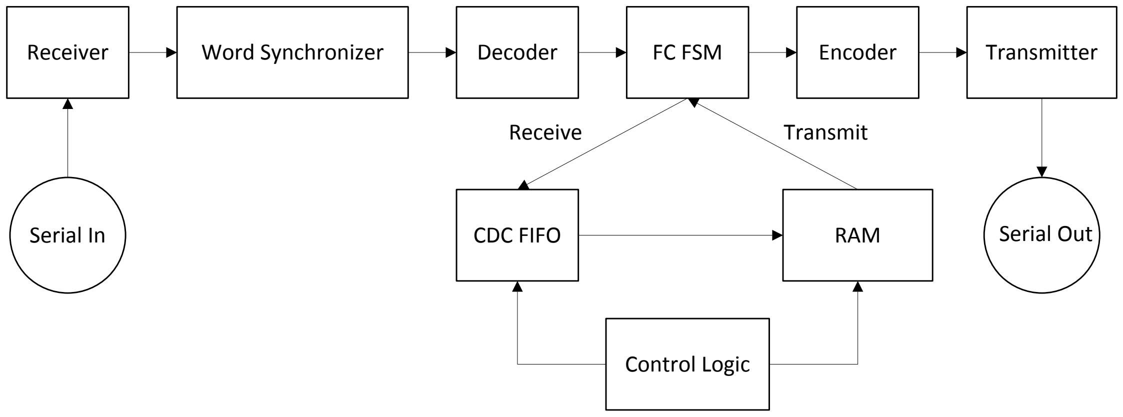


Figure 3.2.3: Simulation - Place holder encryptor

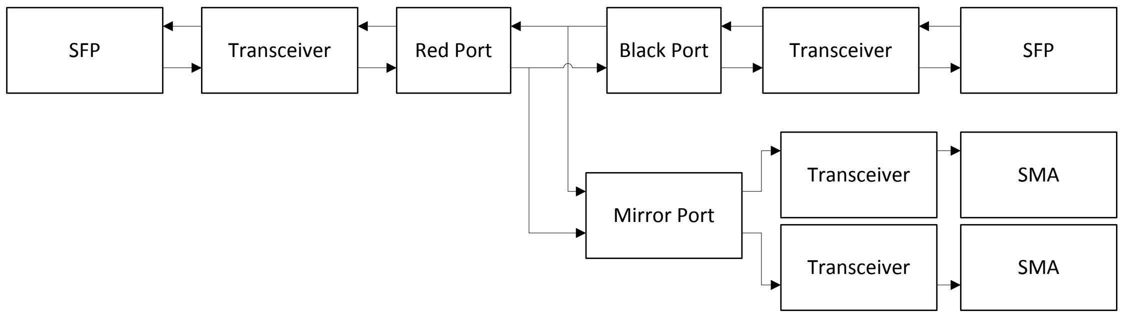


Figure 3.3.1: Implementation of the platform

in figure 3.1.1 while figure 3.3.1 shows the logical aspect. The port blocks contain framing and encapsulation logic as well as logic to detect loss of link. The interface from the port blocks are made to be intuitive and easy to use for the encryption module that will be added later.

Transceiver configuration is closely matched that described in [T11 Technical Committee(2010)] and Xilinx reference implementations. The throughput chosen in this implementation is 2GFC simply because that is the maximum possible throughput with the hardware available for this thesis.

3.3.1 Synthesis Statistics

Table 3.3.1 is provided for comparative purposes. See section 5.2.3 for an analysis.

3.4 Link Snooper Connection Ports

The platform is equipped with two extra transceivers which are used to mirror the outgoing or incoming link from the black SFP port (see lower part of figure 3.3.1). One transceiver is programmed to mirror both outgoing and incoming links depending on which is not idle while the other transceiver always mirrors the outgoing link. Incoming data on these transceivers are discarded.

Connecting one of these ports to the link snooper (See 2b “Link Snooper”) will make it possible to analyse the traffic being transmitted and/or received.

Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	40	28,800	1%
Number used as Flip Flops	40		
Number of Slice LUTs	89	28,800	1%
- Number used as logic	89	28,800	1%
– Number using O6 output only	85		
– Number using O5 output only	1		
– Number using O5 and O6	3		
Number of route-thrus	1		
- Number using O6 output only	1		
Number of occupied Slices	48	7,200	1%
Number of LUT Flip Flop pairs used	98		
- Number with an unused Flip Flop	58	98	59%
- Number with an unused LUT	9	98	9%
- Number of fully used LUT-FF pairs	31	98	31%
- Number of unique control sets	13		
- Number of slice register sites lost to control set restrictions	28	28,800	1%
Number of bonded IOBs	24	480	5%
- Number of LOCed IOBs	24	24	100%
- Number of bonded IPADs	12		
– Number of LOCed IPADs	8	12	66%
- Number of bonded OPADs	12		
– Number of LOCed OPADs	10	12	83%
Number of BUFG/BUFGCTRLs	13	32	40%
- Number used as BUFGs	13		
Number of DCM_ADVs	4	12	33%
Number of GTP_DUALs	3	6	50%
Average Fanout of Non-Clock Nets	3.06		

Table 3.3.1: Resource usage for the forwarding platform

Chapter 4

Link Snooper

The link snooper is a device that takes a Fibre Channel link and records certain frames for a host computer to extract and process. It is implemented as a network device in Linux with support included in Wireshark.

4.1 Hardware

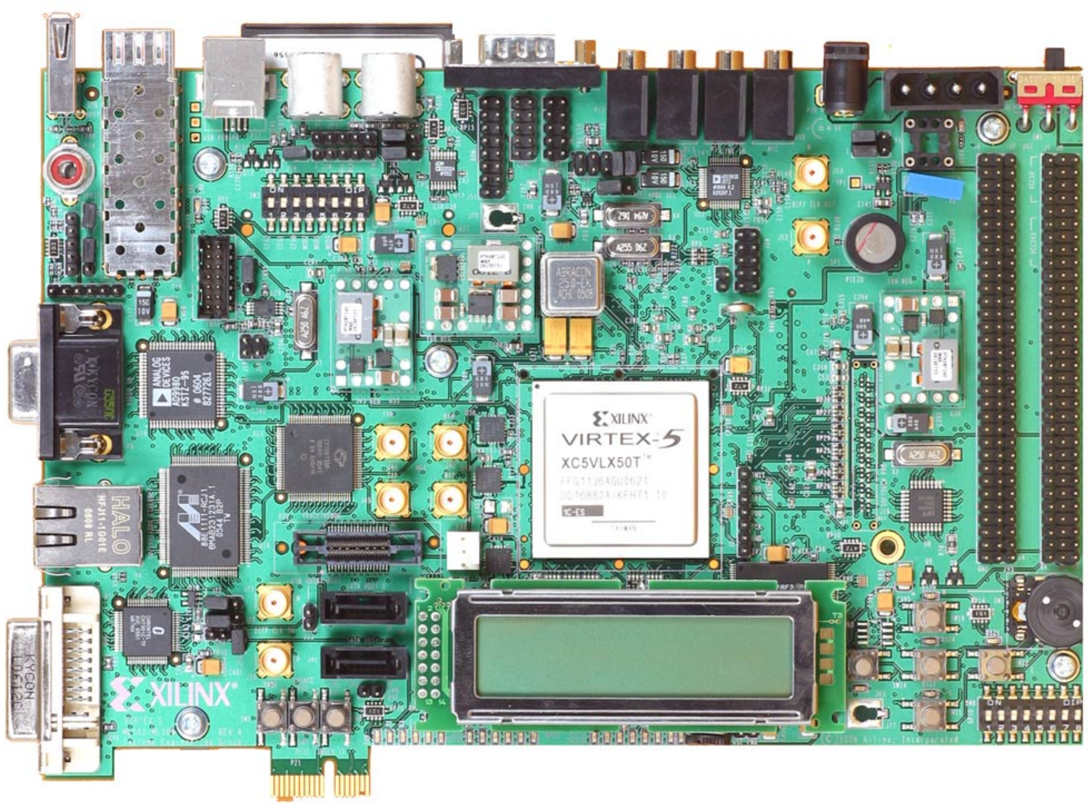


Figure 4.1.1: Xilinx ML505 development kit

The link snooper is implemented using the Xilinx ML505 development kit (figure 4.1.1) featuring an XC5VLX50 FPGA, one transceiver connected to an SFP port as well as an other one connected to SMA ports. Communication with a host computer is achieved via the PCIe interface.

4.2 Implementation

The implementation consists of three parts: on board logic, the linux kernel and linux user space.

4.2.1 On Board Logic

The hardware implementation consists of the same Fibre Channel port blocks as the forwarding platform, an alignment block, access logic and PCIe blocks (see figure 4.2.1).

The alignment block makes sure that the data fed to the access block is aligned on a two byte boundary to make it easier to filter unwanted transmission words. Alignment is done by looking for the IDLE transmission word and making sure that the first outputted two byte tuple is [K28.5, D21.4] followed by the next tuple [D21.5, D21.5]. If alignment is disabled data can under some circumstances be misaligned to [??, K28.5], [D21.4 D21.5], [D21.5, ??] which results in that the matching logic needs to be much more complex. This block is also used in the improved forwarding platform when encryption is added, see chapter 5.

The PCIe access block is where the PCIe and Fibre Channel domains meet (see figure 4.2.2). The concept is to store all Fibre Channel frames that are received to allow for later analysis using a big FIFO memory. The Fibre Channel domain writes each frame to the FIFO and the PCIe domain reads from it. The problem is that Fibre Channel always sends data regardless if it is only control information (IDLE for example) or real data which is why it is needed to filter out the unwanted data. The filter disables the FIFO write enable port when it detects that the current transmission word is a control word such as IDLE, R_RDY, ARBFF and so on resulting in that the FIFO only contains data of interest.

Controlling the PCIe access block is done by reading and writing to memory mapped addresses. For example, reading address 0x0 will produce the status register containing a handful of flags ranging from if the FIFO is empty to if it is currently writing. It is also possible to read data from the FIFO, query the filter for how many transmission words that have been dropped and see how much data has been saved in the FIFO.

The PCIe application and PCIe port blocks are generated from the Xilinx PCIe hard IP core with a few modifications to allow the PCIe access block to work as intended.

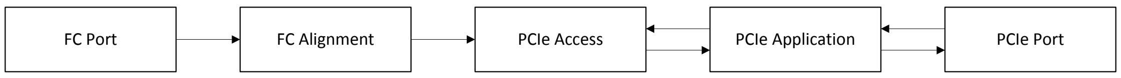


Figure 4.2.1: Block diagram of the link snooper

4.2.2 Linux Kernel

On the software side the first interaction with the hardware is in the Linux kernel. The link snooper has been implemented as a network device to provide out-of-the-box support for Wireshark and other protocol dissectors.

In order to fit the time frame for this thesis it was chosen to implement the link snooper as a polled device instead of an interrupt driven DMA one. This means that the Linux kernel needs to poll the device at specific intervals to check the device status and manually copy data from the device to its own memory. This consumes a lot of CPU time and makes high throughput data captures impossible. The rationale for creating this device was to verify the traffic outbound from the forwarding platform. This does not require capturing data at high rates nor does it require strict timing making the polling approach acceptable.

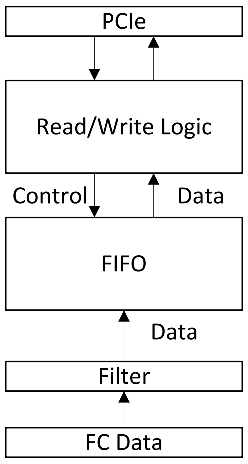


Figure 4.2.2: Block diagram of the PCIe access block

4.2.3 Linux User Space

Since the link snooper is presented as a network interface it is possible to use the industry standard program Wireshark to verify the Fibre Channel protocol contents. A Wireshark bug[1] was found and a libpcap patch[2] was merged to facilitate support for the link snooper. For the result, see figure 2.1.3 and figure 2.2.1.

4.3 Usage

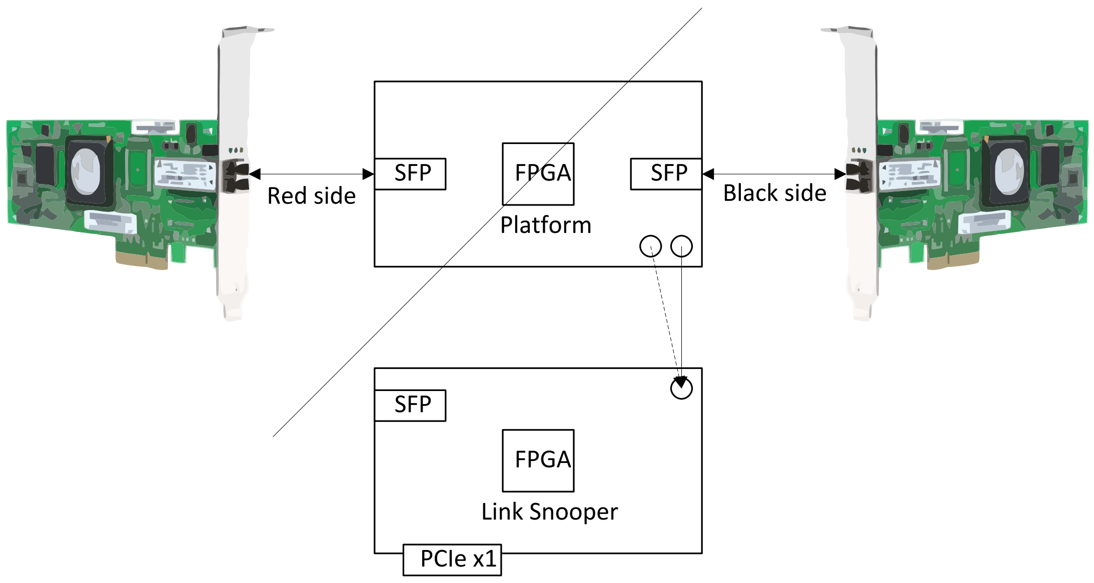


Figure 4.3.1: Link Snooper connected to the forwarding board
Note that the only black traffic is allowed to be inspected.

The link snooper connects to the data source either via SFP or differential SMA (figure 4.3.1). Using a PCIe x1 bus the computer can send commands and receive data and status informa-

[1]https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=8636
[2]https://github.com/mcr/libpcap/commit/cef10552042dcc8dfe55c6210aa492d9253ebfb5

tion from the device. A link capture is initialised by the computer by writing a capture command to the device. This will cause the device to capture data until its buffer is full at which point the computer will need to read the buffer until it is empty and start a new capture.

The device is implemented with a simple polling scheme where the computer asks the device if its buffer is full every 100 ms. Since the device is not implemented with direct memory access (DMA) the read operation is slow causing the buffer to overflow in the case of heavy link utilisation. Dropped frames are common but not necessarily disturbing as in most cases only a brief snapshot of the traffic flow is needed.

Chapter 5

Fibre Channel Encryptor

The Fibre Channel encryptor is a further development of the cut-through forwarding platform. The transceivers are still in a pass through configuration but with an encryptor module inside the forwarding engine, between the endpoints. The module is called encryptor despite that it also decrypts.

5.1 Simulation Model

The simulation model is the same as in section 3.2 but instead of using a place holder for the encryptor module, the real module is used.

5.2 Implementation

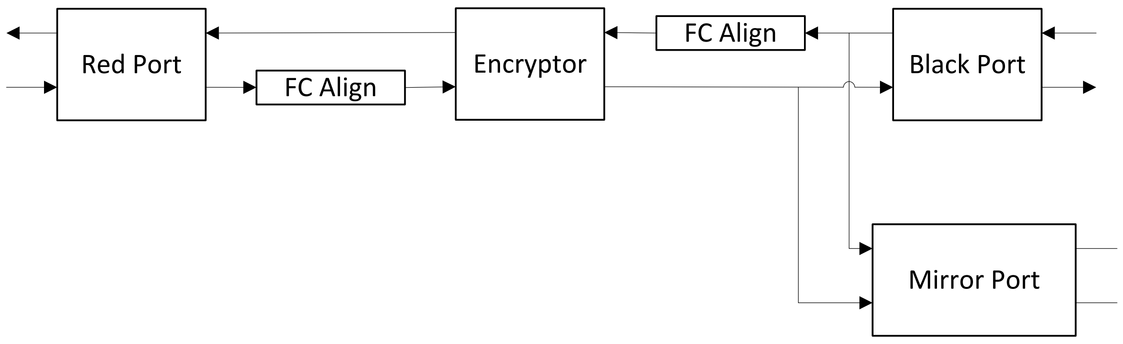


Figure 5.2.1: Partial block diagram of the encryptor platform

The encryptor platform is fundamentally the same as the forwarding platform from section 3.3 with the addition of the same data alignment blocks used in the link snooper and the new encryptor module (see figure 5.2.1).

5.2.1 Data Flow

In order to make a decision what data should be encrypted, decrypted or just passed along the encryptor module uses two finite state machines. The FSMs uses the stream content as well as meta data collected by the stream decoder to keep track of what kind of data is currently flowing. The output from the FSMs are fed to the crypto module which will either encrypt, decrypt, modify or pass the data along.

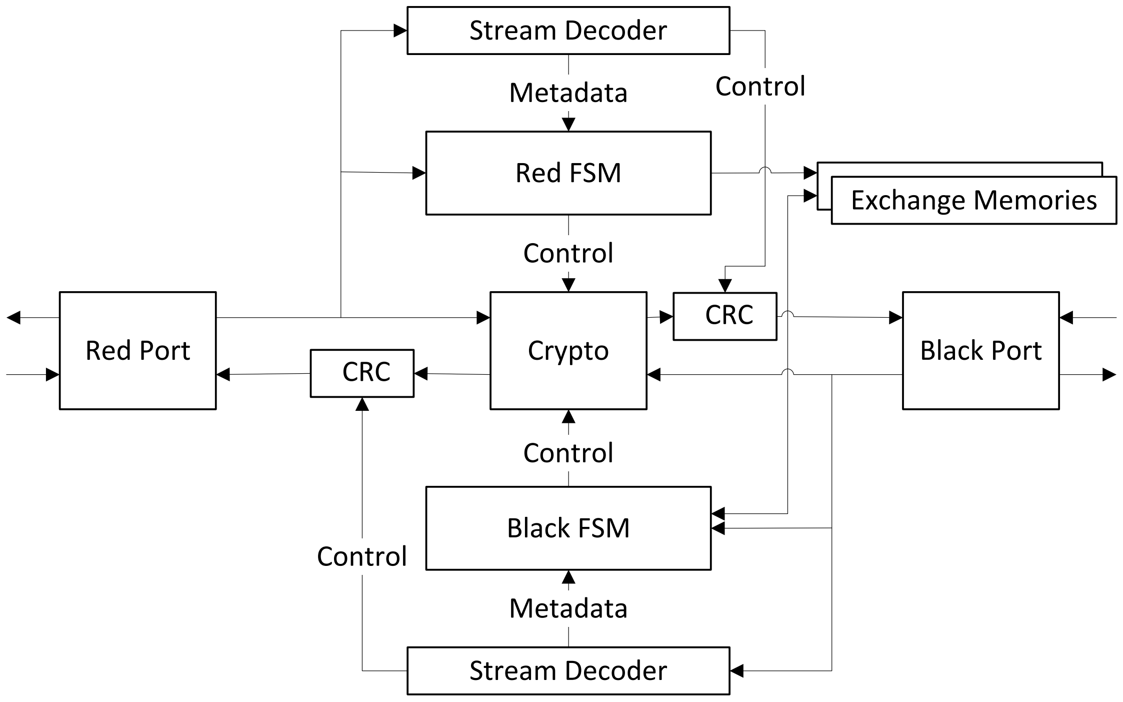


Figure 5.2.2: Block diagram of the encryptor module

Using the Link Snooper it was shown that to encrypt the disk traffic in the test lab only two SCSI commands need modification: Read(10) and Write(10). In extension to this the platform allows the identification string for the device to be modified by changing a structure called CDB.

Since the frames might be modified some care needs to be taken in order to make sure that the checksum is updated to reflect the new frame contents. To make the implementation more straight forward a new checksum is calculated for the frame regardless if it is modified or not. This has the implication that should a corrupt frame reach the encryptor it will not be discarded by rather updated with a correct checksum for the faulty data. This behaviour will result in that invalid data is stored but for this thesis that does not matter.

5.2.2 Encryption

The encryption algorithm used is a simple XOR with a constant key. This encryption is useless in real world applications since it is trivial to reverse engineer the key and decrypt the data.

This simple crypto was chosen since it does not need an initialisation vector and uses a static key. The problem with initialisation vectors are that they need to be stored with the data making the encryption process much more invasive to the data.

5.2.3 Synthesis Statistics

The tables 3.3.1 and 5.2.1 contains the number of logic resources used in the implementation. The first implementation contains very little logic since its purpose is to only forward data while the final implementation contains much more logic. The final implementation also uses a few memory blocks to store meta data about the exchanges.

The resource utilization column shows that the FPGA has a lot of free resources should one wish to extend the implementation further.

Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	706	28,800	2%
- Number used as Flip Flops	706		
Number of Slice LUTs	696	28,800	2%
- Number used as logic	610	28,800	2%
– Number using O6 output only	555		
– Number using O5 output only	29		
– Number using O5 and O6	26		
- Number used as Memory	83	7,680	1%
– Number used as Shift Register	83		
— Number using O6 output only	83		
- Number used as exclusive route-thru	3		
Number of route-thrus	32		
- Number using O6 output only	32		
Number of occupied Slices	375	7,200	5%
Number of LUT Flip Flop pairs used	974		
- Number with an unused Flip Flop - Number with an unused LUT	268 278	974 974	27% 28%
- Number of fully used LUT-FF pairs	428	974	43%
- Number of unique control sets	19		
- Number of slice register sites lost to control set restrictions	23	28,800	1%
Number of bonded IOBs	24	480	5%
- Number of LOCed IOBs	24	24	100%
- Number of bonded IPADs	12		
– Number of LOCed IPADs	8	12	66%
- Number of bonded OPADs	12		
– Number of LOCed OPADs	10	12	83%
Number of BlockRAM/FIFO	4	60	6%
- Number using BlockRAM only	4		
– Number of 36k BlockRAM used	4		
- Total Memory used (KB)	144	2,160	6%
Number of BUFG/BUFGCTRLs	12	32	37%
- Number used as BUFGs	12		
Number of DCM_ADVs	4	12	33%
Number of GTP_DUALs	3	6	50%
Average Fanout of Non-Clock Nets	3.53		

Table 5.2.1: Resource usage for the forwarding platform with encryption

Chapter 6

Results

This chapter contains discussions of the implemented hardware platform, its performance as well as any limitations.

6.1 Measurement Setup

The computer hosting the Fibre Channel LUNs is called the server. The computer accessing the server over Fibre Channel is called the client.

Normally computer A is the server and computer B is the client but in some measurements the roles have been swapped to test what effects the computer hardware have on the measurements. When the opposite configuration is used the graphs have been marked with “reversed” or “rev.”.

6.1.1 Computer A

- HP dx5150 MT
- AMD Athlon(tm) 64 X2 Dual Core Processor 3800+
- 3 GiB DDR I RAM
- QLogic QLE2460 4Gb Fibre Channel to PCI Express HBA
- Debian Wheezy 7.0 64-bit, Linux 3.2.0 with SCST patch for SCSI pass through

6.1.2 Computer B

- HP Compaq 6200 Pro MT PC
- Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
- 4 GiB DDR3 RAM
- QLogic QLE2460 4Gb Fibre Channel to PCI Express HBA
- Debian Wheezy 7.0 64-bit, Linux 3.2.0

6.2 Baseline Performance

Data throughput is measured using a Linux tool called dd (appendix A.1.1). The dd tool is configured with various block sizes in order to find the most efficient block size and what kind of performance impact the platform induces.

Maximum theoretical Fibre Channel data throughput is $\frac{Baudrate}{10}$ since the data is 8b/10b encoded. This gives a maximum theoretical data throughput of 425 MB/s and 212.5 MB/s for 4GFC and 2GFC respectively. To be able to sustain these data rates the backing storage was chosen to be 2 GiB RAM which also contrary to disk based storage provides constant speed.

Figure 6.2.1 shows the read performance of the RAM memory. The variations in read performance are not important since the throughput is well above the 425 MB/s that is required.

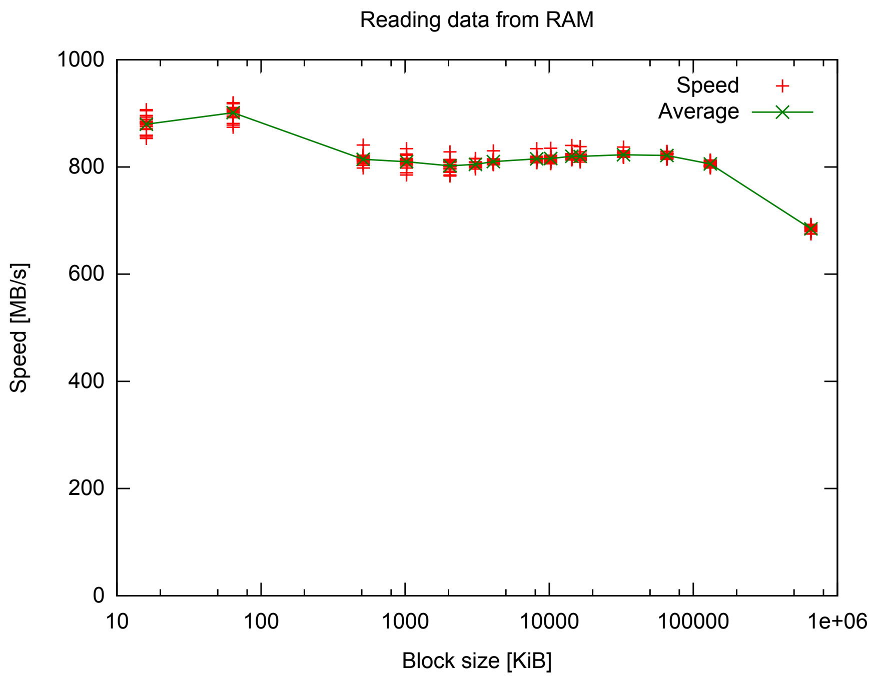


Figure 6.2.1: Reading data from RAM

6.3 Fibre Channel Performance

Running the same script (appendix A.1.1) on the same RAM memory but over Fibre Channel shows (figure 6.3.1) how block size affects data throughput. There is a noticeable discontinuity around 500 KiB as well as high variations of throughput around 10 MiB. The graphs (figure 6.3.1) shows that we want to have as large buffer as possible which makes those oddities irrelevant for our cause but for the sake of reasoning further measurements were done to investigate this. From figure 6.3.2 it is trivial to see that the variations are computer dependent and not inherent to the Fibre Channel protocol.

A comparative plot can be seen in figure 6.3.3 illustrating that the baseline RAM performance is well above that of 2GFC. In section 3.3 it was stated that 2GFC is the speed which the platform implementation can handle and therefore the measurements will from now on be using this speed.

The paper Yu Zhang, Dan Feng, Wei Tong; Jingning Liu, "Implementation of FC-1 and FC-2 Layer for Multi-Gigabit Fibre Channel Transport",[Yu Zhang(2007), Figure 5. Read I/O Performance] presents results resembling those presented here.

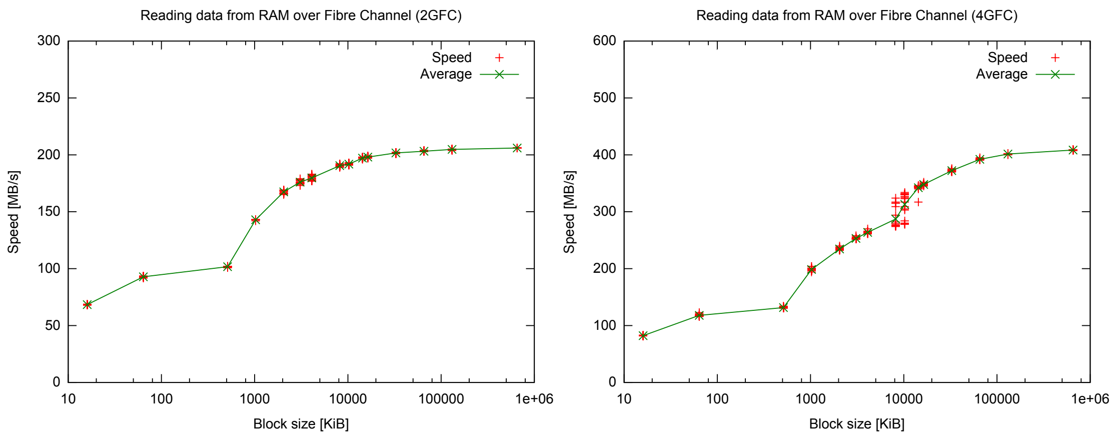


Figure 6.3.1: Reading data from RAM over Fibre Channel

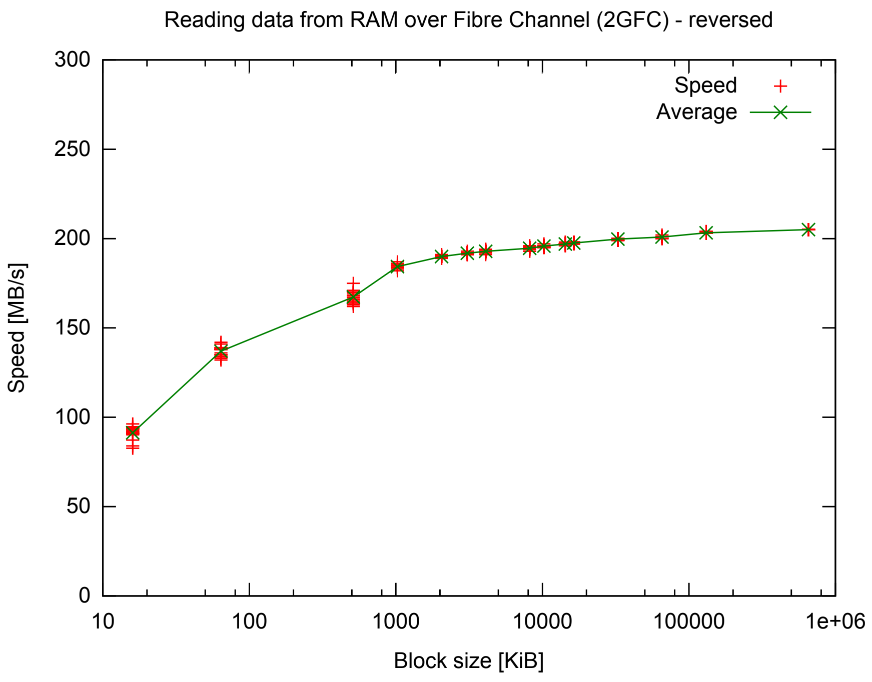


Figure 6.3.2: Reading data from RAM over Fibre Channel reversed

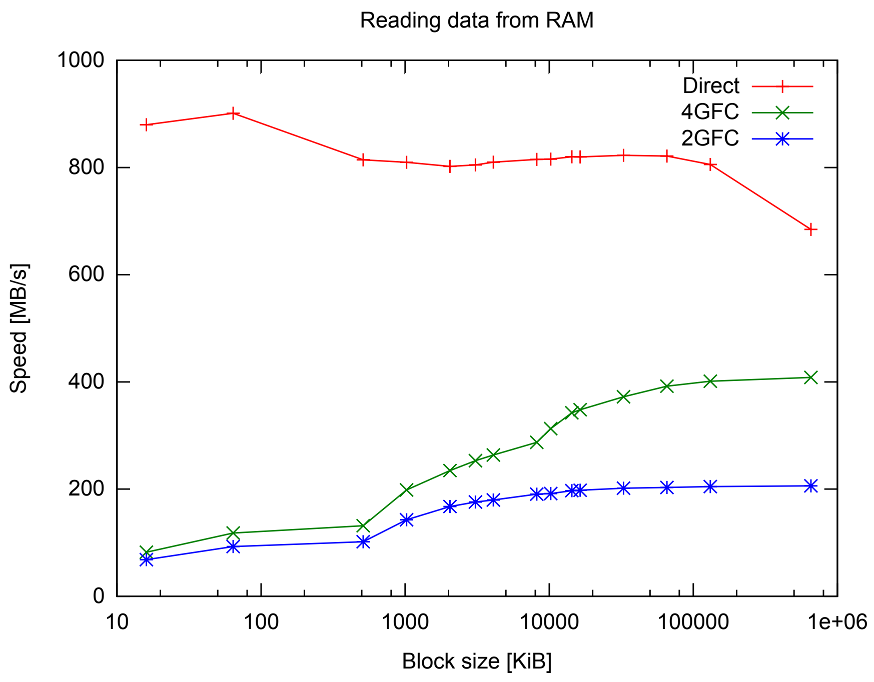


Figure 6.3.3: Reading data from RAM (Summary)

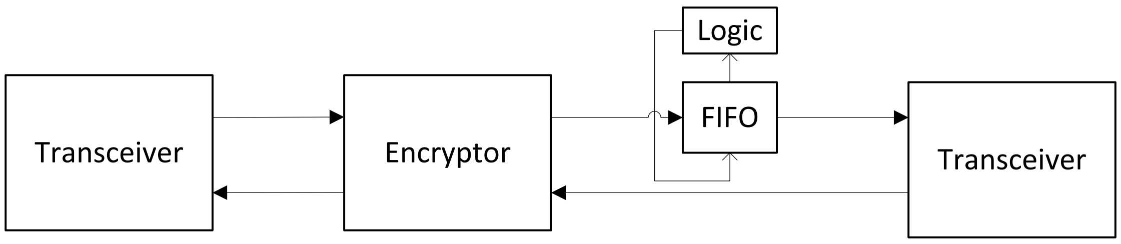


Figure 6.4.1: Latency block diagram

6.4 Latency and Performance Impact

With the results from section §6.3 it has been shown what kind of real world performance we can expect from Fibre Channel. This section will focus on showing the performance impact when adding latency to the system. This is interesting since the platform will introduce extra latency in one way or another. Looking at how extreme latencies the protocol can handle and still perform reasonably well will help to understand what kind of requirements must be put on, for example, the encryption algorithm.

Latency is said to be zero when not having the platform connected at all and increased from that point with a constant latency representing the inherent delay of the implementation. In addition a latency machine has been constructed (see figure 6.4.1) to induce even more latency. The extra latency is measured in the amount of clock cycles the data is held which for this implementation is clocked at 106.25 MHz.

6.4.1 Basic Latency

Running the tests over a link with extra latency produced some unexpected results. When transferring larger blocks of around 1 MB. the direction of traffic on the Fibre Channel link is almost unidirectional since one request will produce much more data than for smaller blocks. This hints that the effect of latency should be lower for bigger blocks and higher for smaller blocks. Figure 6.4.2 shows that this is not the case but rather the opposite. Smaller blocks stay about the same in throughput while larger blocks become alarmingly slow at latencies as low as $2048\,cc \approx 19\,\mu s$.

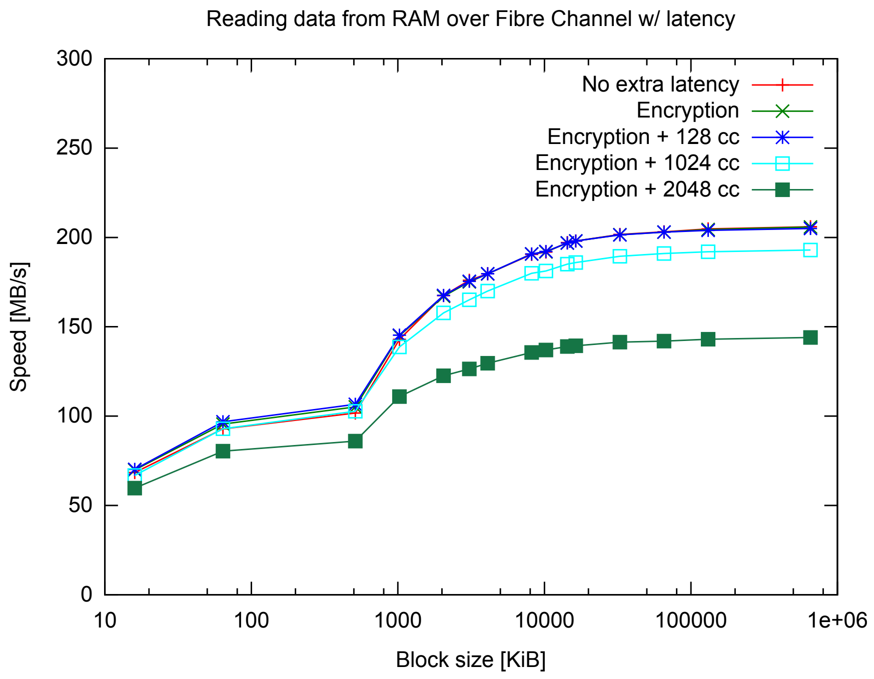


Figure 6.4.2: Reading data from RAM over Fibre Channel with latency (Summary)

6.4.2 Frame Sizes and Latency

The Fibre Channel HBA under test allows the user to specify the maximum payload to be transmitted inside the frames. The options are 512, 1024 and 2048 bytes per frame. Since each frame needs to contain flags, addresses and various control data it incurs an unwanted overhead. The size of this overhead is constant and does not change with the frame size making a frame carrying only 512 bytes of data less effective than one carrying 2048 bytes of data. With this in mind it is no surprise to see that the left-hand plot of figure 6.4.3 shows exactly this behaviour. Larger frames reach higher throughput than the smaller frames.

However, increasing the latency to 2048 clock cycles changes this to an odd middle ground. The right-hand plot of figure 6.4.3 shows that in this case using 1024 byte payloads are the optimal configuration and that using 2048 bytes incur an noticeable throughput penalty.

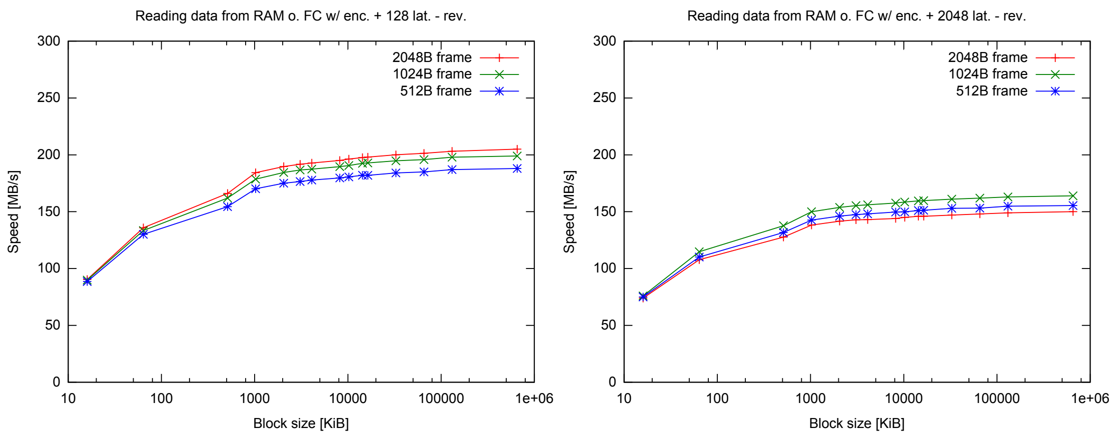


Figure 6.4.3: Reading data over Fibre Channel with varying latency and frame sizes

6.4.3 Asymmetrical versus Symmetrical Latency

As shown in figure 6.4.1 the latency introduced to the system is asymmetrical. To test the hypothesis that the earlier results are due to this fact, one of the tests were run using both asymmetrical and symmetrical latency. The results (see figure 6.4.4) show that there are no differences between asymmetrical or symmetrical latency.

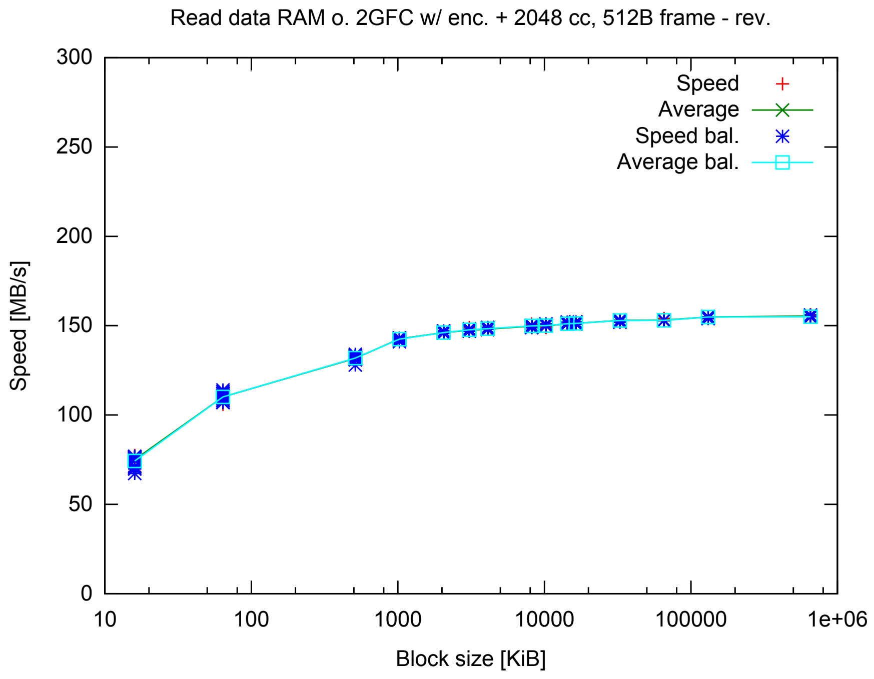


Figure 6.4.4: Comparison of symmetric and asymmetric latency

6.4.4 Summary

The cause of these non intuitive results was never fully understood but one hypothesis is that this is hardware dependent and that the cards or their drivers have a maximum number of frames in transmission and that this somehow limits performance under heavy latency.

One approach for this is to replace part by part with other vendors and see if the effects go away, but for this thesis that does not matter. The thesis focus is platform independent and the testing setup used is far from exotic in the industry, making the results relevant even though it may be hardware related rather than the protocol itself.

Chapter 7

Discussion

7.1 Security Implications

During the implementation phase many choices were made regarding how the data flow should be designed. Some choices greatly effect the integrity of the data flowing through the system, these are explored below.

7.1.1 Open/Secure Cut

One problem with transparent streaming encryption for permanent storage is that the endpoints must still be able to understand the commands sent. This means we cannot encrypt all data on the link but only a subset of the data being exchanged.

Ideally we would like to transform the incoming request to an uncorrelated outgoing request e.g. by buffering many writes/reads or scattering the data to many different places. The implementation of this is very complex and far out of the scope of this thesis. Having this ideal in mind it is easy to see that the line between the black and red side is not absolute in the current implementation. It is trivial to correlate the incoming frames to the outgoing frames and this results in attack vectors that are inherent to this design. For example it might be possible to swap contents of two sectors by manipulating the address inside the request.

7.1.2 Encryption Exchange Record

The implementation uses a memory to keep track of the Fibre Channel exchanges that needs encryption. If the hardware were to experience a bit flip in the memory it could cause an exchange that should have been encrypted to be passed on to the storage without encryption.

This is again inherent in a design where some traffic is modified while some are not.

7.1.3 Unknown Data

When the incoming data is streamed through the hardware the forwarding engine has to make a decision if this frame belongs to an exchange that must be encrypted or decrypted. If the forwarding engine does not recognise the frame layout it will pass it on unmodified potentially leaking sensitive data.

7.2 Encrypted Transport

A solution based on the concept of encrypting the whole link using two units in a point-to-point fashion will not have any of these earlier stated security implications. This is the case since the

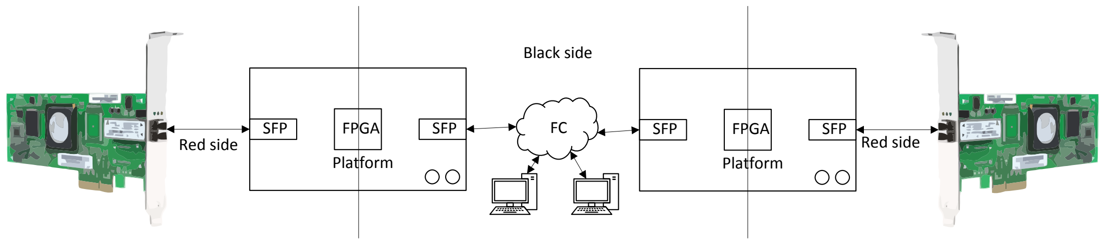


Figure 7.2.1: Encrypted transport implementation

encryption units can chose any form of data representation between themselves and for example chose to encrypt the whole frames between the units.

Note that using an encrypted transport will provide safe exchange of data but not storage of it. With that knowledge it is easy to see that the two unit encrypted transport scenario is fundamentally different from the single unit scenario which has been the focus for investigation in this thesis. Nevertheless the work herein can be used to implement an encrypted transport that will preserve the Fibre Channel protocol which can be useful if the target infrastructure is shared with non-encrypted endpoints.

A concept illustration of how the platform can be used for this scenario is illustrated in figure 7.2.1.

The paper L. Henzen, F. Carbognani, N. Felber, W. Fichtner, "FPGA Implementation of a 2G Fibre Channel Link Encryptor with Authenticated Encryption Mode GCM" shows an implementation of such a system[L. Henzen(2008), IV.A. Frame Encryption].

7.3 Conclusion

This thesis focused on testing whether persistent encryption of Fibre Channel is doable and what kind of security it provides. It has been shown that intercepting, analysing and modifying Fibre Channel traffic is possible without any noticeable performance loss as long as latency is kept within certain boundaries. If latency are outside those boundaries extreme performance loss are to be expected.

This latency demand puts further restrictions on the cryptography to be used. In the case of cut-through forward based encryption the cryptography algorithm has to work with only the data available in a read or write request and may not, for example, inject an initialisation vector without careful consideration.

Chapter 8

Future Work

The security limitations in the implementation has been discussed in section 7.1. These limitations can be solved with a more distinct separation between the red and black sides, for example by terminating the SCSI commands in the platform and generate new ones to the hard drive. This opens up the possibility of having a customised file system and meta data sections capable of storing e.g. initialisation vectors. With big caches and carefully designed read ahead logic for the meta data, the latency of an extra disk read and write could probably be minimised.

In the beginning of this thesis the goal was to implement the platform for 16GFC but due to available hardware 2GFC was chosen instead. The differences between the two are very straight forward and the platform should be easy to port to 16GFC.

For this thesis the choice of using cut-through forwarding was made. It would be interesting to see what kind of behaviour Fibre Channel exhibits when faced with non-constant latency, for example by using store and forward.

Appendix A

Source Code

A.1 Measurement

A.1.1 blk_scan.sh

```
#!/bin/bash

for j in ‘seq 1 20‘
do
        :> blk.data
        for i in 16 64 512 1024 2048 3072 4096 8192 10240 \
          14336 16384 32768 65536 131072 655360
        do
                RES=‘dd if=/dev/$1 of=/dev/null iflag=direct \
          bs=${i}K 2>&1 | grep copied | cut -f 8 -d ’ ’‘
                echo $i $RES >> blk.data
        done

        mkdir -p data/$j/
        cp blk.data data/$j/
done
```

Bibliography

[Juniper Networks(2012)] Inc. Juniper Networks. Latency: Not all numbers are measured the same, 2012. URL http://www.juniper.net/us/en/local/pdf/whitepapers/2000405-en.pdf.

[L. Henzen(2008)] N. Felber W. Fichtner L. Henzen, F. Carbognani. Fpga implementation of a 2g fibre channel link encryptor with authenticated encryption mode gcm, 2008.

[Seagate(2006)] Seagate. *SCSI Commands Reference Manual.* Seagate, February 2006. URL http://www.seagate.com/staticfiles/support/disc/manuals/scsi/100293068a.pdf.

[T11 Technical Committee(2010)] T11 Technical Committee. Fibre channel - framing and signaling - 3, October 2010. URL http://www.t11.org/ftp/t11/pub/fc/fs-3/10-010v3.pdf.

[T11 Technical Committee(2011)] T11 Technical Committee. Fibre channel - physical interface - 5, January 2011. URL http://www.t11.org/ftp/t11/pub/fc/pi-5/11-011v0.pdf.

[Ulf Troppens(2009)] Wolfgang Müller-Friedt Nils Haustein Rainer Wolafka Ulf Troppens, Rainer Erkens. *Storage Networks Explained: Basics and Application of Fibre Channel SAN, NAS, ISCSI, InfiniBand and FCoE.* John Wiley & Sons Inc, 2009.

[Yang Yang(2012)] Cisco Yang Yang. Understanding switch latency, 2012. URL http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps11541/white_paper_c11-661939.pdf.

[Yu Zhang(2007)] Wei Tong; Jingning Liu Yu Zhang, Dan Feng. Implementation of fc-1 and fc-2 layer for multi-gigabit fibre channel transport, 2007.