Master Thesis

Test and Signalling of a 40Gbps Transmitter/Receiver Prototype

by

A. B. Christian Hansen
c956811

February 10, 2003

Supervisors:
Steen Pedersen
Flemming Stassen
Technical University of Denmark
Jesper Birch
Ole Rassing Andersen
Tellabs Denmark A/S

Abstract

Testing digital hardware often requires a high speed transmission and reception of binary data. The transmitted test signal has to simulate the random characteristic of a digital signal, but at the same time it has to be predicable. Pseudo random binary sequences (PRBSs) fulfills just this, and are widely used for transmission tests. This thesis investigates the feasibility of implementing a 40 Gb/s PRBS test module, based on standard FPGAs, capable of generating test signals for a 40 Gb/s multiplex/demultiplex module. The entire design process, ranging from the initial overall demands, to the final tests conducted on the hardware, will be described. Error free transmission and reception of a $2^{31} - 1$ PRBS on several of the data channels, will be demonstrated. It will be shown, that FPGAs indeed are viable choices for designs, that require high speed transmission and reception of data across a parallel interface.

Resumé

Test af digitalt hardware kræver ofte en højhastighedstransmission og -modtagelse af binær data. Det transmitterede signal skal simulere karakteristikken af et tilfældigt digitalt signal, men samtidigt skal det være forudsigeligt. Maksimallængdesekvenser (MLS'er) opfylder netop dette, og er meget udbredte inden for transmissionstests. Dette eksamensprojekt undersøger muligheden for at implementerer et 40 Gb/s MLS test modul, baseret på standard FPGA'er, der er i stand til at generere testsignaler til et 40 Gb/s multiplex/demultiplex modul. Hele designprocessen, fra de indledende overordnede krav, til de afsluttende tests af hardwaren, vil blive beskrevet. Fejlfri transmission af en $2^{31} - 1$ MLS, på flere af datakanalerne, vil blive demonstreret. Det vil blive vist, at FPGA'er bestemt er en brugbar valgmulighed, til designs der kræver højhastighedstransmission og -modtagelse af data over en parallel snitflade.

List of Symbols

Symbol		**Description**	**Unit**
N	:	Number of registers in a linear feed back shift register	Dimensionless
f	:	Frequency	Hz
n	:	Integer	Dimensionless
R	:	Decimation factor	Dimensionless
$R(m)$	:	Correlation between two pseudo random binary sequences	Dimensionless
$R_{\#}$	:	Resistor nr. #	Ohm
T	:	Bit period	s
t_{rf}	:	Signal transition time	s
T_{skew}	:	Device to device skew	s
T_{sync}	:	Synchronization time	s
T_{trace}	:	Skew resulting from differences in PCB trace lengths	s
$T_{2.5Gb/s}$	:	Bit period of a 2.5 Gb/s binary signal	s
T_{valid}	:	Valid data window	s
V_{cc}	:	Power supply voltage	V
$X_{\#}$	:	Binary digit nr. #	Dimensionless
$X^{\#}$	:	Polynomial description of a binary digit nr. #	Dimensionless
$x(k)$	:	The k'th bit in a pseudo random binary sequence	Dimensionless
$y(k+m)$	:	The k'th bit in a pseudo random binary sequence shifted by m bits with respect to $x(k)$	Dimensionless
Z	:	Impedance	Ohm
Z_0	:	Characteristic impedance	Ohm

Abbreviations and Glossary

Abbreviations

BER	:	Bit Error Rate
CDR	:	Clock and Data Recovery
CLB	:	Configurable Logic Block
CML	:	Current Mode Logic
CMU	:	Clock Management Unit
DCM	:	Digital Clock Manager
DDR	:	Double Data Rate
DPA	:	Dynamic Phase Alignment
ECL	:	Emitter Coupled Logic
IOB	:	Input-Output block
LFSR	:	Linear Feed back Shift Register
LSB	:	Least Significant Bit
LVDS	:	Low Voltage Differential Signaling
MSB	:	Most Significant Bit
OTU	:	Optical Transport Unit
PCB	:	Printed Circuit Board
PFC	:	Phase Frequency Comparator
SDH	:	Synchronous Digital Hierarchy
STM###	:	Synchronous Transmission Module
PECL	:	Positive Emitter Coupled Logic
PRBS	:	Pseudo Random Binary Sequence
UART	:	Universal Asynchronous Receiver Transmitter
UI	:	Unit Interval
VCSO	:	Voltage Controlled Saw Oscillator
VHDL	:	Very high speed circuit Hardware Description Language

Glossary

AC coupling: Electrical connection which blocks the DC component from the signal. The resulting signal can then be biased to any voltage.
Bandwidth: Bandwidth is the frequency limits imposed by components (both electrical and optical) within which the component exhibits limited attenuation. For electrical components the attenuation limit is normally -3dB from minimum attenuation (equivalent to 70% of maximum output amplitude).
Bit error rate: Bit error rate is the ratio between the number of incorrect bits received and the number of bits transmitted. The BER is a performance parameter which describes the transmission. The same BER at a lower power level indicates a better transmission (ie. less energy is required to perform the same task).
DC/DC converter: Integrated power supply unit. It converts a DC voltage to another, either fixed or programmable. DC/DC converters are more efficient than linear regulators, since the latter only converts excess power to heat. Another feature is that the DC/DC converter are/can be isolated electrically from input to output.
Double Data Rate: Technique where both edges of a clock signal is used for transmitting or receiving data. ***FPGA*** : A field-programmable gate array (FPGA) is an integrated circuit (IC) that can be programmed in the field after manufacture. FPGAs are similar in principle to, but have vastly wider potential application than, programmable read-only memory (PROM) chips.
Jitter: Dynamic phase offset. Jitter is referenced to the mean value of the same signal.
LFSR : Shift register, where some of the register outputs are modulo-2 added and fed back to the input of the first register. LFSRs are state machines, and are used to generate PRBSs.
PLL: Phase-Locked Loop. A circuit which controls that one voltage controlled oscillator is of same phase and frequency as an incoming reference. The VCO controlled in this way is usually of higher frequency than the reference clock signal, and the output signal is therefore divided before the comparison. Locking oscillators in this way is primarily to avoid skipping bits in the data transfer.
PRBS : A random binary sequence that repeats it self. PRBSs are generated using a Linear Feed back Shift Register (LFSR), and are widely used for testing digital hardware for communication.
PROM : Programmable read-only memory (PROM) is read-only memory (ROM) that can be modified once by a user. PROM is a way of allowing a user to tailor a microcode program using a special machine called a PROM programmer. This machine supplies an electrical current to specific cells in the ROM that effectively blows a fuse in them. The process is known as

burning the PROM. Since this process leaves no margin for error, most ROM chips designed to be modified by users use erasable programmable read-only memory (EPROM) or electrically erasable programmable read-only memory (EEPROM).
Skew: Skew is a static phase offset between two signals, generally referenced to the clock.
Transmission line: An electrical connection with a specified geometry resulting in a known high speed characteristic. The main force of the transmission line is that it appears as a discrete resistor to the transmitting circuit, regardless of the length of line and transmission frequency.
Via: Connection between the different layers of a PCB design. Some vias may be designed as buried, only connecting the layers on which the signal is routed. The most common type is however vias that pass through all layers. The via is a hole drilled through the PCB and lined with conductive material to generate an electrical path.

Numbers in square brackets are literature references. The bibliography is placed in the end of the document.

Preface

This report is a master thesis written at the Technical University of Denmark (DTU) in conjunction with Tellabs Denmark A/S, and is the result of a 10 month master of science project. The project was carried out in the period 01.04.02 - 10.02.03.

The supervisors on this project were :

Jesper Birch, Tellabs Denmark A/S
Ole Rassing Andersen, Tellabs Denmark A/S
Flemming Stassen, Technical University of Denmark
Steen Pedersen, Technical University of Denmark

Christian Hansen

Acknowledgements

The author would like to express his appreciation to Ole Rassing Andersen from Tellabs Denmark A/S, for technical guidance during this project. Furthermore, the author would like to thank Steen Pedersen and Flemming Stassen from the Technical University of Denmark, for handling the practical aspects concerning this project. A special thanks to Kjeld Aage Dalgaard for report technical support.

Contents

List of Symbols **vii**

Abbreviations and Glossary **ix**
Abbreviations ix
Glossary x

Acknowledgements **xv**

1 Introduction **1**
1.1 Project Overview 2
1.2 Document outline 3

2 Pseudo Random Binary Sequences **5**
2.1 Shift Register Generation of Pseudo Random Binary Sequences 5
2.2 Properties of Pseudo Random Binary Sequences 7
2.2.1 Shift and add 8
2.2.2 Subsequences 9
2.2.3 Correlation 10
2.2.4 Acquisition of Pseudo Random Binary Sequences . . . 10
2.2.5 Power Spectrum of Pseudo Random Binary Sequences 11
2.3 Summary 12

3 Test module description and requirements **13**
3.1 40 Gb/s Interface Specification 13
3.1.1 Definitions 13
3.1.2 Description 14
3.1.3 Signal levels 14
3.1.4 Jitter 15
3.1.5 Timing 15
3.1.6 Physical backplane 15
3.2 Functional Block Diagram 16
3.2.1 Oscillator 16
3.2.2 Clock divider 18
3.2.3 Transmitter FPGA 18

3.2.4 PLL CLK Synthesizer 19
3.2.5 Delay line 19
3.2.6 Multiplex stage 20
3.2.7 Demultiplex Stage 20
3.2.8 Receiver FPGA 21
3.3 Summary 21

4 Test module design 23
4.1 Technologies 24
4.1.1 LVPECL 24
4.1.2 CML 25
4.1.3 LVDS 26
4.2 Interfacing between LVDS, CML and LVPECL 28
4.2.1 CML to LVPECL 28
4.2.2 LVPECL to LVDS 30
4.3 Component selection 31
4.3.1 Oscillator 32
4.3.2 Clock divider 32
4.3.3 Transmitter FPGA 33
4.3.4 PLL clock synthesizer 34
4.3.5 Multiple output clock divider 34
4.3.6 Delay line 35
4.3.7 Multiplex Stage 35
4.3.8 Demultiplex Stage 36
4.3.9 High fan-out clock buffers 38
4.3.10 Receiver FPGA 38
4.4 Test Considerations 38
4.4.1 Test equipment 39
4.4.2 Test pads and signals 40
4.5 Schematic Layout 42
4.5.1 The Power Supply 43
4.5.2 Control and Status Signals 45
4.5.3 Test pads and connectors 46
4.5.4 Miscellaneous 46
4.6 Printed Circuit Board 46
4.7 Summary 48

5 FPGA Design 51
5.1 General 52
5.2 4 bit transmitter macro 52
5.3 4 bit receiver macro 54
5.4 Top Entity for the TX FPGA 56
5.4.1 VHDL model for the PRBS Generator 58
5.4.2 VHDL model for the 64 bit transmitter 59

5.5 Top Entity for the RX FPGAs . . . 61
5.5.1 VHDL model for the 32 bit receiver . . . 63
5.5.2 VHDL model for the PRBS receiver . . . 64
5.6 Simulation and test . . . 66
5.6.1 Simulation of the TX FPGA design . . . 67
5.6.2 Simulation of the RX FPGA design . . . 68
5.7 Implementation . . . 70
5.7.1 Synthesis . . . 70
5.7.2 Place and Route . . . 70
5.8 Summary . . . 72

6 Test module test 73
6.1 Power Supply Test . . . 74
6.2 Clock Distribution Test . . . 75
6.3 Data transmission and acquisition test . . . 80
6.3.1 Data transmission test . . . 80
6.3.2 Data acquisition test . . . 83
6.3.3 Error count test . . . 86
6.4 Improvement Suggestion . . . 86
6.4.1 Design errors . . . 87
6.5 Summary . . . 87

7 Conclusion 89

A Schematic 93

B Component data sheets 109
B.1 Data sheet for the 622.08 MHz reference clock oscillator . . . 110
B.2 Data sheet for the MC100EP32 clock divider . . . 113
B.3 Data sheet for the GD16590 PLL clock synthesizer . . . 126
B.4 Data sheet for the MC100LVEL37 clock divider . . . 139
B.5 Datasheet for the MC100EP195 delay line . . . 148
B.6 Data sheet for the LXT16653 multiplexer . . . 165
B.7 Data sheet for the LXT16642 demultiplexer . . . 182
B.8 Data sheet for the NB100LVEP221 high fan-out clock buffers 197

C S4 Backplane information 211

D XAPP 265 application note from Xilinx 215

E Results from the TX FPGA simulation 229
E.1 Waveforms from the simulation of the PRBS generator . . . 230
E.2 Waveforms from the simulation of the 64 bit transmitter . . . 233
E.3 Waveforms from the simulation of the delay line configuration settings and the error transmitting functionality . . . 235

F Results from the RX FPGA simulation 239
F.1 Waveforms from the simulation of the 32 bit receiver and the error counting functionality of the PRBS receivers 240
F.2 Waveforms from the simulation of the PRBS receiver 242
F.3 Waveforms showing a closeup of the error count output from the PRBS receivers 244
F.4 Waveforms showing the functionality of the receiver DCMs . 246
F.5 Waveforms showing that the reset signals function correctly . 248

G UCF files for the TX and RX FPGA designs 251
G.1 Floor plan view of the FPGA designs 266

H Test Report 269

List of Figures

1.1 System overview . . . 3

2.1 General PRBS generator . . . 6
2.2 LFSR generating a PRBS of length $2^4 - 1$. . . . 8
2.3 State diagram for a LFSR generating a PRBS of length $2^4 - 1$ 9
2.4 Decimation of one PRBS into two identical PRBSs at half the rate . . . 10
2.5 Basic PRBS Receiver . . . 11
2.6 Error compensating PRBS receiver . . . 12
2.7 Power spectrum for a PRBS . . . 12

3.1 40 Gb/s Interface . . . 14
3.2 Block diagram illustrating the requirements for the test module 17

4.1 PECL output stucture . . . 24
4.2 CML output stucture . . . 26
4.3 CML output current . . . 27
4.4 CML input stucture . . . 28
4.5 LVDS output stucture . . . 29
4.6 CML to LVPECL interface . . . 29
4.7 LVPECL to LVDS interface . . . 31
4.8 Final test module block diagram . . . 33
4.9 Block diagram for the LXT16653 multiplexer . . . 37
4.10 Loop filter adjustment circuitry . . . 45
4.11 Top view of the test module PCB . . . 49
4.12 Bottom view of the test module PCB . . . 50

5.1 Four bit transmitter macro entity . . . 52
5.2 Bit order on the 4 bit transmitter macro output . . . 53
5.3 Physical implementation of the TX macro . . . 54
5.4 Four bit receiver macro entity . . . 55
5.5 Physical implementation of the 4 bit receiver macro . . . 56
5.6 Functional description of the TX FPGA design . . . 57
5.7 LFSR that generates 512 bits of PRBS every clock cycle . . . 59

5.8 Bit order on the 64 bit transmitter output 61
5.9 RX FPGA Top Entity 62
5.10 Illustration of the VHDL model for the PRBS receiver 65

6.1 General Test Setup 75
6.2 1.25 GHz reference clock 76
6.3 622 MHz trigger signal 77
6.4 TX FPGA input clock 77
6.5 Input clock for the PLL clock synthesizer 78
6.6 Input clock for the MUX stage 79
6.7 1.25 GHz CML clock signal for the 40 Gb/s interface 80
6.8 622 Mb/s PRBS at the input of one of the multiplexers in the MUX stage 81
6.9 2.5 Gb/s PRBS at the output of one of the multiplexers in the MUX stage 82
6.10 2.5 Gb/s output data from multiplexer 82
6.11 622 Mb/s PRBS at a true input of a RX FPGA 84
6.12 622 Mb/s PRBS at an inverted input of a RX FPGA 84
6.13 311 MHz clock signal at the input of one of the RX FPGAs . 85

C.1 Illustration of the S4 backplane 213

List of Tables

1.1 Nominal SDH frequencies used within this project 3

2.1 Number of runs of various lengths for a $2^N - 1$ bit long PRBS 7

4.1 LVPECL input and output specifications 25
4.2 Overview of the schematic layout 42
4.3 Power supply requirement 43
4.4 Current requirement for the 3.3 V and 1.5 V power supplies . 44

5.1 Input and output description for the 4 bit transmitter macro 53
5.2 Input and output description for the 4 bit receiver macro . . 55
5.3 Input and output description for the top entity of the TX FPGA design 58
5.4 Input and output description for the PRBS generator entity . 59
5.5 Input and output description for the 64 bit transmitter 60
5.6 Input and output description for the top entity of the RX FPGA design 63
5.7 Input and output description for the 32 bit receiver 64
5.8 Description of the PRBS receiver entity 65
5.9 Test stimuli for the error_set_in signal 67
5.10 Test stimuli for the delay_set_in signal 68

6.1 Tests conducted on the test module 74

E.1 Signal descriptions for the TX FPGA simulation 237

F.1 Signal descriptions for the RX FPGA simulation 239

Chapter 1

Introduction

The rapid development of digital hardware used for telecommunication, drives the need for test systems capable of following the ever increasing data rates. Test systems have to emulate real traffic, and are used to test the performance of the digital hardware. Today, mature 2.5 Gb/s and 10 Gb/s transmission systems exist, thus the next natural step is to develop 40 Gb/s transmission systems.

Commercial test systems for testing 40 Gb/s systems exist[1, 2], but these test systems are very costly, and do not necessarily provide the interface needed to test a specific system. Thus when using commercial test systems, it is often required to facilitate the interface needed between the test system, and the system being tested, in some way. This can lead to complex test setups that are tedious and difficult to use. Moreover, commercial test systems often provide several functions that may be redundant to some users, which emphasizes the cost of these systems even more. This feeds the interest of developing test systems aimed at a specific use. Specifically designed test systems can save both time and money, as these systems would simplify the test setups, and shorten the time needed to perform the tests. In order for the development of such a test system to be economically viable, it has to be possible to implement the system using standard components.

A standard component of special interest for test systems used for transmission tests, is a field programmable gate array (FPGA). FPGAs have developed a lot over the last couple of years, and provide features today that are very desirable in test systems. Multiple high speed input/outputs, resulting in transmission and reception of data at very high rates, are some of the features that standard FPGAs provide. Moreover, FPGAs can be reconfigured, thus providing a flexibility which is very desirable in a test system. This allows the user to alter the test signal being transmitted, which can prove to be very useful for test purposes.

This thesis investigates the feasibility of implementing a 40 Gb/s test module, based on standard FPGAs, capable of generating test signals for a

40 Gb/s transmitter/receiver module. These test signals have to be delivered across a well defined back plane interface used within Tellabs. The test signals are transmitted in parallel to the transmitter/receiver module where they are multiplexed, and transmitted as a serial bit stream at 40 Gb/s. This 40 Gb/s serial bit stream, resulting from multiplexing of the test signals, has to constitute a pseudo random binary sequence (PRBS) with a sequence length of $2^{31} - 1$ bits.

Basing the design of the test module on the FPGA technology is not only interesting because of the properties FPGAs hold, but also because the FPGA technology is a well founded technology which is expected to keep developing. During the course of this project a new type of FPGA has been developed [3], which provides multiple on-chip microprocessors, and the capability to transmit and receive data at even higher data rates. Investigating the limits and usage of FPGAs in high speed communication systems is important, as it can lay down the foundation for future development of high speed data generating systems based on FPGAs, providing the advantages just described.

1.1 Project Overview

This master thesis is one of two master theses running in parallel, where the other thesis is concerned with the design and implementation of the 40 Gb/s multiplex/demultiplex module (MUX/DeMUX module). The 40 Gb/s MUX/DeMUX module transmits the 40 Gb/s serial electrical signal to an optical module (Opto module), which converts the electrical 40 Gb/s signal into an optical signal, and transmits it onto an optical fiber. After the signal has propagated through the optical fiber, the opto module receives the signal, converts it back into an electrical signal, and transmits it back to the MUX/DeMUX module. The MUX/DeMUX module demultiplexes the 40 Gb/s eletrical signal, and transmits the data back to the test module. The entire system is shown in figure 1.1. The RD-numbers shown in figure 1.1 are internal Tellabs designators. Throughout this thesis the names assigned to the modules in figure 1.1 will be used. The frequencies used by the modules in this system are standard SDH frequencies, but in-text references to these frequencies will be done using approximated values. Table 1.1 contains the standard nominal SDH frequencies[1] used in this project, together with the approximated values used in the text. This thesis describes the entire design process of the test module, including the testing of the test module. Initially a functional block diagram, based on the overall demands for the design, will be constructed. Requirements for each block in the block diagram will be specified. Based on these requirements, components will be

[1]The nominal data rate including overhead but excluding forward error correction scheme.

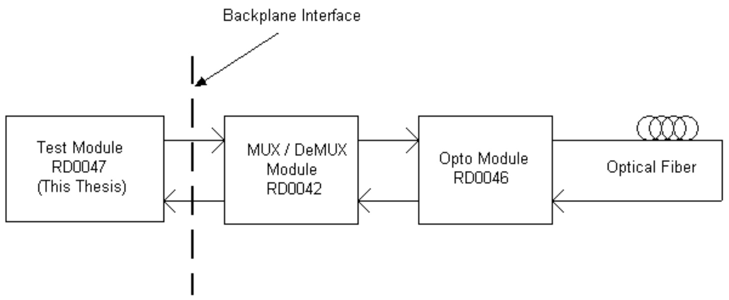


Figure 1.1: The figure shows the entire system relevant to this thesis.

Standard SDH [MHz]	Approximated [MHz]
38.88	39
77.76	80*
311.04	311
622.08	622
1244.160	1250
2488.320	2500
39813.120	40000

Table 1.1: Table containing the nominal SDH frequencies used within this project. * = this standard frequency will be referred to as 80 MHz, but the corresponding data rate will be referred to as 78 Mb/s.

selected, and a final functional block diagram for the test module will be presented. The process of constructing the initial block diagram based on the overall demands, is somewhat biased from the knowledge of the components available on the marked. It has been strived to keep the initial block diagram independent of the component selection, but knowledge about the available components, and their performance, unavoidably affects the design of the initial block diagram to some degree. Also part of this project is three FPGA designs, which will be described in detail.

1.2 Document outline

Chapter 2 introduces PRBSs and their basic properties.
In **Chapter 3**, a functional block diagram based on the overall demands for the design will be constructed, and requirements for each block in the block diagram will be specified.
In **Chapter 4** a brief description of the CML (Current Mode Logic), LVPECL (Low Voltage Positive Emitter Coupled Logic), and LVDS (Low Voltage Dif-

ferential Signalling) technologies will be given. The components selected for implementing the test module will be described, and a functional block diagram representing the final design will be presented. The schematic layout will be described, together with the considerations done in connection with the final test of the test module, and the PCB layout.
Chapter 5 describes the FPGA designs. The PRBS generator and receiver will be described together with the high speed logic needed to transmit and receive the PRBS. The simulation and test of the FPGA designs will be described, followed by a description of how the designs were implemented in the FPGAs.
Chapter 6 outlines and explains the tests conducted on the test module, and presents the test results obtained through these tests.

The appendices listed in the table of contents are included in a separate booklet. Also provided on a CD are VHDL design files, constraint files needed by the FPGA design tools, application notes used for the design of the test module, log files from the synthesis tool, the FPGA data sheets, and timing reports from the FPGA place and route tool.

Chapter 2

Pseudo Random Binary Sequences

Pseudo random binary sequences (PRBSs) are widely used for testing hardware for digital communication. Testing of hardware for digital communication requires transmission and reception of a signal that subjects the transmission channel to the characteristics of random digital signal. A PRBSs is a random bit sequence that repeats it self, thus not truly random, as the name implies. A truly random sequence never repeats it self, but truly random sequences are difficult to generate, and would have very little use in practical systems. However PRBSs with long sequence lengths (several billion bits) show close resemblance to truly random signals, and are sufficient for test purposes. PRBSs have well known properties, and the generation and acquisition of them are simple. Knowing how a PRBS is generated, makes it possible to predict the sequence. This is a very desirable feature when testing hardware for digital communication, as it allows you to predict how an incoming sequence is supposed to look. This makes it possible to register and count any errors that might occur in the sequence.

2.1 Shift Register Generation of Pseudo Random Binary Sequences

PRBSs can be generated by shifting bits through a number (N) of cascaded registers, where some of the register outputs (referred to as tap sets) are added modulo-2 and fed back to the input of the first register. The maximal length of the sequence is determined by the number of possible states that the shift register can assume, and the properties of the sequence is determined by which tap sets that are modulo-2 added and feed back to the first register. This type of PRBS generator is called a linear feedback shift register (LFSR), and figure 2.1 shows a general LFSR. Mathematically, the

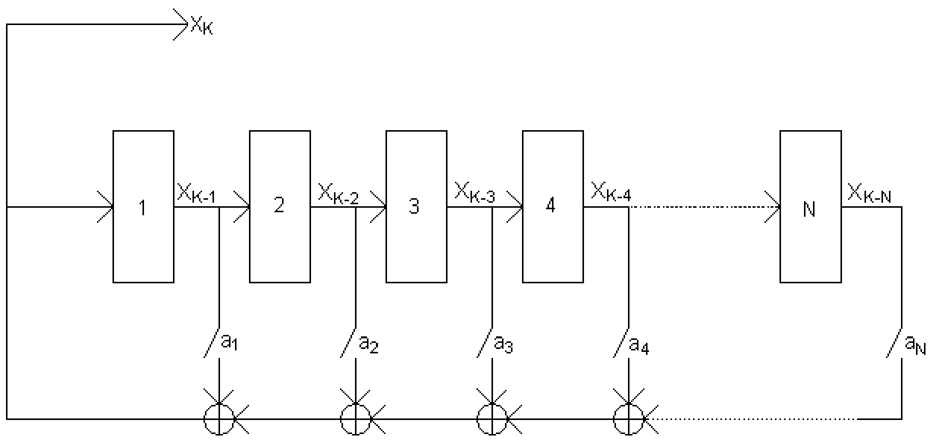


Figure 2.1: General LFSR. The switches a_1 to a_N make out the tap sets, and can be either closed (1) or open (0). The coefficients X_K to X_{K-N} represent the bits being shifted through the LFSR. The output of the LFSR is shown as X_K but could be taken from any of the N register outputs.

LFSR shown in figure 2.1 can be described as shown in equation 2.1.

$$X_K = a_1 X_{K-1} \oplus a_2 X_{K-2} \oplus ...... \oplus a_N X_{K-N} \tag{2.1}$$

The coefficients a_1 to a_N can assume the values '1' (switch closed) or '0' (switch open) and make out the tap sets. X_K to X_{K-N} are the bits being shifted through the registers. X_K is shown as the LFSR output on figure 2.1, but the output from the LFSR can be taken from any of the N register outputs.

It can be shown [4] that for a LFSR of length N, one or several tab sets exists that will result in the generation of a maximal length sequence. Maximal length sequences have a period length of $2^N - 1$ bits, and have many useful properties. A table containing tap sets that result in the generation of maximal length sequences, can be seen in [5]. In the remainder of this chapter focus will be on maximal length sequences. As indicated in figure 2.1, the bits in the shift register are shifted from the left to the right. Each time the bits in the register are shifted, a new state will appear in the register. The number of states that the shift register can assume, is equal to the length of the sequence. In a LFSR like the one in figure 2.1, configured to generate a PRBS, all possible states of the register will appear exactly once (except the all zeroes state). The all zeroes state consisting of only zeroes is not included, as this state would cause the LFSR to output zeroes indefinitely.

2.2 Properties of Pseudo Random Binary Sequences

A PRBS generated as shown in figure 2.1, is a periodic sequence of ‘0’s and ‘1’s, where the length of each period can range from a few bits to several billion bits. Different sequences with equal lengths can be generated using different tap sets, and this can be used for testing the pattern dependant hardware often residing in digital hardware (for instance clock recovery circuits). Independent of the sequence length, the number of ‘0’s and ‘1’s in a sequence, will differ by only one, and the majority of bits will always be ‘1’s. Thus in a sequence that is 7 bits long, there will be four ‘1’s and three ‘0’s. This distribution of ones and zeroes is a consequence of all N bit states in the LFSR except one, appear exactly once [4]. If all N bit states, including the all zero state, had appeared exactly once, the distribution of ones and zeroes would have been equal.

A sequence of consecutive ‘1’s or ‘0’s is called a run, and the number of ‘1’s or ‘0’s in the run is called the run length [5]. In a PRBS of length $2^N - 1$ bits there will be one run of N ‘1’s, and one run of $N-1$ ‘0’s. The number of runs of various lengths in a $2^N - 1$ bits long PRBS is given in table 2.1 [5]. As table 2.1 shows, about half of the runs will be of length 1,

Run length	‘1’s	‘0’s
N	1	0
$N-1$	0	1
$N-2$	1	1
$N-3$	2	2
$N-4$	4	4
..	.	.
..	.	.
2	2^{N-4}	2^{N-4}
1	2^{N-3}	2^{N-3}

Table 2.1: Number of runs of various lengths for a $2^N - 1$ bit long PRBS [5]

one quarter will be of length 2, one eighth will be of length 3 etc.

An example of a LFSR generating a PRBS of length $2^4 - 1$ can be seen in figure 2.2. The sequence generated by the LFSR in figure 2.2 will be `111100010011010`. The state diagram for the LFSR in figure 2.2 consists of 15 different states of 4 bits. Each new state is obtained by shifting the previous state one bit to the right, and then replacing the left most bit by the result of a module 2 addition corresponding to the tap set used. Figure 2.3 shows this state diagram. The LFSR in figure 2.2 can be described by the polynomial [4]

$$1 + X^3 + X^4, \tag{2.2}$$

where X^3 and X^4 refers to tap set used. For LFSRs generating maximal

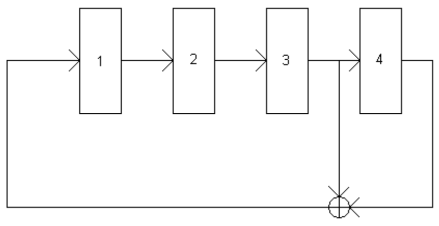


Figure 2.2: LFSR generating a PRBS of length $2^4 - 1$.

length sequences, this polynomial will be primitive[1] [4]. Each state in the LFSR state diagram can be described by polynomials in a similar way, but these polynomials will not all be primitive. The polynomial describing the LFSR is often referred to as the parity check polynomial, as multiplication between this polynomial and any other polynomial that is part of the LSFR state diagram will yield the result of zero. Multiplication, addition, and division of such polynomials are done by counting coefficients module 2, and counting powers of X modulo N [4]. For instance, multiplying the parity check polynomial corresponding to the LFSR shown in figure 2.2, with one of the states (`1001`) from the LFSR state diagram (figure 2.3) yields

$$(1+X^3+X^4)(1+X^4) = 1+X^3+X^4+X^4+X^7+X^8 = 1+2X^3+3X^4 = 1+1 = 0 \tag{2.3}$$

as $2 = 0$, $X^8 = X^4$ and $X^7 = X^3$ when powers of X are counted module 4 and coefficients are counting modulo 2. The polynomial description of the LFSR can be replaced by a description based on vectors if found appropriate. The LFSR in figure 2.2 can be described by the vector `0011`, and the corresponding states of the LFSR state diagram can be described in the same way.

2.2.1 Shift and add

If the sequence generated by a LFSR is shifted in time and added (modulo 2) to a none shifted version of the same sequence, the resulting sequence will be an identical sequence shifted by a certain number of bits. If a PRBS of length $2^4 - 1$ bits is shifted by 4 bits, and added to a non-shifted version of the same PRBS, then the resulting sequence will be the same PRBS shifted by 3 bits [5]. This property can be used to generate several sequences with known large delays between them.

[1]A primitive polynomial is a polynomial that does not contain any factors of lower degree, i.e. the polynomial cannot be expressed as a product of polynomials of lesser degree.

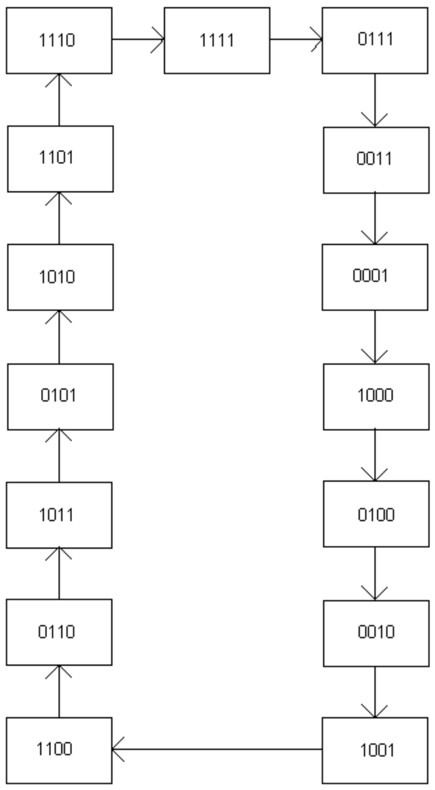


Figure 2.3: State diagram for a LFSR generating a PRBS of length $2^4 - 1$

2.2.2 Subsequences

An interesting property of PRBSs is that the alternate bits in the sequence form the same sequence at half the rate. Figure 2.4 illustrates this. As figure 2.4 shows, alternate bits from the original sequence (middle) form two identical sequences (top and bottom). The phase shift between the two sequences resulting from the decimation is 7.5 clock cycles at $\frac{f}{2}$. The principle can be extended further to achieve a higher order of decimation. A sequence can be decimated by a factor R, where R is a power of 2. This will result in R identical sequences each at a rate of $\frac{f}{R}$. The subsequence property works the other way around as well. If it is desired to generate a sequence at rate f, it can be done by generating two identical sequences at rate $\frac{f}{2}$ delayed by 7.5 clock cycles (at $\frac{f}{2}$), and then multiplex them. This can be very useful if the PRBS is generated by low speed logic, and has to be transmitted at high speed.

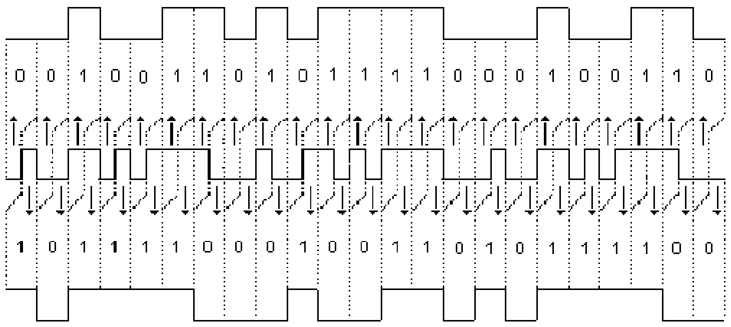


Figure 2.4: Decimation of one PRBS (middle) at frequency *f* into two identical PRBSs (top and bottom) of frequency *f/2* [5]

2.2.3 Correlation

Correlation provides a way to calculate the degree of similarity between two sequences. The correlation between two identical PRBSs is interesting, as it assumes one of two possible values describing if the two sequences are in phase or not. The correlation is obtained by comparing the two sequences bit by bit (module 2), and can be expressed mathematically as [5]

$$R(m) = \frac{1}{L} \sum_{K=0}^{L-1} x(K) y(K+m) \tag{2.4}$$

where x and y are two identical PRBSs, m is a time delay, and L is sequence length. The correlation can be performed by using XNOR gates, which will result in a logic '1' when a match is encountered, and a logic '0' when a mismatch is encountered. The output bits from the XNOR gates are summed up, and divided by the total number of bits compared to yield the correlation. If the the two sequences are in phase, the result of the correlation will be 1 ('autocorrelation' [5]), and if the two sequences are out of phase, the result of the correlation will be 0.5 ('crosscorrelation' [5]). The correlation property of PRBSs can be used for synchronizing a PRBS receiver.

2.2.4 Acquisition of Pseudo Random Binary Sequences

The PRBS receiver is a LFSR identical to the LFSR used by the transmitter. In order for the receiver LFSR to function correctly, it has to be synchronized. Synchronization can be done in several ways. A simple way is to search the incoming bit sequence for an unique bit pattern (synchronization pattern), and when the pattern appears the LFSR is initialized to this value. The synchronization pattern naturally has to be some number of successive bits from a PRBS identical to the PRBS being received. As the composition of the incoming PRBS is known beforehand, the synchronization pattern predetermined. This is one of the advantages of using PRBSs for transmission tests. Once the receiver LFSR is synchronized, the incoming bit stream will be shifted through the LFSR bit by bit. The receiver functions by comparing the result of the modulo 2 addition of the feed back taps to the incoming bits. Recalling how the sequence is generated, it can be seen that the result of the modulo-2 addition of the feed back taps, is identical to the bit which is about to enter the first register in the LFSR. Thus by comparing the result of the module 2 addition of the tab sets, with the incoming bit, errors can be registered. The bits can be compared using a XOR gate, which results in a logic "HIGH" when an error is received, and a logic "LOW" when a correct bit is received. The output of this XOR gate can be used as input to an error counter. The basic receiver circuit can be seen in figure 2.5. For the sake of simplicity the synchronization circuit is omitted in figure 2.5. The circuit in figure 2.5 posses a problem, and

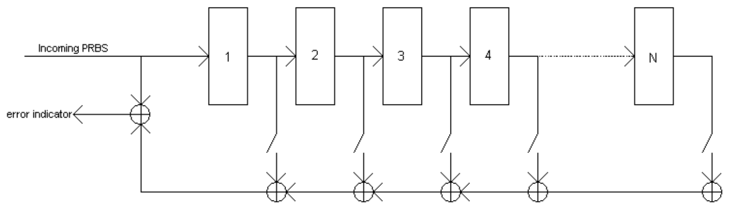


Figure 2.5: Basic receiver LFSR after synchronization

that is multiple registration of single bit errors. When a bit error enters the LFSR, it will toggle the error indicator each time it passes a feed back path. Moreover, if several bit errors separated by the same amount of bits as the feed back tabs, enters the LFSR, it can lead to undetected errors (as one error at each feed back tab can cancel each other). In order to accommodate this problem, the bit error is corrected upon registration, so that it never enters the LFSR. This can be done by insertion of an additional XOR gate as shown in figure 2.6. Again, for the sake of simplicity the synchronization circuit is not included in figure 2.6. The synchronization circuit is simply a N bit comparator, that compares N bit of the incoming PRBS at a time with the synchronization pattern. Once a match is found, the synchronization pattern is loaded into the LFSR, and the error registration can begin.

2.2.5 Power Spectrum of Pseudo Random Binary Sequences

The power spectrum of a PRBS of length $2^N - 1$ has a $(\frac{sin(x)}{x})^2$ envelope as shown in figure 2.7 [4, 5] (the scale in figure 2.7 is not exact). The

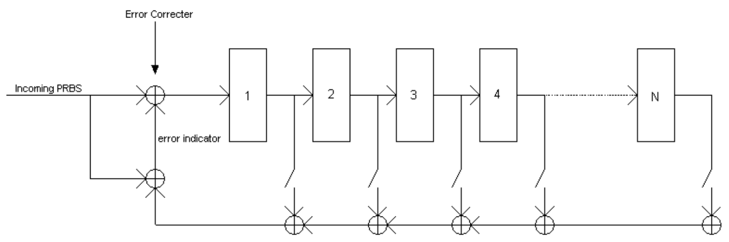


Figure 2.6: Error compensating receiver LFSR

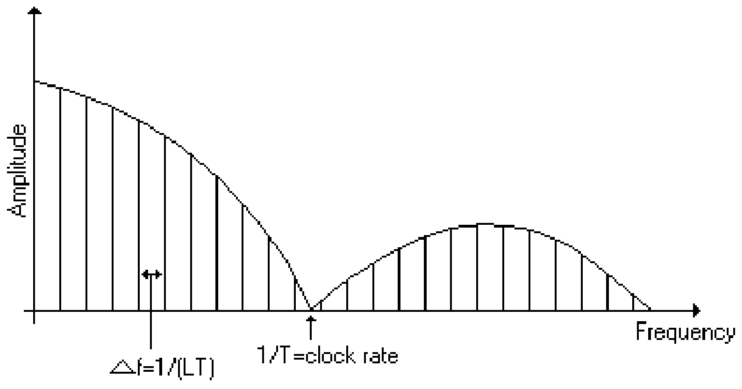


Figure 2.7: Power spectrum for a PRBS [5].

spectrum nulls occur at $f = n/T$, where T is the bit duration and n is an integer. The spacing between the line frequencies is $\frac{1}{(2^N-1)T}$, which means that in order to reduce frequency spacing, the length of the PRBS should be increased. The difference between the spectrum of a true random signal and that of a maximal length PRBS, is that the spectrum of the true random signal is continuous, while that of a PRBS is discrete [4]. But by choosing a PRBS with a long period, close resemblance to a true random signal can be obtained. This property makes PRBSs ideal as test signals.

2.3 Summary

A pseudo random binary sequence is a random bit sequence that repeats it self. The properties that PRBSs hold, together with the simple generation and acquisition scheme, makes them ideal for test purposes. If the sequence length of a PRBS is chosen long enough, the power spectrum of the sequence will show very close resemblance to that of a truly random sequence.

Chapter 3

Test module description and requirements

The first task in a design process is to specify the requirements for the design. In order to specify the requirements for the test module, a functional block diagram illustrating the test module must be constructed. Constructing a functional block diagram for any design requires knowledge about the overall demands for that design. The are two overall demands for the test module, namely that the test module has to obey the 40 Gb/s interface specification (will be described shortly), and that the PRBS has to be generated and received using FPGAs. First the 40 Gb/s interface will be described in detail, and then a functional block diagram illustrating the test module will be constructed, and requirements for each block in the block diagram will be specified.

3.1 40 Gb/s Interface Specification

The 40 Gb/s interface separates the test module and the MUX/DeMUX module, and is decided by the MUX/DeMUX module. This section describes the 40 Gb/s interface in terms of signal levels, timing relations, and physical implementation.

3.1.1 Definitions

Upstream : Signals transmitted from the test module to the MUX/DeMUX module.
Downstream : Signals transmitted from the MUX/DeMUX module to the test module.

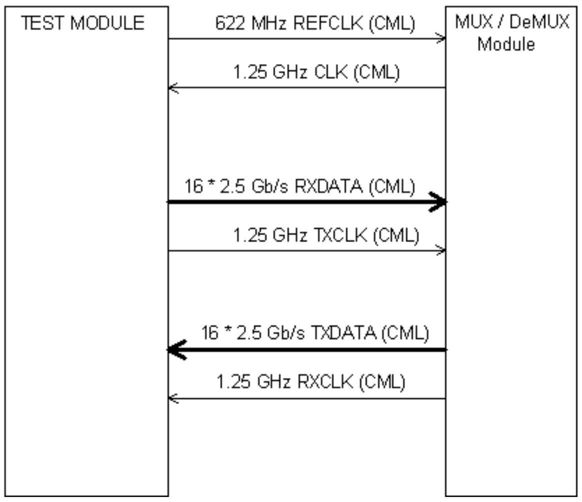


Figure 3.1: The illustration shows the signals that exist between the Test module and the MUX/DeMUX board (The 40 Gb/s Interface).

3.1.2 Description

The 40 Gb/s interface consists of 32 data signals and 4 clock signals. Sixteen data signals together with one clock signal, to which the data signals are locked, are transmitted upstream. Sixteen data signals together with one clock signal, to which the data signals are locked, are transmitted downstream. Furthermore a 622 MHz reference clock signal is transmitted upstream, and one 1.25 GHz clock signal is transmitted downstream. It is intended that the downstream 1.25 GHz clock signal should be locked to the upstream 622 MHz clock signal. However, on the initial prototype of the MUX/DeMUX module, the downstream 1.25 GHz clock signal will not be locked to the upstream 622 MHz clock signal. Thus the 622 MHz upstream clock signal cannot be used for processing data on the test module. All generation and transmission of data on the test module, have to be referenced to the downstream 1.25 GHz clock signal. This will ensure that the test module and the MUX/DeMUX module will be processing data at the same speed. The data rate of the upstream and downstream data signals is 2.5 Gb/s, and the frequency of their respective clock signals is 1.25 GHz. Figure 3.1 illustrates the 40 Gb/s interface. The data bits are transmitted LSB first, both upstream and downstream. The signal names specified in figure 3.1 will used in the remainder of this chapter.

3.1.3 Signal levels

The *TXDATA* and *RXDATA* signals between the test module and the MUX/DeMUX module have to be differential CML (Current Mode Logic).

The upstream *REFCLK* clock signal has to be differential and correspond to the LVPECL signal levels. The *TXCLK* and *RXCLK* clock signals, have to be differential CML. The *TXDATA* signals and the *TXCLK* signal must be referenced to 1.8V (maximum), or be AC coupled, and their signal swing must be between 100 mV_{pp} to 450 mV_{pp} single ended. The *TXDATA* signals are received by a FIFO on the MUX/DeMUX module, which uses the *TXCLK* signal to receive the data. The FIFO has a maximum allowable input current of 16 mA, which means that the test module output current, on the *TXDATA* and *TXCLK* signals, must be less than or equal to 16 mA. The output current is here defined as the constant current that runs in the output stage of the CML driver. CML output stages are described in detail in section 4.1.2. The common mode voltage on the *REFCLK* clock signal must be between -1 V and 0 V, and have a signal swing within LVPECL specifications. The common mode voltage on the *RXDATA* signals is 1.8 V, and the signal swing is within 100 mV_{pp} - 450 mV_{pp}. The output current from the output stage of the DeMUX on the MUX/DeMUX module is within 13 mA - 15 mA.

3.1.4 Jitter

The jitter in the frequency range 50 KHz to 80 MHz on the upstream *REFCLK* signal should not exceed 5 ps. The jitter on the 1.25 GHz CML upstream *TXCLK* signal affects the size of the sampling window of the 2.5 Gb/s upstream *TXDATA* signals. Its maximum permissable amplitude should be kept so valid sampling of the *TXDATA* signals can take place. The size of the sampling window cannot be determined before all components have be selected.

3.1.5 Timing

The *TXDATA* signals interfaces a FIFO which has a setup and hold time requirement of 75 ps each. The duty cycle of the *TXCLK* and *TXDATA* signals must be 0.5 ± 0.05 UI (Unit Interval = 400 ps for 2.5 Gb/s). The *TXCLK* signal has to be delayed with respect to *TXDATA*, so that the data signals will be sampled in the middle of the “data eye” . Thus the *TXCLK* signal has to be delayed 200 ps with respect to *TXDATA*. The skew on the *RXDATA* signals does not exceed ± 40 ps, and the duty cycle of the *RXDATA* signals is 0.5 ± 0.05 UI.

3.1.6 Physical backplane

The test module has to interface the MUX/DeMUX module through an internal Tellabs backplane called S4[1]. The backplane is a large switching

[1]The official designation for this backplane is: Tellabs 6350 Switch Node

matrix designed to be used with up to 8 sets 10 Gb/s modules, and 2 sets 40 Gb/s modules [6]. Each set consists of an electrical and an optical PCB. The physical connections chosen, and the positions selected for the test module, the MUX/DeMUX module, and the opto module is decided in [6]. An illustration of the S4 backplane can be seen in appendix C.

3.2 Functional Block Diagram

The 40 Gb/s interface impose constraints on the signals connecting the test module to its surroundings (the MUX/DeMUX module). The demand that the PRBS has to be generated in a FPGA imposes constraints on the internal signals on the test module. Regular FPGAs are not able to receive or transmit data at a rate higher than about 800 Mb/s using LVDS I/Os. Hence, transmitting the PRBS at a rate of 40 Gb/s, will require 64×622 Mb/s LVDS signals. In order to fulfill the 40 Gb/s interface specification, the 64 622 Mb/s LVDS signals have to be converted into 16 2.5 Gb/s differential CML signals. This means that a multiplex stage (MUX stage) will be needed. The data generated by the test module, has to be generated at the same rate as the MUX/DeMUX module processes it. Hence, the clock signal used to generate the PRBS in the FPGA, has to be derived from the 1.25 GHz clock signal generated by the MUX/DeMUX module. The MUX stage will need an input clock in order to multiplex and transmit the data signals transmitted from the FPGA. This clock signal has to be locked to the clock signal that the FPGA uses to generate data with, or else the FPGA will generate data at a different rate than the MUX stage transmits it.

The receiving part of the test module is governed by the same considerations. The MUX/DeMUX module transmits 16 2.5 Gb/s differential CML signals to the test module, and in order for these signals to interface the receiver FPGA, they have to be demultiplexed into 64 622 Mb/s LVDS signals. Thus a demultiplex stage (DeMUX stage) is needed. The DeMUX stage has to facilitate CDR circuitry, or use the downstream 1.25 GHz clock signal to receive the 16 2.5 Gb/s data signals. The DeMUX stage has to deliver one or several clock signal(s) to which the 64 622 Mb/s data signals are locked. This/These clock signal(s) will be used by the receiver FPGA to receive the 64 622 Mb/s data signals. Based on these considerations, a block diagram illustrating the required functionality for the test module, has been constructed. This block diagram is shown in figure 3.2. In the following sections a more detailed description of the functionality and requirements for each block in the block diagram, will be given.

3.2.1 Oscillator

The oscillator block (OSC) represents the oscillator that generates the 622 MHz reference clock for the MUX/DeMUX module. The oscillator output

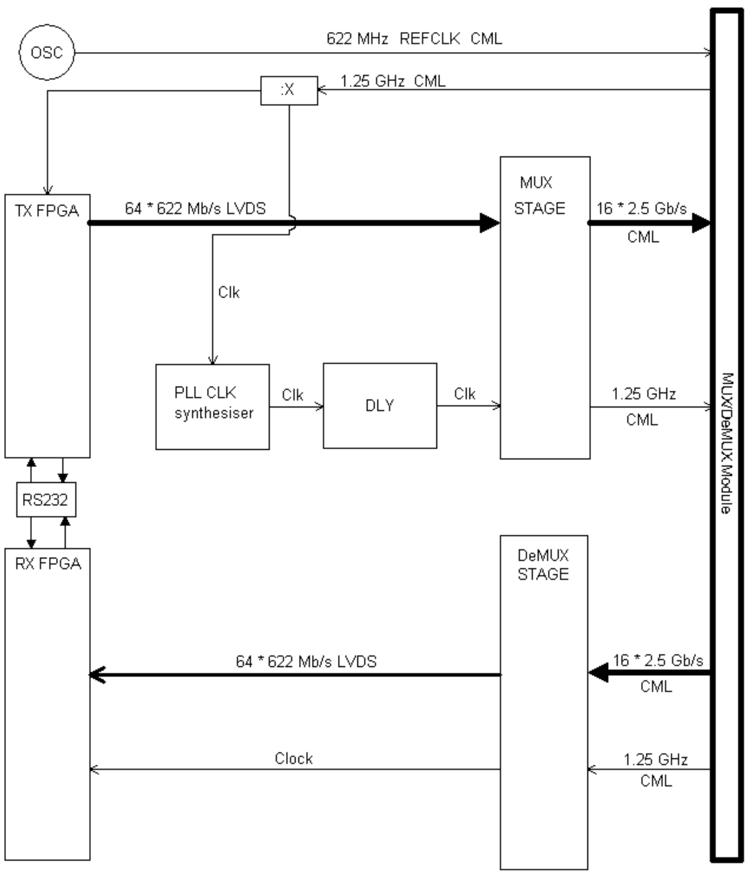


Figure 3.2: Block diagram illustrating the requirements for the test module

has to correspond to the LVPECL signal levels. According to the 40 Gb/s interface specification, the jitter in the frequency range 50 KHz to 80 MHz on the this clock signal, should not exceed 5 ps.

3.2.2 Clock divider

The clock divider block (:X) represents one or several clock dividers for dividing the 1.25 GHz CML clock signal that the MUX/DeMUX module generates and transmits to the test module. The 1.25 GHz clock has to be divided because it is to be input to the transmitter FPGA and to the phase locked loop (PLL) clock synthesizer. With the current technology, it is not possible to input signals with a frequency higher than about 400 MHz into regular FPGAs.

The clock divider stage has to accept CML signals as input, or it has to be possible to convert the input CML signal into the appropriate signal standard. The clock divider stage has to output two clock signals, and as the FPGA accepts LVDS signals, and the PLL clock synthesizer most likely accepts LVPECL or LVDS signals, the output of the clock divider stage has to be able to interface to both LVPECL and LVDS.

3.2.3 Transmitter FPGA

One of the main demands for the test module is that the transmitter (TX FPGA) has to be implemented in an FPGA. The PRBS transmitter has to facilitate a PRBS generator as described in section 2.1, and the PRBS generator has to be able to provide the PRBS at a rate of 40 Gb/s on its parallel outputs. On the first prototype of the test module the PRBS transmitter only has to transmit a PRBS with a length of $2^{31} - 1$ bits, but on the final edition of the test module it has to be possible to change the length of the PRBS from a PC using an RS-232 interface. The universal asynchronous receiver transmitter (UART) for this RS-232 interface has to be implemented in the FPGA, thus the only additional hardware needed to implement the RS-232 interface will be a level converter and a connector. The level converter and the connector will be included on the test module prototype, but on the first edition of the prototype the UART will not be implemented.

The output data signals will be LVDS signals, as LVDS is the most commonly used signal standard for high speed I/O's in the FPGAs currently available. Current technology allow FPGAs to output data at a rate of up to approximately 800 Mb/s using double data rate (DDR). In order for the FPGA to supply the PRBS at a rate of 40 Gb/s on its outputs, 64 622 Mb/s LVDS data outputs will be needed. These 64 data signals interface the MUX stage, and as the output signals of the MUX stage have to be synchronous, the data signals transmitted from the FPGA have to be synchronous too.

This imposes constraints on the timing of the high speed I/Os from the FPGA.

The bit period of the 622 Mb/s data signals is 1.6 ns, and according to the IEEE 1596.3 standard on LVDS signals [7], the transition time for LVDS signals must not exceed 500ps. This means that there is a valid data window of 1100 ps left. The skew between the 64 data signals from the FPGA, and the jitter level on the clock signal supplied to the multiplexers, will reduce the size of this sampling window. The FPGA and the multiplexers have to be chosen, so that the 64 data signals on the FPGA output, can be sampled simultaneously by the MUX stage.

3.2.4 PLL CLK Synthesizer

The clock frequency needed for the MUX stage will most likely differ from the clock frequency that the FPGA uses to generate data with. This could be facilitated by a clock divider, however using a PLL makes it possible to attenuate any jitter that should be present on the clock signal for the MUX stage. This is important as the jitter on this clock signal can influence the performance of the MUX stage.

The PLL CLK Synthesizer has to provide a clock signal with the frequency needed for the MUX stage, and this clock signal has to be locked to the clock signal that the FPGA uses to generate data with. This is a consequence of the TX FPGA and the MUX stage having to process data at the same speed.

The input clock for the PLL CLK synthesizer is generated by the :X stage, thus the PLL clock synthesizer has to accept LVDS or LVPECL signals as input. The output clock interfaces the MUX stage, and has to correspond to the LVDS or LVPECL signal levels.

Designing phase locked loops is a difficult task, and is beyond the scope of this thesis. Thus it is desired to use an integrated solution to implement the PLL CLK synthesizer, where loop filter values are specified in terms of guarantied performance.

3.2.5 Delay line

The delay line (DLY) block represents an adjustable delay line, and this delay line is inserted to make it possible to align the clock input to the MUX stage correctly with respect to the data input to the MUX stage. The necessity of the delay line arises from the fact that the clock and data signals to the MUX stage originates from different sources. The clock signal for the MUX stage should preferably be output from the TX FPGA, as the data, which would make the timing relations between the clock and data input to the MUX stage well defined. However the clock signal for the MUX stage is not output from the TX FPGA, but is generated by the PLL clock synthesizer. The reason for this is to keep the jitter level on the clock signal as low as possible. Routing the clock signal through the FPGA would add jitter to the clock signal. The delay line should accept a LVPECL or LVDS signal as input, and output a signal corresponding to the LVDS or LVPECL signal levels. The delay line must be able to add a delay of one 622 Mb/s bit period (1.60 ns) to the clock signal.

3.2.6 Multiplex stage

The multiplex stage (MUX stage) has to accept the 64 622 Mb/s LVDS data signals transmitted from the TX FPGA, together with one or several clock signals. The LVDS data signals will be locked to the clock signals. As output, the MUX stage has to provide 16 2.5 Gb/s CML data signals, together with one 1.25 GHz clock signal, to which the data signals are locked. This is not a trivial interface, hence the MUX stage will most likely have to be made up by several multiplexers.

As mentioned in the 40 Gb/s interface specification, the FIFO on the MUX/DeMUX module, which the MUX stage interfaces, has a setup and hold time requirement of 75 ps each. This means that the minimum allowable valid data window on the MUX stage output is 150 ps, or expressed in an other way, a maximum of 250 ps is left for skew between the 16 data outputs, clock jitter on the 1.25 GHz output clock, and signal transition time of the output data signals. The skew between the data outputs from the MUX stage, is directly related to the device to device skew, or output to output skew, of the mutliplexer(s) chosen. Hence this parameter will be important to consider when selecting the components, and must be selected so that synchronous sampling of all 16 data outputs can take place. Furthermore, if the MUX stage is implemented using several multiplexers, it has to be possible to synchronize them.

3.2.7 Demultiplex Stage

The demultiplex (DeMUX) stage has to accept 16 CML data signals together with one CML clock signal. The data rate of each CML data signal is 2.5 Gb/s and the frequency of the CML clock signal is 1.25 GHz. The 2.5 Gb/s CML data signals and the 1.25 GHz CML clock signal obey the timing specifications outlined in section 3.1. The DeMUX stage has to output 64 LVDS data signals, and one or several LVDS clock signals. As these data and clock signals are transmitted to the RX FPGA, the data rate of the data signals must be 622 Mb/s, and the frequency of the LVDS clock signal(s) should preferably be 311 MHz.

The timing relation between the 622 Mb/s LVDS data signals and the 311 MHz LVDS clock signal is not important, as it is possible to delay the clock signal with respect to the data signals inside the FPGA (will be explained in detail later). All 64 LVDS data signals output from the DeMUX stage does not need to be synchronous. Instead, using the subsequence property of PRBSs, the 64 LVDS signals can be divided into smaller groups of signals, where each group of signals would represent identical PRBSs. Each group has to be represented by a clock signal, in order for the RX FPGA stage to receive the group correctly. Dividing the 64 data signals into smaller groups of signals, means that only signals belonging to the same group, have to be synchronous. This eases the PCB layout considerably, as fewer signals will need to be well controlled timing wise.

3.2.8 Receiver FPGA

One of the main demands for the test module is that the receiver has to be implemented in a FPGA (RX FPGA). The RX FPGA stage has to receive 64 622 Mb/s LVDS data signals and one or several 311 MHz LVDS clock signals. The 622 Mb/s LVDS data signals will be locked to the 311 MHz LVDS clock signal(s), but the phase relationship between the data signals and the clock signal(s) will be undefined, and the clock signal(s) will have to delayed inside the FPGA, in order for the FPGA to sample the data correctly. The skew between the data signals interfacing the RX FPGA, will not exceed 200 ps.

The RX FPGA stage will have to facilitate one or several PRBS receivers, depending on how the data is transmitted from the DeMUX stage, and each of these receivers will have to register and count errors. There are no specific demands for the PRBS receivers in terms of synchronization time etc. On the final edition of the test module the error counts from the receivers has to be added up and output to the TX FPGA stage across a RS-232 like interface. This RS-232 like interface will initially not be implemented, but it will be possible to implement this interface without having to add additional hardware to the PCB.

3.3 Summary

The 40 Gb/s interface separating the test module and the MUX/DeMUX module is well described in terms of signal levels and timing relations. The usage of FPGAs for data generation/reception results in the transmission and reception of 64 622 Mb/s LVDS signals. In order to accommodate the interface specified by the 40 Gb/s interface specification, it is necessary to include a serialization/deserialization functionality on the test module. Furthermore, clock dividers and an integrated phase locked loop functionality has been included to manage the clock distribution on the test module. The subsequence property of pseudo random binary sequences, has enabled an asynchronous receiver interface, which eases the PCB layout considerably.

Chapter 4

Test module design

The requirements and functionality of the test module has been specified, thus the components available to employ the required functions can now be investigated. After having selected components to employ all the functions illustrated in the test module block diagram, exact timing relations can be outlined, and the schematic design can commence. Schematic design requires interfacing the selected components, and the interface used between two components depends on which technology the components are based on. The most commonly used technologies in high speed components today are CML, LVDS and ECL, or LVPECL which is the positive low voltage version of ECL. The interface used between CML, LVDS and LVPECL is well described in application notes supplied by the component manufactures, and with some modifications and recalculations most interfaces can be implemented using these application notes.

At high frequencies the traces connecting the components on the PCB are to be considered as transmission lines, which means that considerations regarding correct termination of the transmission lines have to be made. The theory on transmission lines will not be covered in this report, as a thorough treatment of this subject is beyond the scope of this project.

While doing the schematic layout of a design, it is important to do some considerations regarding the test of the design. Very often additional hardware, connecters or pads need to be included in order to be able to test the design when the PCB returns with all components assembled. This chapter includes a brief description of the three technologies mentioned above, and the interface used between them. The component selection and schematic layout for each block in the test module block diagram will be described together with the timing relations for each block. Considerations regarding testing of the test module will be described.

4.1 Technologies

4.1.1 LVPECL

LVPECL originates from ECL but is a low voltage version that uses a positive 3.3 volt power supply. The output stage of a PECL circuit can be seen in Figure 4.1 [8]. The PECL output stage consists of a differential pair that

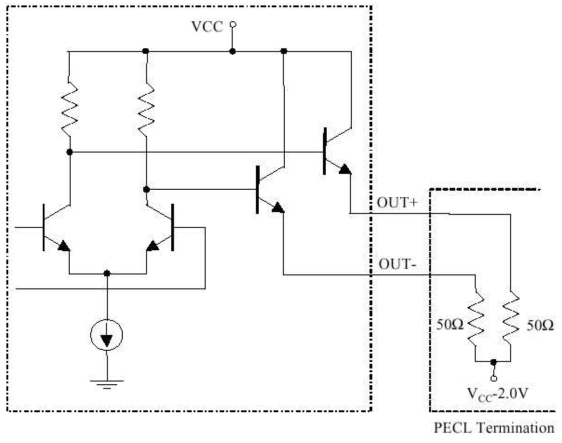


Figure 4.1: PECL output stage [8]

drives two emitter followers. The output impedance of the emitter followers is made up by the impedance of the BJT transistor when it is turned on, and is typically on the order of 5 Ω. By varying the voltage on the bases of the emitter followers, the emitter follower output shifts between output high voltage and output low voltage.

In order for the BJT transistors in the emitter followers to stay in the active operating region, the output of the emitter followers have to be biased to V_{cc} - 2 volt through a resistor. The resistor functions as a DC passage to V_{cc} - 2 volt, and the size of the resistor is chosen as a compromise between the discharge time of the transmission line and the power consumption [9]. When the the voltage on the emitter follower outputs is shifted, the transmission line connected to the PECL output is charged or discharged depending on the direction of the shift (high to low or low to high).

In most cases the resistor on the outputs of the emitter followers is chosen to be 50Ω, and it is placed at the receiver end of the transmission line. In this way it functions both as a DC passage to V_{cc} - 2 volt and as a termination resistor for the transmission line. The current that runs in the 50 Ω resistor will not be constant, as this current depends on the logic state of the LVPECL output. This resistor will be connected to the power supply, hence it is important to stabilize the power supply properly in order to avoid oscillations in the power supply voltage. Because of the symmetry of the LVPECL output stage, the total current drawn from the internal supply of the LVPECL circuit should be constant (there will always be one 'LOW' output and one 'HIGH' output, hence the same amount of current will be drawn from the supply). Thus decoupling of the internal supply of the LVPECL circuit, is not as critical as decoupling of the supply connected to the termination resistors ($V_{cc} - 2V$).

The PECL input structure is a differential pair, and the positive and negative inputs to the differential pair are both connected to the base of a BJT transistor, thus the input structure has a high input impedance. The positive and negative input to the differential PECL input structure should be biased to V_{cc} - 1.3 volt for optimal operation. The input and output specifications for LVPECL based components can vary slightly from one component to an other, and should always be looked up in the data sheet for the relevant component. Input and output specifications for Maxims LVPECL components can be seen in table 4.1.1 [8].

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Output high voltage	$T_A = 0°C$ to $+85°C$	$V_{cc} - 1.025$		$V_{cc} - 0.88$	V
	$T_A = -40°C$	$V_{cc} - 1.085$		$V_{cc} - 0.88$	V
Output low voltage	$T_A = 0°C$ to $+85°C$	$V_{cc} - 1.81$		$V_{cc} - 1.62$	V
	$T_A = -40°C$	$V_{cc} - 1.83$		$V_{cc} - 1.55$	V
Input high voltage		$V_{cc} - 1.16$		$V_{cc} - 0.88$	V
Input low voltage		$V_{cc} - 1.81$		$V_{cc} - 1.48$	V

Table 4.1: LVPECL input and output specifications. (from [8])

4.1.2 CML

The CML input and output structure is simple, and has the advantage of not needing any external termination resistors. The termination resistors are an integrated part of the input and output structure. The output structure can be seen in figure 4.2. As figure 4.2 illustrates, the CML output stage consists of a differential pair, and functions by shifting the current between the two halves of the differential pair. Normally the constant current source in the CML output stage is about 16 mA as shown in figure 4.2. When one of the differential outputs (OUT+ or OUT- in figure 4.2) is in "low" state, 8 mA will be drawn from the power supply V_{cc}, and 8 mA will be drawn from the power supply of the CML input stage to which the differential CML output is connected. This is shown in figure 4.3. The current is drawn equally from the two supplies because the input impedance of the CML input stage is 50 Ω, and thus equal to the collector resistor in the output stage. Assuming that the current source in the CML output stage is 16 mA, the single ended output "high" voltage is V_{cc}, and the single ended output "low" voltage is V_{cc} - 0.4 volt. This yields a single ended voltage swing of 400 mV, and a differential voltage swing of 800 mV.

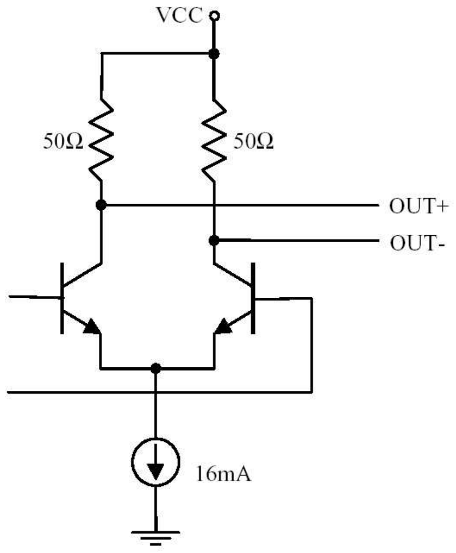


Figure 4.2: CML output stage [8]

The CML input structure consists of two emitter followers driving a differential pair. A 50 Ω resistor is connected from the base of each of the emitter follower BJT transistors to the power supply. This resistor makes up the input impedance of the CML input structure. Figure 4.4 illustrates the CML input structure. The input and output specifications for CML based components can vary from one component to another, but generally the differential voltage on both the input and output of a CML component should be about 800 mV, and the input common mode voltage should be V_{cc} - 0.2 volt (DC coupled).

4.1.3 LVDS

LVDS stands for low voltage differential signalling, and has the advantage of consuming very little power. The very low power consumption makes LVDS ideal for applications that require a high I/O count. Figure 4.5 shows the LVDS output structure. The LVDS output structure has a differential output impedance of 100 Ω (typically) [8]. In order for the differential output to be terminated correctly, a 100 Ω resistor has to be connected between OUT+ and OUT- at the receiver end. A maximum of 4 mA will run through the 100 Ω termination resistor, and the direction of this current will change every time the output shifts from “high” to “low” or the opposite. The direction of the output current is controlled by letting one side of the differential output stage either sink or source current, while the other side of the differential output stage does the opposite (push pull).

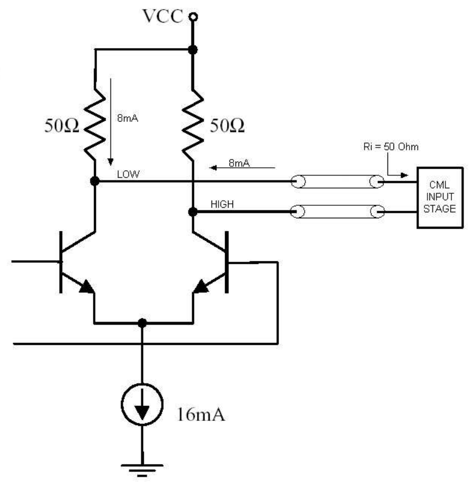


Figure 4.3: CML output current. When one of the differential outputs are in low state, 8 mA will be drawn from the power supply of the output stage, and 8 mA will be drawn from the power supply in the input stage to which the output stage is connected.

There are two output specifications for LVDS, one full range output specification, and one reduced range output specification. The full range output specification specifies a single ended output voltage swing in the interval 250 mV to 400 mV, and the reduced output range specification specifies a single ended output voltage swing in the interval 150 mV to 250 mV. The output common mode voltage will be about 1.2 volt. LVDS based circuits can use either 3.3 V or 2.5 V power supplies. The exact output specifications can be seen in the IEEE 1596.3-1996 standard [7].

The LVDS input structure has the advantage of a wide input voltage range (0 volt - 2.4 volt), and a low differential input voltage threshold of 100 mV. The wide input voltage range combined with the low differential input voltage threshold, allows for a $\pm$ 1 volt difference in ground potential between the LVDS driver and the LVDS receiver. Details on the LVDS receiver circuitry will not be given here, but basically the LVDS receiver consists of an adaptive level shifter and a schmitt trigger [8]. Following the schmitt trigger is a differential amplifier. The adaptive level shifter sets the common mode voltage to a constant value on the input of the schmitt trigger. The schmitt trigger provides hysteresis, so that unconnected inputs is prohibited from causing oscillations on the receiver output. The exact LVDS input specifications can be seen in [7].

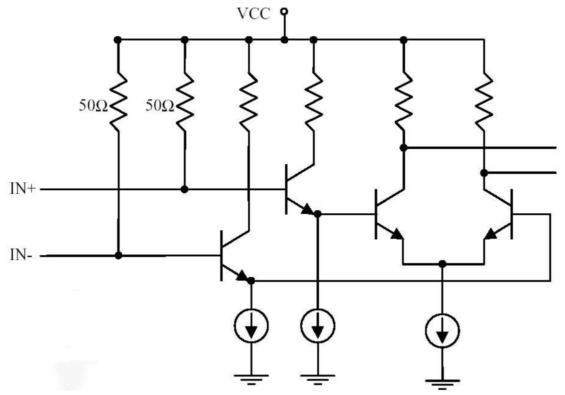


Figure 4.4: CML input stage [8]

4.2 Interfacing between LVDS, CML and LVPECL

As mentioned in the beginning of this chapter, the interfaces used between different component technologies are well described in application notes supplied by the component manufactures. For completeness, this section includes a brief description of the most frequently used interfaces in this design.

The interface used between components based on different technologies usually consists of a resistor termination network. The resistor termination network has to convert the common mode voltage and signal swing of the transmitting component, into voltage levels acceptable by the receiving component. The resistor termination network is usually placed at the receiver end, and thus has to function as a termination of the transmission line as well. In some cases the output signal from the transmitting component, has to be AC coupled and re-biased at the receiving end. This is done in order to create the right common mode voltage demanded by the receiving component. When PECL based components are used, the resistor termination network has to function as a 50 Ω passage to V_{cc} - 2 V as well.

4.2.1 CML to LVPECL

Interfacing from CML to LVPECL requires AC coupling, as the output common mode voltage of the CML driver is about 1 volt higher than the LVPECL input common mode voltage. Figure 4.6 [8] shows an interface that can be used to interface from CML to LVPECL. The transmission line connecting the CML driver to the LVPECL receiver is assumed to have a characteristic impedance of 50 Ω (100 Ω differential). The 100 Ω resistor placed close to the AC coupling capacitors terminates the transmission line so that no reflections will occur. On the receiver side of the AC coupling

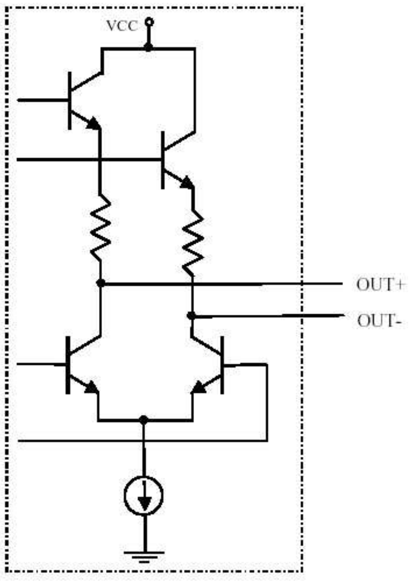


Figure 4.5: LVDS output stage [8]

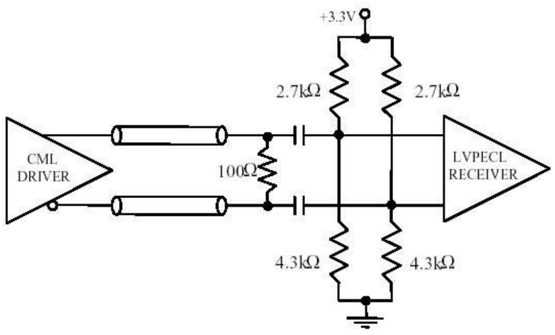


Figure 4.6: CML to LVPECL interface [8]

capacitors, a resistor network creates the common mode voltage (2 volt) needed by the LVPECL receiver. The signal swing at the receiver end will be 400 mV single ended (800 mV differential), and this corresponds nicely to the signal swing needed by the LVPECL receiver.

The interface shown in figure 4.6 will cause the common mode voltage on the output of the CML driver to shift 0.2 volt down, compared to a normal DC coupled interface between two CML components. In section 4.1.2 it was explained how the current running in the differential pair, residing in the output structure of the CML driver, is drawn equally from the power supply of the CML driver and the power supply of the CML receiver. The current was split equally between the CML driver and CML receiver, because the input impedance of the CML receiver equals the collector impedance in the CML driver. This match in impedance does not exist in the interface shown in figure 4.6. The input impedance of the CML receiver is here replaced by a 150 Ω resistor (the 100 Ω termination resistor in series with the 50 Ω collector resistor residing in the other half of the differential pair in the CML driver output structure). This mismatch in impedance will cause 2/3 of the 16 mA to run in one side of the differential pair in the CML driver (the half in which the BJT transistor is active), and 1/3 of the 16 mA to run in the opposite side of the differential pair in the CML driver (refer to figure 4.2 for details on the CML output structure). The shift in common mode voltage will not affect the LVPECL receiver as the signal is AC coupled.

4.2.2 LVPECL to LVDS

In order to interface from LVPECL to LVDS, there are several things that need to be considered. The LVPECL driver outputs have to be terminated to V_{cc} - 2 V through a 50 Ω resistor, and the transmission line connecting the LVPECL driver to the LVDS receiver, has to be terminated correctly (100 Ω differentially). The differential LVPECL output voltage can vary from about 1 V to about 1.9 V [8], and the maximum single ended LVPECL output voltage is 2.42 V [8]. Thus the LVPECL signal has to be attenuated, so that the maximum single ended input voltage to the LVDS receiver, does not exceed the LVDS input voltage range specification. At the same time the attenuation should be small enough to keep the differential LVDS input voltage above 100 mV, which is the lower limit of the LVDS input sensitivity. Figure 4.7 [10] shows the resistor termination network used to interface from LVPECL to LVDS. In order to satisfy the first demand described above, the DC voltage in point A is set to $V_{cc} - 2V$, and the impedance seen from point A to ground is set to 50 Ω (Thevenins theorem). This yields equation 4.1 and equation 4.2 respectively.

$$V_A = \frac{(R_2 + R_3)}{(R_1 + R_2 + R_3)} V_{cc} = 1.3V \tag{4.1}$$

$$R_1 || (R_2 + R_3) = 50\Omega \tag{4.2}$$

The DC voltage in point B has to be within the LVDS input voltage range, and preferably as close to 1.2 V as possible. However, as long as the input voltage in point B stays within the LVDS input voltage range of 0 V - 2.4 V, the signal will be received correctly. The DC voltage in point B can be described by equation 4.3.

$$V_B = \frac{R_3}{(R_1 + R_2 + R_3)} V_{cc} \approx 1.2V \tag{4.3}$$

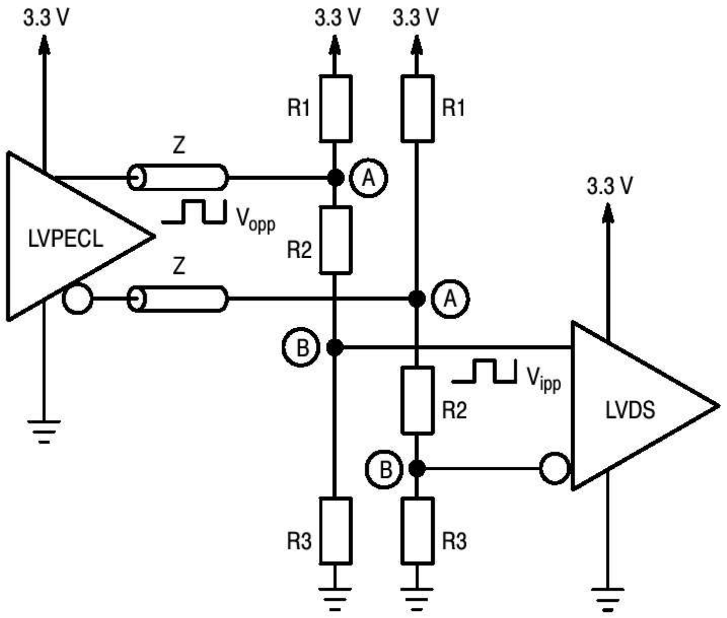


Figure 4.7: LVPECL to LVDS interface [10]

In order to keep the differential LVDS input voltage in point B above 100 mV but at the same time avoid a single ended voltage level higher then 2 V at the LVDS input, the gain from point A to point B should be within the interval 0.2 to 0.8. The gain from point A to point B can be described by equation 4.4.

$$Gain = \frac{R_3}{R_2 + R_3} \in [0.2; 0.8] \tag{4.4}$$

Using equation 4.1 to 4.4 appropriate resistor values can be determined. The resistor values used in this design are $R_1 = 121\Omega$, $R_2 = 56.2\Omega$ and $R_3 = 21.5\Omega$. These values yield a DC voltage in point A of 1.29 V, a DC voltage in point B of 357 mV, a gain from point A to point B of 0.277, and a impedance seen from point A to $V_{cc} - 2V$ of 47.3 Ω. The differential input voltage swing in point B will be in the interval 277 mV to about 500 mV which is within the LVDS input voltage specification.

4.3 Component selection

Several considerations need to be done when selecting components. The primary concern is to chose components that correspond to the specified requirements, but there are practical issues that sometimes prevent the selection of an otherwise acceptable component. In most companies a component database exists, that specifies which components are preferred choices. A component is a preferred choice because its footprint used in the PCB layout exists, and because experience in mounting the component on the PCB has already been made. Furthermore a preferred component is usually in the company stock, and thus delivery time is not an issue. Whenever a new footprint has to be made, or a component has to be mounted for the first time, errors might occur. Errors can mean that the PCB will have to be redone, or other components have to be selected, and this can greatly delay the fabrication of any module.

Based on the requirements specified for each block in the test module block diagram outlined in chapter 3, and on the practical considerations just explained, components for each block in the block diagram have been selected. The construction of the test module block diagram outlined in chapter 3, was based on the overall demands for the test module. Having selected all components allows for a more detailed version of the test module block diagram. As a consequence of the selected components, some blocks have been added to the test module block diagram. The detailed version of the test module block diagram can be seen in figure 4.8. In the following sections the components selected for each block in figure 4.8 will be described, and data sheets will be commented where necessary.

4.3.1 Oscillator

The oscillator (OSC) selected to generate the 622 MHz reference clock is a VCSO (Voltage Controlled Saw Oscillator) called VS-500, supplied by the company Vectron. The oscillator has a very low output jitter (less than 1ps rms in the frequency range 50 kHz - 80 MHz at 622 MHz) which is below the maximum allowable jitter specified in section 3.1.4. The VS-500 VCSO is already used within Tellabs, and thus makes it a natural choice. The data sheet for the VS-500 VCSO can be seen in appendix B.1.

4.3.2 Clock divider

Several “divide by two” clock dividers (:2) are used in the design. One is used for dividing the 1.25 GHz CML clock received from the MUX/DeMUX module, and 16 are used for dividing the 622 MHz LVDS output clock signals from the DeMUX stage (will be explained shortly). The MC100EP32 divider supplied by ONSEMI (Motorola) was selected, as this divider functions at the specified frequencies (and above), and the MC100EP32 divider is already used within Tellabs. The data sheet for MC100EP32 can be seen in appendix B.2.

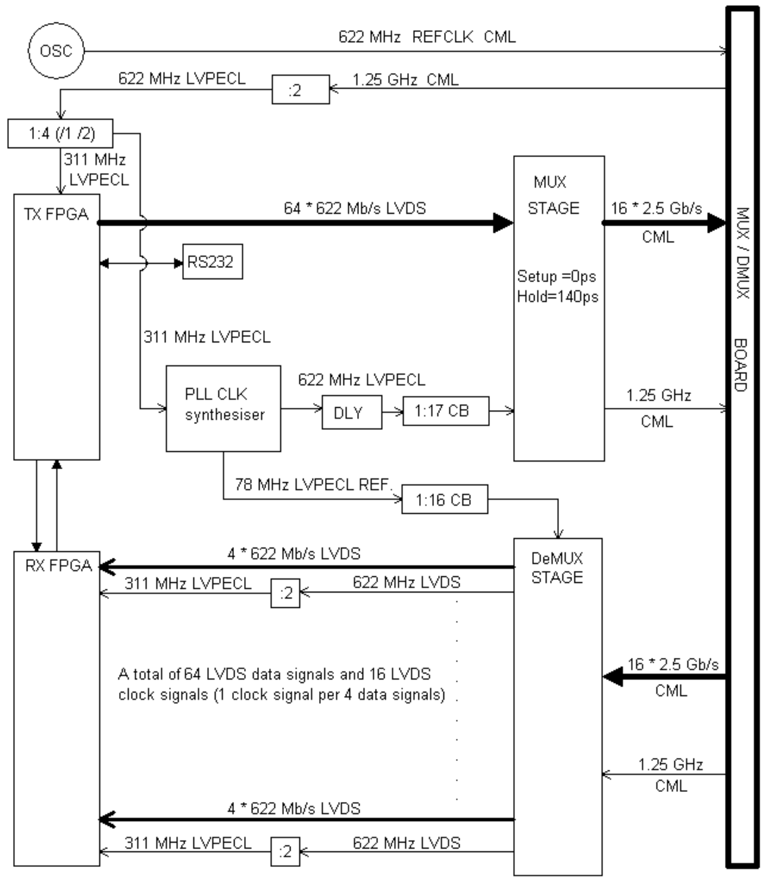


Figure 4.8: Test module block diagram modified according to the demands imposed by the selected components

4.3.3 Transmitter FPGA

The FPGA selected as transmitter FPGA (TX FPGA) is a XC2V3000 (Virtex II series) FPGA supplied by the company Xilinx. With the XC2V3000 FPGA it is possible to output data at a rate of up to 840 Mb/s using DDR and differential LVDS I/O's, and theoretically it is possible to output 64 differential LVDS high speed I/O's from the same physical edge of the FPGA. Transmitting the 64 high speed I/O's from the same physical edge of the FPGA is desired in order to ease the PCB layout. The 64 differential high speed LVDS I/O's have to be synchronous, thus transmitting them from the same physical edge of the FPGA will ease the task of making the 128 PCB traces equal in length.

The Virtex II FPGAs facilitate between 4 and 12 digital clock managers (DCMs) depending on the size of the FPGA. Some important features available in the DCMs are clock dividing, clock multiplying, clock deskewing and clock phase shifting. Furthermore the Virtex II FPGAs can handle up to 16 global clock domains, and provide 16 BUFGs, which are clock buffers used to connected clock signals to the global clock routing. The size of the transmitter FPGA is decided mainly by the high I/O count demand, thus plenty of logical resources will be available for implementing the PRBS generator and the RS-232 UART.

The Virtex II FPGAs exists in 3 different speed grades. The lowest speed grade is 4 and the highest is 6. The speed grade is a measure of the performance of the internal logic in the FPGA. Clock skew, maximal clock frequency, and switching characteristics of synchronous elements in the FPGA are important parameters described by the speed grade. Based on the number and speed of LVDS I/Os needed in this design, speed grade 5 was recommended by Xilinx for the transmitter FPGA. According to Xilinx, speed grade 5 results in a maximum output to output skew of 400 ps, when using LVDS I/Os. As it will be shown in section 4.6, this allows for synchronous sampling of all 64 LVDS I/Os. The data sheets for the XC2V3000 FPGA can be found on the CD that came with this report.

4.3.4 PLL clock synthesizer

Constructing PLLs using discrete components is not an easy task, and it would be beyond the scope of this project. Instead an integrated solution was chosen. The GD16590 PLL clock synthesizer supplied by GIGA has a wide range of input reference clock frequencies, and using this reference clock it generates 6 different output clock signals with different frequencies. The reference clock frequency can range from 19 MHz to 622 MHz, and the output frequencies range from 19 MHz to 1.25 GHz depending on the reference frequency used. The GD16590 has an external loop filter consisting of a resistor and a capacitor, and values for these two components are specified in the data sheet. Using the specified loop filter guaranties that the output clock signals meet the ITU-T recommendations regarding jitter. The choice of the GD16590 PLL clock synthesizer is done on the basis of technical support from GIGA, and experience within Tellabs. The GD16590 accepts LVPECL clock signals as input, and generates LVPECL clock signals as output. The data sheet for GD16590 can be seen in appendix B.3.

4.3.5 Multiple output clock divider

The MC100LVEL37 clock buffer/divider (1:4 (/1 /2)) supplied by ONSEMI (Motorola), was selected for dividing the 622 MHz LVPECL clock signal, originally transmitted from the MUX/DeMUX module as a 1.25 GHz CML clock signal, by two. The reason for implementing the "divide by 4" function in two stages, is that components already existing within Tellabs could be used in this solution. The MC100LVEL37 has 4 clock outputs of which two are at the same frequency as the input clock, and two are half the frequency of the input clock. The MC100LVEL37 can function from both negative and positive 3.3 V power supplies, and the input and output clock signals can be either LVPECL or LVNECL signals. The data sheet for MC100LVEL37 can been seen in appendix B.4.

4.3.6 Delay line

The MC100EP195 programmable delay chip (DLY) was chosen for implementing the DLY stage. The MC100EP195 is an ECL based chip that can function from both negative and positive 3.3 V and 5 V power supplies. The delay is adjusted digitally by setting a 10 bit digital word, and this 10 bit word can be controlled using either TTL, CMOS or ECL voltage levels. This makes it possible to control the delay from one of the FPGAs. The resolution of the delay is as low a 10 ps, and the total delay can be as large as 10.2 ns. A resolution as high as 10 ps, allows for precise fine tuning of the clock/data alignment on the MUX stage input. The delay line will function at frequencies higher than 2.5 GHz, which is more than sufficient for delaying the 622 MHz clock signal on the test module. The data sheet for the MC100EP195 programmable delay chip can be seen in appendix B.5.

4.3.7 Multiplex Stage

As mentioned previously, the multiplex (MUX) stage has to accept 64 622 Mb/s differential LVDS signals, and multiplex them into 16 2.5 Gb/s differential CML data signals together with one 1.25 GHz clock signal. This is not a trivial interface, and it cannot be fulfilled by a single multiplexer. Thus several multiplexers are needed to implement the MUX stage. The MUX stage was chosen to consist of 17 4:1 multiplexers (LXT16653) supplied by the company GIGA.

The 17th multiplexer is needed in order to generate the 1.25 GHz clock signal demanded by the 40 Gb/s interface. Generating the clock in the same way as the data is generated, insures a well defined timing relationship between the clock and the data. The 1.25 GHz clock signal could have been generated by dividing the 2.5 GHz clock signal that the LXT16653 outputs along with the data, but dividing the clock signal would introduce an uncertainty in the timing relationship between the clock and the data.

One of the main reasons that the LXT16653 multiplexer was selected, is that several multiplexers can be synchronized with respect to each other, and that the device to device skew, according to GIGA, should be below 50 ps. That is, the phase on the data signals from two different LXT16653 multiplexers, will not deviate more than 50 ps when used correctly. This is necessary in order to achieve the synchronous interface demanded by the 40 Gb/s interface to the MUX/DeMUX module. Details concerning the timing of the MUX stage output will be addressed in section 4.6, where skew resulting from differences in PCB trace lengths is specified. Furthermore the LXT16653 multiplexer accepts 622 Mb/s LVDS signals as input, and outputs a 2.5 GHz CML data signal as desired. The output common mode voltage level for the LXT16653 exceeds the 1.8 V specified in the 40 Gb/s interface specification, but this can be overcome by using AC coupling.

In order to synchronize the multiplexers, the clock signals interfacing the multiplexers have to be connected in an other way than the one described in the data sheet. Figure 4.9 shows a block diagram of the multiplexer, and the external connections necessary to synchronize several multiplexers are shown too. The data sheet for the multiplexer can be seen in appendix B.6.

The LXT16653 multiplexer, specifically the mux address counter, cannot be reset. But the subdivided clock from the 2.5 GHz CMU VCO (se figure 4.9), which is provided on CNTCK/CNTCKN is uniquely phase related to this address counter. Using the DPA PFC it is possible to slave this subdivided clock to the incoming 622 MHz LVDS clock. In order to insure that all 17 multiplexers are synchronous, the clock signals used to slave the CMU VCOs inside the multiplexers, has to be synchronous too. The dimension of the loop filter connected between the TXCHP and VCTL PINS on the multiplexer, was recommended by GIGA to be $R = 2.15k\Omega$ and $C = 100nF$ at the specified frequency.

4.3.8 Demultiplex Stage

The demultiplex (DeMUX) stage has to constitute the reverse function of the MUX stage, and no single component exists for this. 16 LXT16642 demultiplexers supplied by GIGA were selected for implementing the DeMUX stage.

The LXT16642 is an integrated clock and data recovery (CDR) device with an on chip 1:4 demultiplexer. The LXT16642 uses a 39 MHz reference clock signal to lock onto the incoming data, and when it has achieved lock, the recovered data signals are output together with the recovered clock. The minimum number of consecutive identical bits (CID) that have to occur before the CDR circuitry looses lock, is 400. This is sufficient, as a maximum of 31 identical bits will occur in a $2^{31} - 1$ bit PRBS.

Several LXT16642 devices supplied with the same data and clock signals will not necessarily achieve lock at the same time, and it is not possible to reset the address counter for the DeMUX. Thus it is not possible to synchronize several LXT16642 devices. This means that the 64 LVDS data signals output from the DeMUX stage, will not be synchronous, but only groups of 4 signals will be synchronous.

According to the subsequence property for PRBSs, each of the 16 downstream 2.5 Gb/s data signals will constitute identical PRBSs. Therefore,

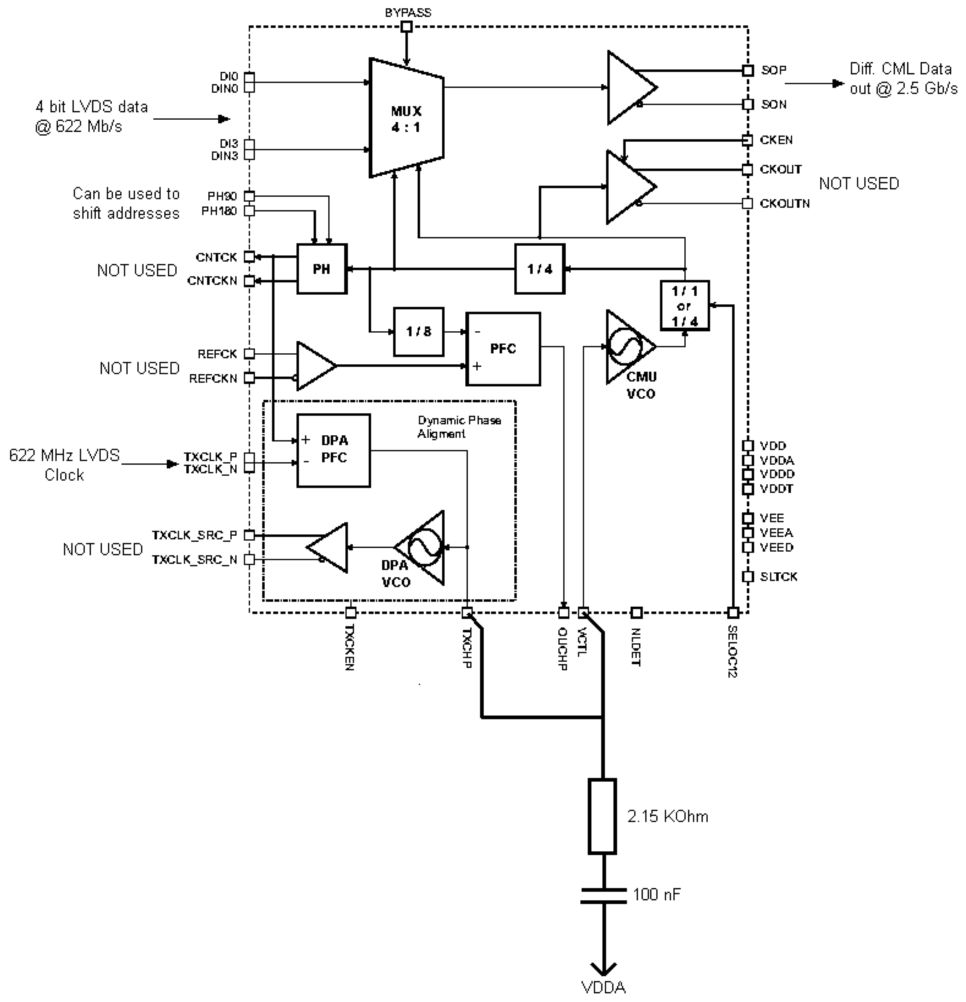


Figure 4.9: The block diagram illustrates the functionality of the LXT16653 multiplexer. The external connections needed in order to synchronize several LXT16653 multiplexers are shown too

the 64 data signal on the low speed side of the LXT16642, can be treated as 16 groups of 4 signals, where each group consists of the same PRBS. As each LXT16642 device recovers a clock signal from the incoming data, each of the 16 identical PRBSs will be represented by its own clock signal. The FPGA RX stage will have to receive the 16 PRBSs separately, and register errors on each of them. The data sheet for the LXT16642 can be seen in appendix B.7.

4.3.9 High fan-out clock buffers

In order to distribute the clock signals for the MUX and DeMUX stage two identical 1:20 clock buffers (1:17 CB and 1:16 CB) from ONSEMI were selected. The product ID for the two clock buffers was NB100LVEP221, and they were supplied as engineering samples (not in production yet) at the time of order. The clock distribution to the MUX stage is critical as the phases of the 17 622 MHz clock signals have direct impact on the phases of the output signals from the MUX stage.

The clock distribution for the DeMUX stage is not critical, because the 16 demultiplexers in the DeMUX stage does not have to be synchronous. The NB100LVEP221 clock buffer has a very low output to output skew of 20 ps, and function at frequencies higher than 1 GHz. The low output to output skew combined with the high fan out, makes the NB100LVEP221 clock buffer well suited for distributing the clock signals for the MUX and DeMUX stages. The data sheet for the NB100LVEP221 clock buffer can be seen in appendix B.8.

4.3.10 Receiver FPGA

Two XC2V1000 FPGAs from Xilinx were selected for implementing the RX FPGA stage. The reason that two FPGAs are needed, is that a total of 16 differential LVDS clock signals is output from the DeMUX stage, and a maximum of 8 can be handled by each FPGA. The differential LVDS clock signals will have to be connected to the global clock routing in the FPGA in order to minimize skew, and only 8 differential inputs to the global clock routing is available on one FPGA.

The XC2V1000 FPGA has the same high speed I/O performance as the XC2V3000 selected for the FPGA TX stage. It is possible to place 32 differential LVDS inputs on the same physical edge of the XC2V1000 FPGA, which yields the same advantages PCB layout wise, as described in section 4.3.3. The data sheet for the XC2V1000 FPGA is identical to the data sheet for the XC2V3000 FPGA selected for the FPGA TX stage.

4.4 Test Considerations

The first important thing to consider in connection with testing of the test module, is how the signals transmitted from the MUX stage can be connected to the DeMUX stage. The goal is to transmit and receive the signals via the MUX/DeMUX module and the opto module. As the MUX/DeMUX module is being designed and implemented in an other master thesis running in parallel with this master thesis, an independent solution is needed. The solution is a loop back module which was constructed as part of the other master thesis. The function of the loop back module is simple, as it merely connects the transmitted data and clock signals to the respective signals at the receiver end. Some clock generation is also part of the loop back module, and the possibility to connect signals from the loop back module to an oscilloscope exists. Details concerning the loop back module can be found in [6]. If the transmitting part of the test module does not function, it has to be possible to test the receiving part independently. A 2.5 Gb/s test set exists within Tellabs, and using this test set makes it possible to test each of the 16 CML channels on the receiver side independently.

Before designing the schematic layout, it is important to consider how the the test module is to be tested. Which signals are to be measured, what the purpose of the measurement is, and in which way are the signals going to be measured, are important considerations that have to be done. Very often, in order to access certain signals, connectors and test pads have to be added to the layout. Furthermore the equipment available for testing the hardware has to be identified. The performance of the test equipment can influence the kind of, and the amount of additional hardware needed to facilitate the required tests. In the following sections, the test considerations done during the design of the test module, will be described. The names given to the different stages constituting the test module in figure 4.8, will be used throughout this section.

4.4.1 Test equipment

The test equipment available for testing the test module is listed below :

- Oscilloscope with a band width of 40 GHz
- 500 Ω and 50 Ω probes able to measure signals with a frequency up to 5 GHz
- 2.5 Gb/s test set
- Power supply with high output current and adjustable current limiter
- Voltmeter

The oscilloscope requires a trigger signal in order to trig the signal being measured (the input signal), and the trigger signal has to be locked to the input signal. The rising edge of the trigger signal indicates, to the oscilloscope, when the period of the input signal starts. Every time a rising edge occurs on the trigger signal, the oscilloscope displays a new image (on top of the old one) of the waveform present on its inputs. Therefore, the waveforms present on the oscilloscope input on every rising edge of the trigger signal, must be identical. Otherwise the oscilloscope will display an image of different waveforms drawn on top of each other. Thus the trigger signal should have the same period as the input signal, or have a period that is attainable by dividing the period of the input signal by a positive even integer. This will ensure that every time a rising edge occurs on the trigger signal, one or an even amount of periods of the input signal have passed. The 622 MHz reference clock signal derived from the 1.25 GHz CML clock signal received from the MUX/DeMUX module, can be used as trigger signal. Furthermore it is possible to use one of the 2.5 Gb/s data outputs from the MUX stage as trigger signal. This signal is accessible through the loop back module. Using this as trigger signal makes it possible to generate a trigger signal of any period, simply by altering the signal input to the MUX stage from the TX FPGA. Trigger signals appropriate for displaying a period of a PRBS are possible to generate using this method. High frequency connectors should be used to output the trigger signals from the PCB.

Signals, that one desires to measure, is usually input to the oscilloscope using a probe. Connecting a probe to a signal loads the signal, and this can cause undesired reflections when used on high frequency signals. The probe will normally act in parallel with a termination resistor, or with the input impedance of the input structure of a component. In both cases the probe will alter the impedance used to terminate the transmission line. In order for the probe not to cause a change in impedance higher than 10%, the impedance of the probe should be at least 10 times greater than the impedance it acts in parallel with. A change in impedance of 10% is considered acceptable, as the reflection caused by this change will be small. The size of the reflection is determined by the reflection coefficient, see equation 4.5 [9].

$$\frac{Z - Z_0}{Z + Z_0} = \frac{(50||500) - 50}{(50||500) + 50} = -0.053 \tag{4.5}$$

Where Z (45 Ω) is the equivalent impedance made up by the probe impedance in parallel with the termination resistor (50 Ω). As the result in equation 4.5 shows, changing the impedance of the termination by 10%, results in a reflection coefficient of -5%. On the test module, the most frequently used value for termination resistors, is 50 Ω. Thus a 500 Ω probe should be used for the measurements. Use of a 50 Ω probe would require insertion of a series resistor with a value of ideally 450 Ω and a capacitor, as this would ease the resistive load on the signal corresponding to the use of a 500 Ω probe. A reflection coefficient of -5% will result in a reduction of 5 % in the amplitude of the measured signal, as 5 % will be reflected and transmitted back to the transmitter.

4.4.2 Test pads and signals

Very often, the majority of signals are accessible through the terminal of a resistor or capacitor, but where this is not the case, a test pad has to be added. When placing a test pad it is important not to capacitively load the target signal in an undesired way. Thus the size of the test pad should be as small as possible, but naturally large enough for the probe to connect. A test pad size of 1 mm^2 is considered acceptable.

Generally all signals should be accessible for test purposes, but some signals have higher priority than others. Access to signals that are part of an interface where the timing is critical is important. It has to be possible to verify that the timing requirements specified for the design are met. All clock signals should be accessible, as clock signals are vital for the function of any synchronous system.

On the test module, the interface between the TX FPGA and the MUX stage is critical, and it has to be possible to measure the phases of the clock and data signals on the MUX stage input. It has to be possible to compare the phases of the 64 data signals, and the phases of 17 clock signals. Furthermore the phase of a data signal on a multiplexer input has to be comparable to the phase of a clock signal on the same multiplexer. As the 64 LVDS data signals on the MUX stage input are terminated internally in the multiplexers, test pads have to be added to these signals. Adding 64 test pads will complicate the PCB layout unnecessarily, thus it has been chosen to add only one test pad at the input of every multiplexer. As the 4 data signals input to one multiplexer originates from the same output block in the FPGA (will be explained in detail later), the skew between these data signals is expected to be acceptable. The clock input on the multiplexers is not internally terminated, and therefore access to these clock signals will be possible through a termination resistor placed close to the multiplexer.

The MUX stage output has to be accessible, and it has to be possible to measure and compare the phases of the 16 2.5 Gb/s CML output data signals, in order to verify that the MUX stage output meets the 40 Gb/s interface specification. The data signals on the MUX stage output, will be accessible through the AC coupling capacitors needed to protect the MUX/DeMUX module, from the output common mode voltage of the MUX stage.

The interface between the DeMUX stage and FPGA RX stage is critical, and the phases of the LVDS data and clock signals output from the DeMUX stage, have to be accessible at the input of the RX FPGAs. The RX FPGAs are housed in BGA (Ball Grid Array) packages, which means that in order to connect a trace to a pin on the RX FPGA package, a via will be needed. Hence, the data and clock signals on the input of the RX FPGAs will all be accessible through these vias.

When testing a design that includes FPGAs, it has to be possible to monitor internal signals in the FPGA. This can be facilitated by connecting some of the pins on the FPGAs to test pads. In order to monitor clock signals, it has to be possible to connect a 100 Ω resistor between the test outputs, as clock signals usually are output as LVDS signals. In order to change the configuration of the internal logic in the FPGA during a test, a DIP switch and a reset push button should be connected to the FPGA. The DIP switch settings can be made to decide the function of the internal logic in the FPGA. This will make it possible to test several models of a specific circuit in the FPGA during a test, or make it possible to change the data being transmitted during a test. A lot of time can be saved this way, as changes that usually require reconfiguration of the FPGA, can be performed by changing the configuration of the DIP switch.

The considerations described above, insure that the hardware needed in order to test the test module, has been identified. This hardware can now be included in the schematic layout.

4.5 Schematic Layout

As mentioned in the beginning of this chapter, the schematic layout requires interfacing the selected components to each other and the surroundings. The test module interfaces its surroundings (the MUX/DeMUX module) through the 40 Gb/s interface. This interface has been specified in terms of signal levels and timing relations, and the physical backplane used to implement the 40 Gb/s interface, is a well defined backplane already used within Tellabs. The method used for interfacing components based on different technologies, has been described. Based on this information, and on information obtained from the component data sheets and application notes, the schematic layout has been made. The schematic layout can be seen in appendix A, and table 4.2 provides an overview of the sheets in the schematic layout. Some control

Sheet Nr.	Sheet Title	Description
1	40 Gb/s PRBS Tester	Top sheet
2	Back Plane	Back plane connections
3	Multiplexer/Demultiplexer clocks	Clock distribution for MUXs and DeMUXs
4	Demultiplexers 0-7	First 8 demultiplexers
5	Demultiplexers 8-15	Last 8 demultiplexers
6	Receiver FPGA 1	The first receiver FPGA
7	Receiver FPGA 2	The second receiver FPGA
8	Reference clock gen.	Reference clock TX/RX
9	Receiver Clock Divide	Division of the 16 RX clocks
10	Transmitter FPGA	Transmitter FPGA (1st sheet)
11	Transmitter FPGA	Transmitter FPGA (2nd sheet)
12	Multiplexers 0-7	First 8 multiplexers
13	Multiplexers 8-15	Last 8 multiplexers
14	CML transmit clock gen.	Generation of the 1.25 GHz CML TX clock
15	Power supply and 48V filter	Power supply

Table 4.2: The table gives an overview of the sheets in the schematic layout, and a short description of every sheet is included in the table.

and status signals, together with the architecture of the power supply, have not been addressed yet. This, together with some general information about the schematic layout, will be addressed in this section.

4.5.1 The Power Supply

With all the components selected, the requirements for the power supply can be specified. Label and signal names written in the `verbatim font` refers to signals and labels in the schematic layout. First, the minimum amount of current needed from the power supply, and the supply voltage levels needed, have to be specified. Table 4.3 shows the total current drawn by each component, and which supply voltage each component in the design uses. According to table 4.3, both a 3.3 V supply and a 1.5 V supply is

Component ID	Component Description	Number of components	Total supply current [mA]	Supply voltage [V]
LXT16653	MUX	17	2210	3.3
LXT16642	DeMUX	16	1760	3.3
NB100LVEL221	Clock Buffer	2	260	3.3
GD16590	PLL Clock synth.	1	165	3.3
VS500A	VCSO	1	140	3.3
MC100EP32	Clock divider	17	680	3.3
MAX3221	RS-232 Lvl. conv.	1	1	3.3
Bias and Termination	Resistors		2000	3.3
XC2V3000	FPGA	1	6000*	1.5
XC2V3000	FPGA	1	400*	3.3
XC2V1000	FPGA	2	6000*	1.5
XC2V1000	FPGA	2	300*	3.3
MC100EP195	Delay line	1	180	3.3
XC18V04	PROM	4	100	3.3
MC100LVEL37	Clock divider	1	55	3.3

Table 4.3: The table shows the current consumed by every component in the design. Termination and bias resistors have been included too. The values specified are the minimum requirements for the power supply. Entries marked with an "*" are estimated values.

needed in order to supply all the components selected to implement the test module.

The current consumed by the FPGA was estimated by considering the number of LVDS I/Os implemented in the FPGA (each LVDS I/O uses a maximum of 4 mA as described earlier), and considering the current consumed by other FPGA designs of similar size. The minimum current that the two supplies have to source is shown in table 4.4. Added to the requirements specified in table 4.4, a peak in the current consumption caused by the FPGAs, might occur on power up. If the VCCO supply on the FPGAs (refer to the FPGA data sheet for supply names and descriptions) is turned on before the VCCAUX supply, each bank in the FPGAs will consume 300 mA each. These supplies will be turned on simultaneously, but for the sake of safety, the power supply is constructed so it is able to handle this peak

Supply Voltage [V]	Total current [A]
1.5	12
3.3	8.251

Table 4.4: The table shows the minimum amount of current the 3.3 V and 1.5 V supplies have to source.

in the current consumption.

The components used to implement the power supply are standard DC/DC converters supplied by the company Synqor, and one step down controller (MAX797) supplied by Maxim. The DC/DC converters have a very high efficiency, and their physical size makes them well suited for the design. The schematic layout for the power supply can be seen on sheet 15 in appendix A. As the schematic shows, the power supply consists of 8 individual outlets (three 1.5 V supplies, four 3.3 V supplies, and one 5 V supply). The 5 V supply is used to supply the 3.3 V supply implemented with the MAX797 step down controller, and it also supplies the RS-232 connector. The reason for dividing the supply into several individual supplies, is to isolate the noise generated on the supplies from each other. The noise is generated by the switching outputs of the components. The `+3V3T` supply supplies the LVPECL terminations only, as these terminations often generate noise. The `+3V3C` supplies only the sensitive oscillator circuits, as these circuits are very sensitive to noise. The `+3V3` and `+3V3B` supplies supply the FPGAs and the rest of the components. The `+1V5`, `+1.5VA`, and `+1.5VB` supplies supply the internal logic in the FPGAs.

Local power supplies

The main power supply just described is not sufficient to supply the high speed components. Decoupling of the power supply pins close to the components, is necessary to avoid voltage dips caused by the shorting current during output switching. Recommended values for the decoupling capacitors can be found in the component data sheets, or in application notes supplied by the component suppliers. However, because of practical issues in connection with mounting the components on the PCB, only few different sizes of capacitors should be used for decoupling. Each time a new capacitor value is taken into use, the PCB mounting equipment has to be reconfigured, which increases the cost and time spend on the assembly. A combination of 100 nF and 100 pF capacitors has been used for decoupling the supplies close to the components. These capacitors are housed in very small SMD packages, resulting in a very low induction.

Further away from the component supply pins, 10 μF and 100 μF capacitors are placed. These larger capacitors function as a charge depot for the small capacitors, and are able to supply the small capacitors with charge faster than the main supply. Furthermore inductors are inserted to create noise attenuating low pass filters. By using these filters, it is possible to restrain the noise in a controlled way. The dimension of the filters are based on experience from within Tellabs, and the 3db frequency for the filters varies from about 100 KHz to about 400 KHz.

4.5.2 Control and Status Signals

Some control and status signals can be seen on the schematic layout in appendix A, and some of them deserve some explanation. The `LOCK` signal is an output signal from the LXT16642 Demultiplexer. It indicates if the CDR circuitry in the demultiplexer has achieved lock. The `LOCK` signals are input to the transmitter FPGA so that it is possible to react upon a failure to achieve lock.

The `del` signals are input signals to the MC100EP195 delay line. The state of these signals determine the size of the delay that the delay line imposes on its input signal. The `del` signals are controlled from the TX FPGA. The `MUXCLK_LOCK` signal is an output signal from the GD16590 PLL clock synthesizer, and it indicates if the GD16590 has locked to the reference clock. The `MUXCLK_LOCK` signal is input to the TX FPGA.

The `LFA` signals are included in order to be able to adjust the phase of the 16 2.5 Gb/s signals on the MUX stage output individually. The signals are connected to a circuit as shown in figure 4.10 The resistors "R1" and

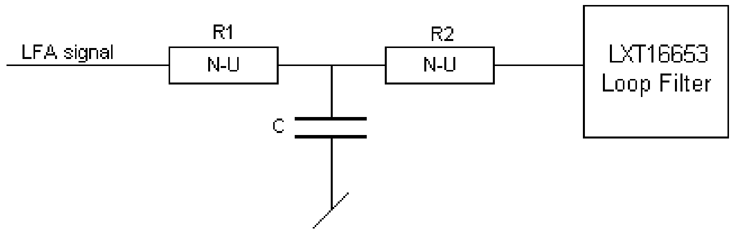


Figure 4.10: The figure illustrates the circuitry used to create a charge leakage from the loop filters on the multiplexers. Creating a controlled charge leakage will alter the phase of the signal on the multiplexer outputs.

"R2" shown in figure 4.10 are by default not used (N-U, which means that they are not present on the PCB), and will only be mounted if it proves necessary to adjust the phases on the MUX stage output. A bit sequence will be transmitted on the LFA signal, and by controlling the composition (number of high bits and number of low bits) of this sequence, it is possible to control the voltage across the capacitor C in figure 4.10. The voltage across C will determine the current running in the resistor R2, and thus determine the amount of charge leaking from the loop filter. Removing charge from the loop filter will cause the PLL circuitry in the multiplexer to alter the phase of the CMU VCO output clock signal, which directly affects the phase of the multiplexer output.

4.5.3 Test pads and connectors

According to the considerations made in section 4.4, several test pads and one high frequency connector has been included in the schematic layout. The test pads are labelled `TP#` where "#" is a four digit number. A high frequency connector is used to output the trigger signal from the PCB, and is included on sheet 8 in the schematic layout. The connector is labelled `J0001`.

4.5.4 Miscellaneous

The PINS chosen as LVDS I/Os on the TX FPGA and on the RX FPGAs, are determined by the placement of the high speed logic inside the FPGA. It is vital that the PINS chosen as I/Os are placed close to the high speed logic transmitting or receiving the data and clock signals. Further details regarding the placement of the internal high speed logic relative to the PINs on the FPGA, will be given in section 5.2. There are a number of PROMs (XC18V04) included in the schematic layout (sheet 6,7,10). They are used for configuring the FPGAs, and they are supplied by Xilinx. The interface between the PROMs and the FPGAs is well described in application notes supplied by Xilinx. The JTAG connectors `J0007`, `J0008`, and `J0009` on sheet 6-7 and sheet 10, are connectors used to connect a parallel cable to the FPGAs. This cable is used for configuring the FPGAs from a PC.

4.6 Printed Circuit Board

Making the PCB layout for the test module is beyond the scope of this thesis, as the PCB layout is a discipline of its own. However some requirements regarding the trace length of critical signals need to be specified, before the PCB layout can commence. The maximum allowable trace length differences will be specified in terms of skew measured in ps. The trace lengths of data signals that are part of a synchronous interface need to be controlled timing wise, in order not to corrupt the timing specification of that interface.

The maximum allowable deviation in trace lengths between the 64 622 Mb/s LVDS data signal on the TX FPGA output has to be specified. The setup time for the LXT16653 multiplexer is 0 ps and the hold time is 140 ps. According to Xilinx the maximum skew between any two of the 64 LVDS outputs is 400 ps. Recalling the the maximum transition time for a LVDS signal is 500 ps, the resulting valid data window can be calculated as $1600ps - 500ps - 140ps - 400ps = 560ps$. A valid data window of 400 ps is considered acceptable, thus the skew resulting from differences in trace lengths, should not exceed 160 ps. Any jitter present on the clock signal supplied to the multiplexers, will reduce the size of the valid data window. As the size of the valid data window has been set to 400 ps, the data can be sampled correctly even with a considerable amount of jitter present.

The maximum allowable skew on the MUX stage output (2.5 Gb/s signals) is theoretically 250 ps, as the FIFO that the MUX stage interfaces has a setup and hold time requirement of 75 ps each. The maximum device to device skew of the multiplexers in the MUX stage is 50 ps, and the typical output to output skew on the clock buffer supplying the MUX stage with the 17 622 MHz clock signals, is 20 ps. As the phases of the 17 622 MHz clock signals directly affects the phases of the MUX stage output signals, the total skew caused by the multiplexers and the clock buffer is 70 ps ($50ps + 20ps$). The rise and fall time on the MUX stage output signals is 100 ps according to GIGA. A valid data window of 200 ps is considered acceptable on the MUX stage output signals, thus the skew added to the MUX stage output signals by differences in the PCB trace lengths, can be calculated as

$$T_{trace} = T_{2.5Gb/s} - T_{valid} - t_{rf} - T_{skew} \tag{4.6}$$

Where T_{trace} is the maximum allowable skew resulting from differences in PCB trace lengths, $T_{2.5Gb/s}$ is the period of a 2.5 Gb/s data signal (400 ps), T_{valid} is the acceptable size of the valid data window, t_{rf} is the rise/fall time on the MUX stage output signals, and T_{skew} is the skew resulting from the multiplexer device to device skew, added to the output to output skew of the clock buffer supplying the MUX stage with the 17 622 MHz clock signals. Using the values outlined above yields $T_{trace} = 30ps$. This means that no more than 15 ps skew, peak to peak, resulting from PCB trace length differences can be tolerated on the traces connecting the 17 622 MHz clock signals to the MUX stage, and on the 17 2.5 Gb/s signals on the MUX stage output. Using the above values a valid data window of 200 ps will be accomplished. However some jitter will be present on the 1.25 GHz CML clock signal transmitted from the 17th multiplexer. The amplitude of this jitter must not exceed 50 ps (theoretically), as this would result in violation of the setup and hold time requirement for the FIFO on the MUX/DeMUX module. Assuming a signal velocity in the interval 14 cm/ns - 20 cm/ns on the PCB, the 34 PCB traces connecting the 17 differential clock signals to the MUX stage must not deviate more than 2 mm - 3 mm from each other, and the same demand is imposed on the 34 traces connecting the 17 MUX stage output signals to the back plane.

On the receiver side of the test module the demands are not as strict as on the transmitter side. The 16 demultiplexers in the DeMUX stage are not synchronous, thus no special demands are imposed on the 16 2.5 Gb/s differential signal traces connecting the back plane to the DeMUX stage.

The trace lengths of the 622 Mb/s LVDS signals output from the DeMUX stage, is subject to the same demands as the trace lengths of the 622 Mb/s signals on the TX FPGA stage output. But only groups of 4 signals is subject to these demands, and no demands exists for the trace lengths of one group of signals compared to the trace lengths of an other group of signals. The trace length of the 622 MHz clock signals output from the DeMUX stage is not important, as these signals can be delayed inside the FPGA, in order to sample the incoming data correctly.

The trace lengths of PCB traces between test pads and components, have to be controlled where phase comparisons are to be made. The length of the traces connecting the test pads to the data inputs of the multiplexers in the MUX stage, has to be the same on all 17 multiplexers. Furthermore the termination resistors terminating the 622 MHz clock inputs on the multiplexers, have to be placed the same distance away from the multiplexer on all 17 multiplexers. This is necessary as the termination resistors will function as test points, when measuring the phase of the clock signals. The AC coupling capacitors on the multiplexer outputs has to be placed the same distance away from the multiplexer on all 17 multiplexers, as the phase of the multiplexer output signals have to be compared.

The test module PCB with all components mounted can be seen in figure 4.11 (TOP), and figure 4.12 (bottom). The PCB consists of 32 layers, and is roughly 4 mm thick.

4.7 Summary

The technology, on which the components selected to implement the test module are based, has been described. Methods for interfacing components based on different technologies has been explained. Based on the requirements specified in chapter 3, and on practical issues related to component assembly and PCB layout, components for implementing the test module have been selected. Considerations regarding the testing of the test module have been done, and the additional hardware and test pads needed to test the test module, have been identified. Using information from the component data sheets, and some estimation, the requirements for the power supply have been specified. The schematic layout of the test module has been described, and table 4.2 gives an overview of the schematic layout. Finally considerations regarding the PCB layout have been done, and the necessary requirements for the PCB layout have been specified.

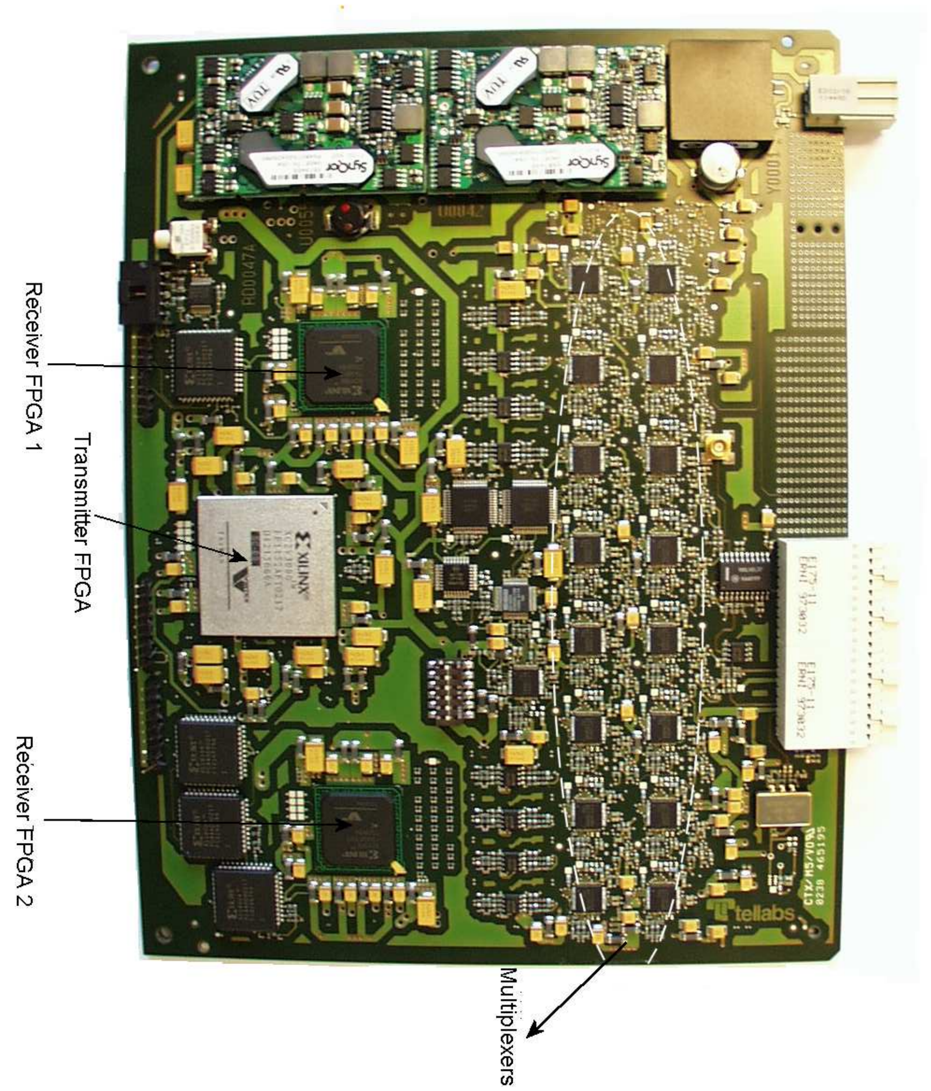


Figure 4.11: The figure shows a top view of the test module PCB

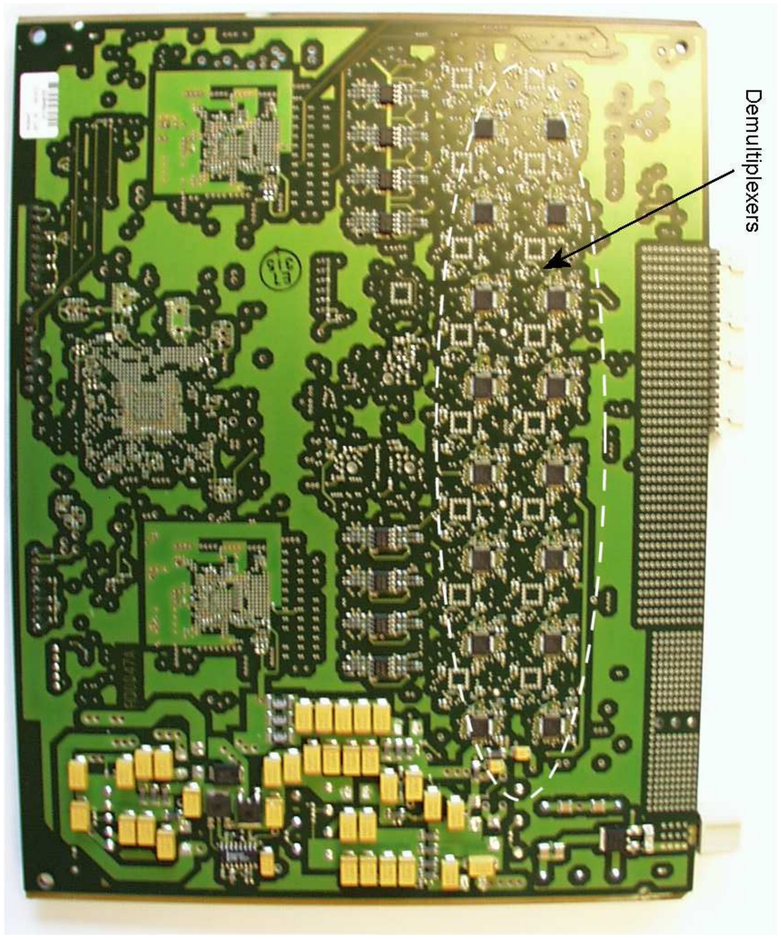


Figure 4.12: The figure shows a bottom view of the test module PCB.

Chapter 5

FPGA Design

Designing the hardware that is to be implemented in an FPGA, includes several tasks. Based on the requirements for the design, a block diagram illustrating the required functionality can be constructed, and VHDL models for the required hardware can be designed.

A VHDL model consists of an entity and an architecture. The entity defines the input and output ports of the model, and the architecture defines the behavior of the model. Each VHDL entity can be compared to a discrete component, and it is important to specify the interfaces between the VHDL entities. When the VHDL models are ready, simulation can begin. Simulation of the VHDL models is used to verify that they behave as intended. Using a test bench and the appropriate test stimuli, the VHDL models can be tested using a simulation tool. The initial simulations do not take into account the delay information inherent in the hardware, but only accounts for the functionality of the models.

After having verified that the VHDL models behave as intended, synthesis of the design can begin. The synthesis transforms the VHDL models into physical hardware, and outputs information regarding the timing of the design. The timing information that the synthesis tool provides, is based on the constraints and information given to the tool prior to the synthesis. The synthesis tool has to be informed about clock frequencies, and optimization options provided by the tool, have to be configured. After having performed a successful synthesis, the hardware generated (provided as a file) is input to the place and route tool supplied by the FPGA supplier. The place and route tool places the hardware generated by the synthesis tool on the FPGA floor plan. In order for the place and route tool to place the hardware as desired, all critical hardware in the FPGA design should be constrained. The place and route tool outputs a file that can be used for static timing analysis and back annotation. The static timing analysis will show if the hardware generated, functions at the required frequencies, and if all constraints are obeyed. The back annotation can be used to verify that the hardware generated has the functionality specified in the VHDL models.

5.1 General

As described previously, the PRBS has been chosen to be generated and received using Xilinx Virtex II FPGAs. Implementing a PRBS generator or receiver in a FPGA and making it run at 80 MHz is not critical, but transmitting or receiving the PRBS at a rate of 40 Gb/s from the FPGA, using 64 622 Mb/s LVDS I/Os is critical. It requires using specifically designated hardware resources inside the FPGA, and constraining the placement of this hardware correctly. Xilinx provides VHDL macros that enable transmission and reception of data at a rate of up to 840 Mb/s. The basic macro is a 4 bit macro, and several macros can be cascaded to enable transmission of more than 4 bits at a rate of up to 840 Mb/s. These macros have been used in the design of the PRBS transmitter and receiver. In this chapter the design, simulation, test and implementation of the FPGA designs will be described, but first the 4 bit macros used in the designs will be described.

The VHDL code for the models described in this chapter is included on the CD that came with this report.

5.2 4 bit transmitter macro

The 4 bit transmitter macro (TX macro) functions as a 32:4 multiplexer, and takes 32 bits of data synchronous to an input clock as input, and provides 4 bits of data at 8 times the input clock frequency as output. Several input signals are needed for the the TX macro to function. Figure 5.1 shows the TX macro entity. Signal names written in the `verbatim` font, refer to signals in the VHDL code. The `clk` signal is used to input the 32 bits of

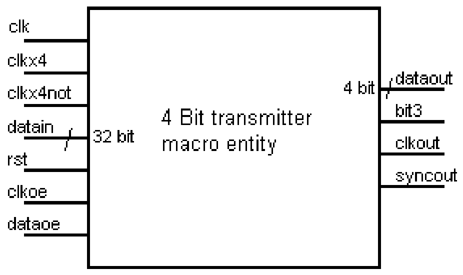


Figure 5.1: The figure shows the TX macro entity

data (`datain`) on the TX macro input, thus the `clk` signal and the `datain` signals have to be synchronous. The `clkx4` signal is a clock signal with a frequency four times higher than the frequency of the `clk` signal. The `clkx4not` signal is an inverted version of `clkx4`. The rising edges of the `clkx4` and `clkx4not` signals are used to output data from the TX macro. Using both edges of a clock signal to transmit data with, is referred to as double data rate.

The `clk`, `clkx4`, and `clkx4not` signals are usually generated using one of the DCMs in the FPGA. In this way they will be synchronous, and a minimum of skew will exist between them. An overview of the input and output signals on the TX macro entity, together with a short description of each signal, is shown in table 5.1. The framing signal on the macro output

Name	Direction	Description
clk	Input	Clock input (80 MHz)
clkx4	Input	Clock input (311 MHz)
clkx4not	Input	Inverted clock input (311 MHz)
datain	Input	32 bit parallel data input sync. to clk
rst	Input	Reset signal
clkoe	Input	Output enable for the output clock signal
dataoe	Input	Output enable for the output data
dataout	Output	4 bit parallel data output (622 Mb/s x 4) sync. to clkx4
bit3	Output	Copy of dataout(3) (NOT USED)
clkout	Output	Clock signal (311 MHz)
Syncout	Output	Optional frame signal (NOT USED)

Table 5.1: The table describes the inputs and outputs on the TX macro. The text embraced by parenthesis concerns this design only

is optional, and will not be used in this application. The order of the bits on the TX macro output is LSB first, see figure 5.2. The bit order on the TX macro output can be reversed if needed.

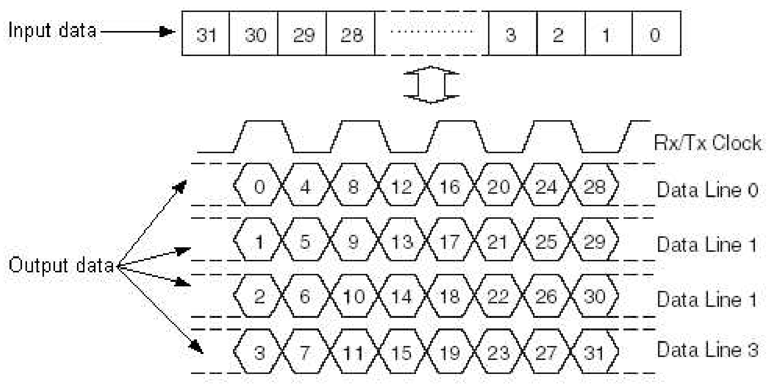


Figure 5.2: The figure shows the bit order on the output of the TX macro. The LSB is transmitted first.(From [11])

The TX macro has to be placed correctly on the FPGA floor plan (the silicon). This is done by specifying the placement (`RLOC_ORIGIN`) of the TX macro in a constraint file, which the Xilinx place and root tool accepts. The physical implementation of the TX macro can be seen in figure 5.3. The squares making up the TX macro in figure 5.3 are slices, and each slice

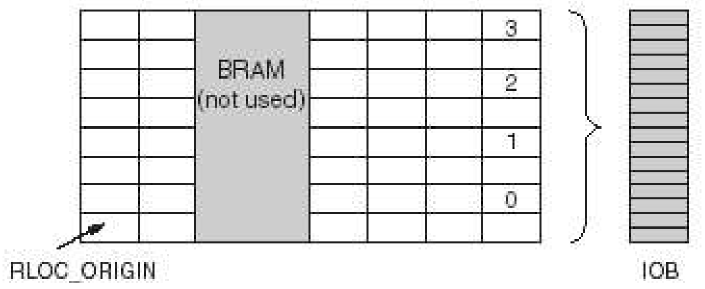


Figure 5.3: The figure shows the physical implementation of the TX macro. The RLOC_ORIGIN must be specified in a constraint file, and determines where on the silicon the macro will be placed.(From [11])

contains similar logic resources. Four slices make up one configurable logic block (CLB), thus the TX macro is 3 CLBs wide and 4 CLBs high. The `rloc` attributes specified in the VHDL code for the TX macro, specifies in which slice within the macro, the logical resources must be located. The TX macro is designed to be placed on the right hand edge of the FPGA. The PINS selected as output on the TX macro must be located within ± 2 CLBs vertically of the macro. The best way to select PINS is by considering the FPGA floor plan. When all the TX macro placements are specified, the PINS can be selected easily using a floor plan viewer. Detailed information regarding the TX macro and its usage can be found in [11], which is included in appendix D.

5.3 4 bit receiver macro

The original 4 bit receiver macro (RX macro) supplied by Xilinx includes signals for synchronizing several macros. In this application the macros will run independently, thus most signals and hardware related to the synchronization feature have been removed from the macro.

The RX macro functions as a 4:32 demultiplexer. The RX macro takes 4 bits of data, synchronous to the input clock, as input, and outputs 32 bits of data synchronous to the output clock. The frequency of the input clock is four times higher than the frequency of the output clock. The RX macro uses a 32 bit wide FIFO as interface between the two clock domains. The RX macro entity can be seen in figure 5.4.

The `rxclk` and `rxclknot` signals are generated using one of the DCMs

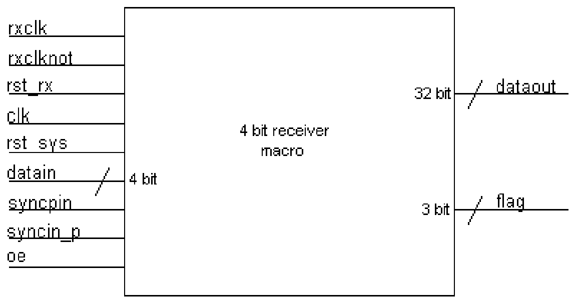


Figure 5.4: The figure shows the RX macro entity.

in the FPGA. The two clock signals are used to input data into the RX macro, and are synchronous to the input data signals.

Using a DCM to create both the `rxclk` signal and the `rxclknot` signal, results in the use of two BUGFs (global clock buffers). As 8 different clock signals will be input to each of the RX FPGAs, 16 BUFGs will be needed if the RX macro shown in figure 5.4 is used. Thus no BUFGs will be available for distributing the system clock, which is not acceptable. The skew on the system clock is critical, which makes it is necessary to distribute the system clock using the global clock routing. In order to solve this problem, a slightly modified version of the macro shown in figure 5.4 was made. The modified version of the RX macro relies on local inversion of the `rxclk` signal in order to create the `rxclknot` signal, hence only the `rxclk` signal has to be distributed from the DCM to the RX macro, thereby using only one BUFG. The only difference between the two RX macro entities is the `rxclknot` input signal, which is removed from the modified RX macro entity. The input and output ports of the RX macro entities is shown in table 5.2.

Name	Direction	Description
rxclk	Input	Input clock from DCM (311 MHz)
rxclknot	Input	Inverted input clock from DCM or locally generated (311 MHz)
rst_rx	Input	Reset signal for resetting receiver circuitry and control signals
clk	Input	Clock signals for outputting data from the FIFO (78 MHz)
rst_sys	Input	Reset signal for resetting FIFO output
datain	Input	4 bit parallel input data (622 Mb/s x 4)
syncpin	Input	Frame signal from PIN (NOT USED = LOW)
syncin_p	Input	Input for synchronization chain (NOT USED = LOW)
oe	Input	Output enable for FIFO
dataout	Ouput	32 bit parallel data output sync. to clk
flag	Ouput	3 bit flag indicating FIFO status : flag(0) = empty to 1/2 flag(1) = 1/4 to 3/4, flag(2) = 1/2 to full

Table 5.2: The table describes the inputs and outputs on the RX macro. The text embraced by parenthesis concerns this design only. The rxclknot signal is not used on the modified version of the RX macro.

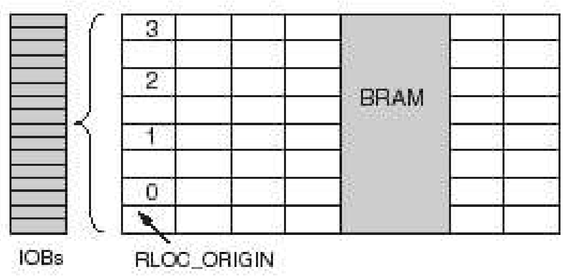


Figure 5.5: The figure shows the physical implementation of the RX macro. The block ram (BRAM) is used to implement the FIFO, which facilitates the clock domain shift in the RX macro.

The placement of the RX macro, and the PINs chosen as inputs, are governed by the same rules as for the TX macro. The RX macro is designed to be placed on the left hand edge of the FPGA. The physical implementation of the receiver macro can be seen in figure 5.5. Detailed information about the RX macro and its usage can be found in [11], which is included in appendix D.

5.4 Top Entity for the TX FPGA

The TX FPGA has to provide a PRBS at a rate of 40 Gb/s on its output, using 64 622 Mb/s LVDS outputs. Based on this requirement, and on the macros supplied by Xilinx, a block diagram describing the required functionality of the TX FPGA has been constructed. This block diagram is shown in figure 5.6. The functionality described by figure 5.6 is contained in the VHDL model of the top level entity for the TX FPGA design.

The top level entity for the TX FPGA design consists of a PRBS generator, a 64 bit transmitter and some logic used to generate reset and enable signals. The PRBS generator generates 512 bits of PRBS synchronous to the `clk` signal, and these 512 bits are transmitted to the 64 bit transmitter. The 64 bit transmitter accepts 512 bits of data synchronous to the `clk` signal as input, and outputs 64 bits of data synchronous to the `clkx4` and `clkx4not` signals. An external 311 MHz clock signal is connected to a DCM, and using this DCM the `clk`, `clkx4`, and `clkx4not` clock signals are generated. The `clkx4` signal has a frequency of 311 MHz, and the `clk` signal has a frequency of 80 MHz. Using a 311 MHz input clock and then dividing it by 1 and 4 in order to create the `clk`, `clkx4`, and `clkx4not` signals, results in the best jitter performance [11].

The reset signal `rst_clk` is controlled by the `clk_locked` signal output from the DCM. This output signal indicates whether the DCM has achieved lock to the incoming clock signal. If the DCM looses lock the logic connected to the `rst_clk` signal will be reset, se figure 5.6. The output clock enable

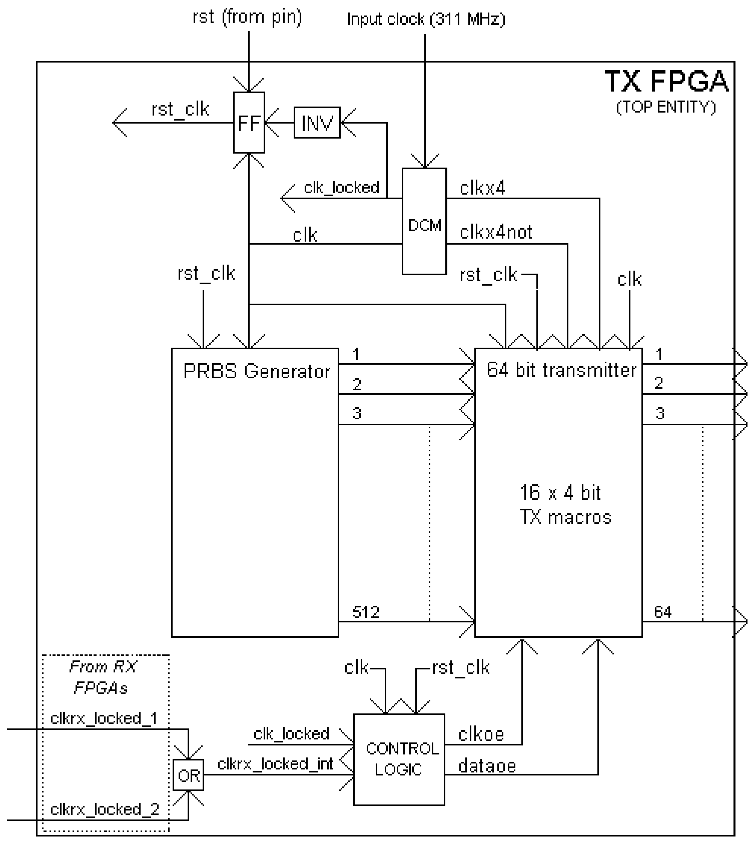


Figure 5.6: The figure shows the functional description of the TX FPGA design. The PRBS generator outputs 512 bits of PRBS on every rising edge of the clk signal. The 64 bit transmitter uses 16 4 bit TX macros to multiplex the 512 bits at 78 Mb/s into 64 bits at 622 Mb/s. The 64 bits at 622 Mb/s are output from the FPGA using LVDS outputs.

signal (`clkoe`), and the data output enable signal (`dataoe`), are controlled by the `clk_locked` signal and the `clkrx_locked` signals. When the TX DCM has started up properly and achieved lock, the 64 bit transmitter starts transmitting the clock signal. This feature is included for simulation purposes, as the output LVDS clock from the TX FPGA is not used on the test module. When all DCMs in the two RX FPGAs has achieved lock, the 64 bit transmitter starts to output the data.

Some signals are not included in figure 5.6, but these are included in table 5.3, which describes all the input and output signals on the top entity. The `error_set_in` signal is connected to an error generating circuit. This circuit is included for simulation and test purposes, and functions by inverting some of the bits in the PRBS that is to be transmitted. This makes it possible to test the error counting functionality in the receiver.

Name	Direction	Description
error_set_in	Input	Input from DIP switch that enables transmitting errors
delay_set_in	Input	Input from DIP switch that enables adjusting the delay settings on the MC100EP195 delay line.
rx_sync	Input	Signal indicating that one of the PRBS receivers has synchronized. Used for test purposes only
clkin_p, clkin_n	Input	Differential 311 MHz clock input
rstin	Input	Reset pushbutton
clkrx_locked_1	Input	Signal indicating the all DCMs in RX FPGA 1 has achieved lock
clkrx_locked_2	Input	Signal indicating the all DCMs in RX FPGA 2 has achieved lock
dataouta_p, dataouta_n	Output	LVDS output data (64 x 622 Mb/s)
clkouta_p, clkouta_n	Output	311 MHz differential LVDS output clock used for simulation purposes only
clktx_locked_out_1	Output	Signal indicating that the TX DCM has achieved lock. The signal is transmitted to the RX FPGA 1
clktx_locked_out_2	Output	Signal indicating that the TX DCM has achieved lock. The signal is transmitted to the RX FPGA 2
delayline_out	Output	11 bit signal for configuring the MC100EP195 delay line. Controlled by `delay_set_in`.

Table 5.3: The table describes the inputs and outputs on top entity of the TX FPGA design

5.4.1 VHDL model for the PRBS Generator

The PRBS generator block in figure 5.6 can be implemented in several different ways. One possibility is to implement 16 identical $2^{31}-1$ LFSRs, and then delay the sequences generated by these 16 LFSRs correctly with respect to each other. This would result in 16 identical PRBSs on the output of the TX FPGA, and as these PRBSs are delayed correctly (subsequence property) with respect to each other, the 16 bit output from the MUX stage would yield the same PRBS of length $2^{31}-1$ bits. An easier way to generate the PRBS is to create one LSFR that generates 512 bits of a $2^{31}-1$ bits long PRBS every clock cycle, and then distribute these bits correctly to the 16 TX macros in the 64 bit transmitter. An implementation of such an LFSR can be seen in figure 5.7.

The VHDL model for the PRBS generator block in figure 5.6 is based on the LFSR shown in figure 5.7. Implementing the LFSR in the FPGA as shown in figure 5.7, and making it function at 80 MHz is not possible, as this would require the registers in the LFSR to be clocked 512 times within one 80 MHz clock cycle, equivalent to 40 GHz operation. Instead a synthesis tool is used to generate the hardware needed to implement the function described by the LFSR in figure 5.7. The input and output ports for the PRBS generator entity are described in table 5.4.

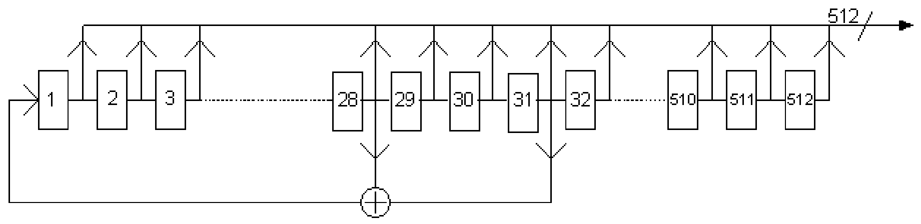


Figure 5.7: The figure shows a LFSR that generates 512 bits of a $2^{31} - 1$ PRBS every clock cycle. The LFSR uses 512 registers, but the feedback tabs used correspond to a $2^{31} - 1$ PRBS.

Name	Direction	Description
reset	Input	Reset signal that initializes the LFSR
clk	Input	80 MHz clock signal
prbs_data	Output	512 bit PRBS synchronous to clk
output_errors	Output	Flag used for simulation purposes

Table 5.4: This table describes the input and output port on the PRBS generator entity

5.4.2 VHDL model for the 64 bit transmitter

As shown in figure 5.6, the 64 bit transmitter has to accept 512 78 Mb/s data signals as input, and convert these into 64 622 Mb/s LVDS data signals on its output. This interface is accomplished by using 16 of the TX macros supplied by Xilinx. The 16 TX macros has to be synchronous, and this is assured by supplying them with synchronous low skew clock signals. The low skew clock signals are generated using a DCM, and distributed to the 16 TX macros using the global clock routing in the FPGA. The input and output ports on the 64 bit transmitter entity are described in table 5.5.

The `clkout` signals are not used, but are included for simulation purposes. The `syncout` frame signals are not used, and are left open on the top entity. The 512 bits of PRBS on the 64 bit transmitter input have to be distributed correctly to the 16 TX macros. A correct distribution of the 512 bits will result in a PRBS on the 40 Gb/s serial MUX/DeMUX module output. This PRBS will be identical to the PRBS generated by the LFSR in the FPGA. The VHDL code that specifies the bit distribution for the 16 TX macros is shown below.

Name	Direction	Description
clk	Input	Clock signal (80 MHz)
clkx4	Input	Clock signal (311 MHz)
clkx4not	Input	Inverted version of clkx4 (311 MHz)
datain	Input	512 bit data from the PRBS generator (512 x 78 Mb/s) synchronized to the clk signal
rst	Input	Reset signal
clkoe	Input	Clock output enable
dataoe	Input	Data output enable
dataout	Output	64 bit data output (64 x 622 Mb/s) synchronous to the clkx4 and clkx4not signals
clkout	Output	Clock signals from the 16 TX macros (311 MHz)
syncout	Output	Frame signals from the 16 TX macros (NOT USED)

Table 5.5: The table describes the inputs and outputs on the 64 bit transmitter. The text embraced by parenthesis concerns this design only.

```
-------------------------------------------------------
--VHDL CODE FOR DISTRIBUTING BITS TO THE 16 TX MACROS--
-------------------------------------------------------
gen_data_to_mux_001 : for i in 0 to 31 generate
  buffera(i)  <= datain((16*i)+0)  ; --Data for TX macro  1
  bufferb(i)  <= datain((16*i)+1)  ; --Data for TX macro  2
  bufferc(i)  <= datain((16*i)+2)  ; --Data for TX macro  3
  bufferd(i)  <= datain((16*i)+3)  ; --Data for TX macro  4
  buffere(i)  <= datain((16*i)+4)  ; --Data for TX macro  5
  bufferf(i)  <= datain((16*i)+5)  ; --Data for TX macro  6
  bufferg(i)  <= datain((16*i)+6)  ; --Data for TX macro  7
  bufferh(i)  <= datain((16*i)+7)  ; --Data for TX macro  8
  bufferi(i)  <= datain((16*i)+8)  ; --Data for TX macro  9
  bufferj(i)  <= datain((16*i)+9)  ; --Data for TX macro 10
  bufferk(i)  <= datain((16*i)+10) ; --Data for TX macro 11
  bufferl(i)  <= datain((16*i)+11) ; --Data for TX macro 12
  bufferm(i)  <= datain((16*i)+12) ; --Data for TX macro 13
  buffern(i)  <= datain((16*i)+13) ; --Data for TX macro 14
  buffero(i)  <= datain((16*i)+14) ; --Data for TX macro 15
  bufferp(i)  <= datain((16*i)+15) ; --Data for TX macro 16
end generate ;
```

Here `buffera(i)` to `bufferp(i)` are the 32 bit input data signals for the 16 TX macros, and `datain` is the 512 bit data input signal from the PRBS generator. The code specifies that TX macro 1 will get the bits 0, 16, 32, 48,.....etc and TX macro 2 will get the bits 1, 17, 33, 49.....etc. This bit distribution will result in a PRBS on the MUX/DeMUX module output

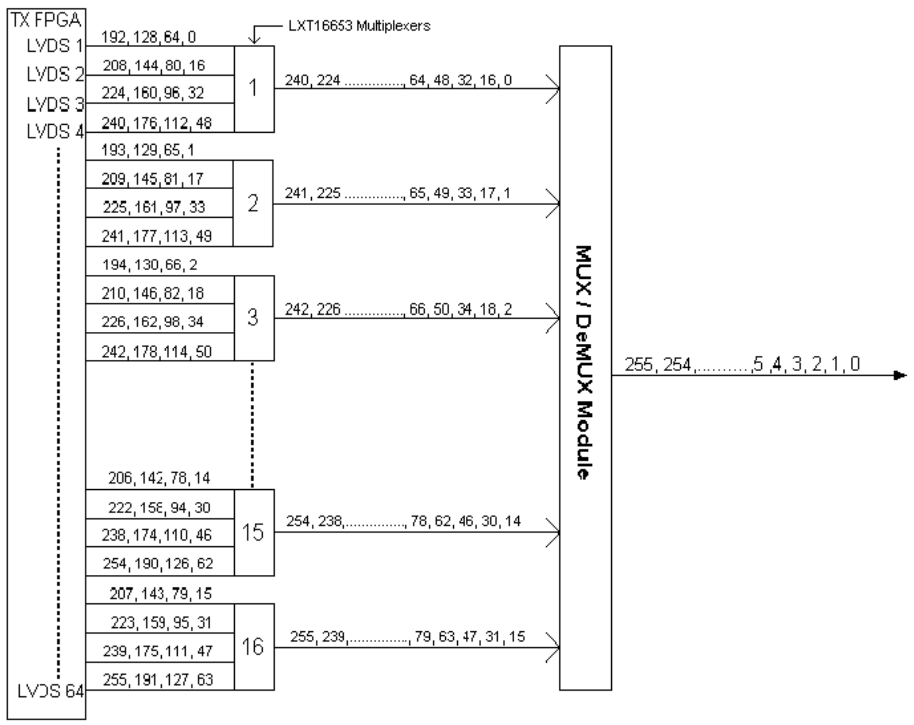


Figure 5.8: The figure shows the bit order on the 64 bit transmitter output, and the resulting bit order on the MUX/DeMUX module output.

as desired, and this PRBS will be identical to the PRBS generated by the LFSR in the TX FPGA. Figure 5.8 illustrates the bit distribution from the TX FPGA output to the MUX/DeMUX module output.

5.5 Top Entity for the RX FPGAs

Each RX FPGA has to accept 32 622 Mb/s LVDS signals together with 8 LVDS clock signals. The 32 LVDS data signals are divided into 8 groups of 4 signals, and each group of signals is represented by a clock signal. The 8 groups of signals will not be synchronous to each other. Thus each group of signals has to be received and demultiplexed using separate RX macros, which means that 8 RX macros will be needed in each RX FPGA. As the 8 groups of signals are not synchronous with respect to each other, it is not possible to use only one PRBS receiver to receive the incoming PRBS. Each group must be received by a separate PRBS receiver, thus a total of 8 PRBS receivers will be needed in each RX FPGA. Receiving the PRBS this way is possible, as each group of signals constitute identical PRBSs.

Each PRBS receiver has to register and count errors, and output this error count to a logical unit that can process the errors further, and ultimately output a total error count to a PC. As the RS-232 UART will not be implemented on the initial version of the test module, it will not be possible to output a total error count to a PC. Instead hardware signals will be used to indicate the amount of errors registered by each receiver. Based on the demands just described, a block diagram showing the top entity for the RX FPGA design and its architecture has been constructed, se figure 5.9. Figure 5.9 shows the entities constituting the architecture of the top

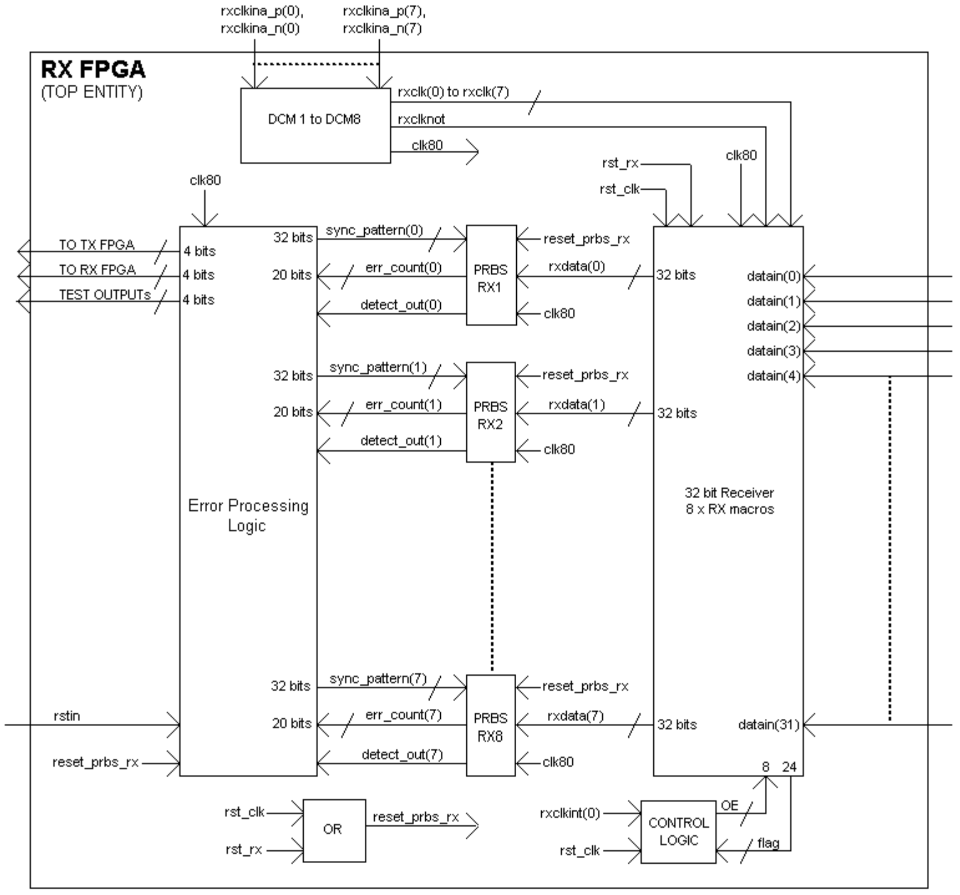


Figure 5.9: The figure gives a detailed view of the top level entity of the RX FPGA design and its architecture. The number of rxclknot signals used, depends on the composition of the RX macros used (one or two bufgs, refer to section 5.3).

entity, and the interface used between these entities. Table 5.6 describes the input and output ports of the top entity of the RX FPGA design. There are 4 reset signals shown in figure 5.9, namely `rst_clk`, `rst_rx`, `rstin`, and `reset_prbs_rx`. The latter is generated by ORing the `rst_rx` and `rst_clk` signals. The `rst_clk` signal is active when the TX DCM looses lock, and is connected to the FIFOs in the 8 RX macros, together with the control

Name	Direction	Description
rstin	Input	Reset push button
rxclkina_p, rxclkina_n	Input	311 MHz LVDS clock signals
dataina_p, dataina_n	Input	LVDS data signals, synchronous to rxclkina
clktx_locked	Input	Indicates that TX DCM is locked
clkrx_locked_out	Output	Indicates that RX DCMs are locked
test_output_1	Output	Test output for test purposes
test_output_2	Output	Test output for test purposes
test_output_3	Output	Test output for test purposes
test_output_4	Output	Test output for test purposes
test_output_5	Output	Test output for test purposes

Table 5.6: The table describes the inputs and outputs on the top entity of the RX FPGA design.

logic controlling the output enable signal for the FIFOs. Thus when the TX DCM looses lock, the FIFOs will be reset, and data will not be output from the FIFOs before the TX DCM has achieved lock again. The `rst_rx` signal is active when one or several DCMs in the RX FPGA looses lock, and is connected to the receiving circuitry in the RX macros. The `reset_prbs_rx` signal is active when either the TX DCM has lost lock, or when one or several of the RX DCMs has lost lock, and is connected to the 8 PRBS receivers and to the error processing logic. The `rstin` signal is active when the reset push button on the test module is activated, and is connected to the error processing logic. In this way error counts can be reset during a test if desired. Some signals for communicating with the TX FPGA and the other RX FPGA is shown in figure 5.9, but only two of these are utilized (`clktx_locked` and `clkrx_locked`). The signals are included to facilitate a RS-232 like interface between the FPGAs, so that the RX FPGAs can transmit error counts to the TX FPGA. This feature will not be implemented on the initial version of the test module, hence the signals are not included in table 5.6. In the following the entities contained in the block diagram in figure 5.9 will be described.

5.5.1 VHDL model for the 32 bit receiver

The 32 bit receiver is composed of the 8 RX macros used to demultiplex the incoming 32 622 Mb/s LVDS signals. The clock signals for the 8 RX macros are generated using 8 DCMs, as 8 clock domains will exist on the high frequency side of the RX macro. On the low frequency side of the RX macro, the data from all 8 macros will be output synchronous to the same clock signal. This clock signal is a 80 MHz clock signal generated by one of the 8 DCMs, thus only one clock domain exists on the low frequency side of the 8 RX macros. Table 5.7 describes the input and output port on the entity of the 32 bit receiver.

Name	Direction	Description
rxclk	Input	Eight 311 MHz clock signals from the DCMs
rxclknot_0	Input	One inverted 311 MHz clock signal from a DCM
rst_rx	Input	Reset signal for the receiving circuitry in the RX macros. Active when a RX DCM looses lock
clkin	Input	80 MHz clock for outputting data from the FIFO
rst_sys	Input	Reset signal for resetting FIFOs. Active when TX DCM looses lock
datain	Input	31 622 Mb/s LVDS signals synchronous to rxclk and rxclknot
oe	Input	FIFO output enable
Flag	Output	Eight 3bit flags indicating the status of the FIFOs
dataouta	Output	32 bit data synchronous to clkin, input to PRBS receiver 1.
dataoutb	Output	32 bit data synchronous to clkin, input to PRBS receiver 2.
dataoutc	Output	32 bit data synchronous to clkin, input to PRBS receiver 3.
dataoutd	Output	32 bit data synchronous to clkin, input to PRBS receiver 4.
dataoute	Output	32 bit data synchronous to clkin, input to PRBS receiver 5.
dataoutf	Output	32 bit data synchronous to clkin, input to PRBS receiver 6.
dataoutg	Output	32 bit data synchronous to clkin, input to PRBS receiver 7.
dataouth	Output	32 bit data synchronous to clkin, input to PRBS receiver 8.

Table 5.7: The table describes the inputs and outputs on the 32 bit receiver.

5.5.2 VHDL model for the PRBS receiver

The PRBS receiver has to receive 32 bits of data on every rising edge of a 80 MHz clock signal. These 32 bits have to be checked for bit errors, and the error count resulting from checking these 32 bits, has to be output to the error processing logic. Based on these demands, and on the principles discussed in section 2.2.4, a VHDL model for the PRBS receiver has been constructed. The VHDL model used to implement the PRBS receiver is illustrated in figure 5.10. The circuit shown in figure 5.10 processes the incoming PRBS for errors continuously, but does not output valid results before it has synchronized to the incoming data. Synchronization is done by searching the incoming PRBS for a unique 32 bit pattern (the synchronization pattern), and when this synchronization pattern appears, the receiver LFSR is loaded with the synchronization pattern. After the first synchronization has been achieved, the PRBS receiver will keep synchronizing every time the synchronization pattern appears on its input. This ensures that the receiver will re-synchronize if it should loose synchronization. The time it will take the receiver in figure 5.10 to synchronize (T_{sync}) is determined by the length of the PRBS, and the frequency used by the PRBS receiver. T_{sync} for a PRBS with a length of $2^N - 1$ bits is given by equation 5.1.

$$T_{sync} = (2^N - 1)\frac{1}{80MHz} \tag{5.1}$$

With $N = 31$ equation 5.1 yields 26 seconds, thus a maximum of 26 seconds will pass before the 16 PRBS receivers residing in the two RX FPGAs will

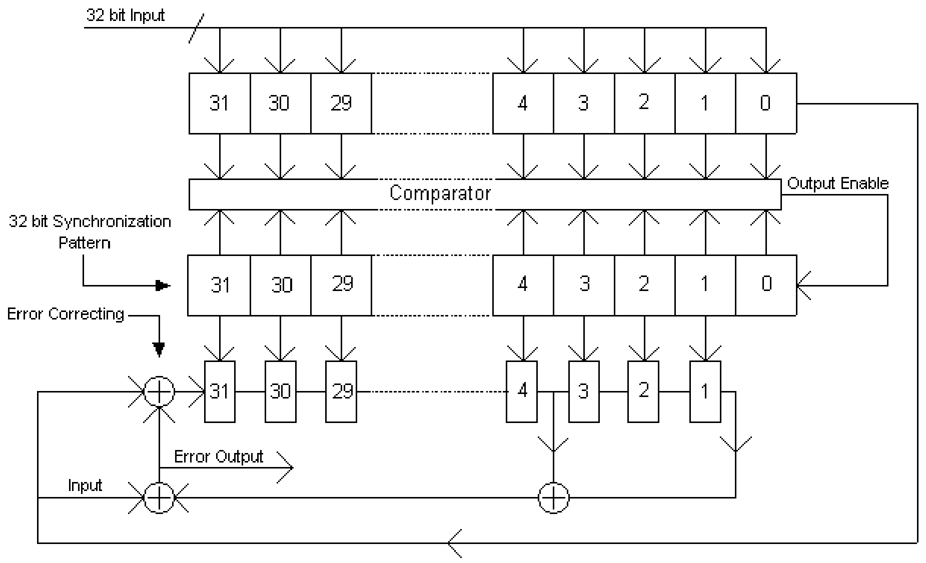


Figure 5.10: The figure illustrates the PRBS receiver. The comparator works by modulo-2 adding each bit in the incoming bit stream, with the corresponding bit in the synchronization pattern. The result from each of the 32 modulo-2 additions are summed up, and will yield 0 if the incoming bit stream matches the synchronization pattern, and 1 if not.

be synchronized. Other faster ways to synchronize the PRBS receiver exists [5], but the synchronization scheme just described is chosen as it is simple, and as no demands concerning the synchronization time exists. Table 5.8 describes the input and output ports on the entity of the PRBS receiver. On the final version of the test module, the error counts from the 16 PRBS

Name	**Direction**	**Description**
reset	Input	Reset signal that resets the LFSR and the error count
clk	Input	80 MHz clock
prbs_rx_data	Input	32 bit data synchronous to clk
sync_pattern	Input	32 bit synchronization pattern
err_out	Output	Error count (20 bit)
detect_out	Output	Toggles upon synchronization

Table 5.8: The table describes the input and output ports on the PRBS receiver entity

receivers, will be added in the TX FPGA. Thus the size of the register used to output the error count from a PRBS receiver, should correspond to the number of bits needed to represent the total amount of errors registered in the 40 Gb/s input data. The total amount of bit errors registered in the 40 Gb/s input data, will depend on the expected worst case BER, and on the time interval in which the errors are being counted. It is expected that the BER wont exceed 10^{-5} errors/bit, which yields 400000 errors/s at 40 Gb/s. If the total error count is output once every second, a 19 bit register will be sufficient to represent the error count (400000), and there will be more then enough time to transmit the 19 bits to a PC across a RS-232 interface, using for instance a bit rate of 9600 bits/s. The register used to output errors from the PRBS receiver (`err_out`) was chosen to have a size of 20 bits, thus a BER as high as 2.6×10^{-5} can be processed.

On the final version of the test module, a functionality indicating loss of synchronism should be included. If the PRBS receiver looses synchronism with the incoming bit stream, it will not register and count errors correctly. Synchronism can be lost if, for instance, a glitch in the clock signal on the MUX/DeMUX module occurs. Loss of synchronism could be triggered by exceeding a certain predetermined error rate threshold, and should be indicated to the user.

5.6 Simulation and test

In order to verify that the VHDL models work as intended, they have to be tested. VHDL models are tested using a simulation tool, and a test bench. The test bench simulates the surroundings to the design, and through the test bench it is possible to apply the appropriate combination of test stimuli to the design. Test stimuli define the input signals for the VHDL entities, and it is important to specify a sufficient amount of test stimuli to test the required functionality of the design. The VHDL models for the FPGA designs were simulated using Modelsim 5.5b.

Testing of the FPGA designs for the test module is simple, as the models composing these designs only have few functions that need to be verified. However, in order to construct a test bench it is necessary to specify which tests are to be conducted, and which test stimuli that will be needed to perform these tests.

First it has to be verified that the TX FPGA design functions as intended. The purpose of testing the TX FPGA design, is to verify that the PRBS generator and the 64 bit transmitter function as desired. Furthermore the functionality of the delay line configuration, and the error generator in the transmitter top entity, has to be verified. The PRBS generator and the 64 bit transmitter only requires the clock signals generated by the DCM in the transmitter FPGA to function. A few test stimuli have to be specified in order to test the error generating function of the transmitter, and the adjustment of the delay line configuration. When the functionality of the TX FPGA design is verified, testing of the RX FPGA design can begin. The purpose of testing the RX FPGA, is to verify that the 32 bit receiver functions as intended, and that the 8 PRBS receivers function correctly. The signals needed to test the RX FPGA design are the data signals generated by the TX FPGA, and eight 311 MHz clock signals.

5.6.1 Simulation of the TX FPGA design

In order to test the TX FPGA design, an external 311 MHz clock signal has to be generated and applied as input clock to the top entity of the TX FPGA design (refer to figure 5.6). Furthermore test stimuli for the `error_set_in` and `delay_set_in` inputs on the top entity have to be specified.

Using the external 311 MHz input clock signal the DCM will generate the necessary 80 MHz and 311 MHz clock signals for the PRBS generator and the 64 bit transmitter. After the external 311 MHz clock signal is applied, the functionality of the 64 bit transmitter and the PRBS generator can be verified by monitoring their input and output signals.

The PRBS generator is, for test purposes, configured to generate a PRBS with a length of $2^4 - 1$ bits. This sequence is only 15 bits long and easy to recognize on the simulation waveforms. Two periods of this PRBS is shown below

```
111101011001000111101011001000
```

Configuring the PRBS generator to transmit a PRBS with a period of $2^{31}-1$ bits instead, is simple and only requires changing the feedback tabs in the VHDL code for the PRBS generator. The functionality of the generator will be the same regardless of the feedback tabs used. Thus if the generator functions correctly when transmitting a PRBS with a period of $2^4 - 1$ bits, it will function correctly when transmitting a PRBS with a period of $2^{31} - 1$ bits as well.

The test stimuli used to test the error generating functionality and the delay line adjusting functionality is shown in table 5.9 and table 5.10. These stimuli are generated in the test bench. The TX FPGA design was sim-

Stimuli specification for the error_set_in signal				
`error_set_in(3)`	`error_set_in(2)`	`error_set_in(1)`	`error_set_in(0)`	Expected Function
0	0	0	0	No errors
0	0	0	1	Insertion of 1 error pr. channel
0	0	1	0	Insertion of 2 errors pr. channel
0	0	1	1	Insertion of 5 errors pr. channel
X	X	X	X	No errors

Table 5.9: Specification of test stimuli for the error_set_in signal

ulated using the specified signals and stimuli, and the resulting waveforms can be seen in appendix E.1 to appendix E.3. A table describing the signals shown in the waveforms is included in appendix E.

Stimuli specification for the delay_set_in signal		
`delay_set_in(1)`	`delay_set_in(0)`	`delayline_out(10 downto 0)`
0	0	00101000000
0	1	00100110000
1	0	00100100000
1	1	00100011100

Table 5.10: Specification of test stimuli for the delay_set_in signal

The waveforms in appendix E.1 show that the output from the PRBS generator (`tx_top/prbs/prbs_data`) is a $2^4 - 1$ PRBS as expected, and the PRBS generator output is synchronous to the 80 MHz `clk` signal as required. The 512 bit output from the PRBS generator is input to the 64 bit transmitter (`tx_top/tx0/datain`, se appendix E.2), and the 512 bits are distributed to the 16 TX macros as expected (`tx_top/tx0/buffera` to `buffero`). The correct distribution of the 512 bits can be verified by realizing that `buffera` to `bufferp` represent the way that the bits will appear on the 16 2.5 Gb/s data signals on the 40 Gb/s interface. The `tx_top/tx0/bufferp` signal is used as a trigger signal for test purposes, but would normally receive bits in the same way as `buffera` to `buffero` receives bits. The `tx_top/tx0/tx0/datain` and `dataout` signals represent the input and output of TX macro 0. As the waveforms illustrate, the 32 bit input signal is multiplexed into a four bit signal representing the same PRBS, but at 8 times the data rate as required. The `delay_int` and `delayline_int` signals in appendix E.3 show that the adjustment functionality of the delay line configuration functions as specified in table 5.10. The wave forms show the 64 bit transmitter input signal together with the `error_set_in` signal, and it can be seen that errors are inserted in the PRBS corresponding to the configuration of the `error_set_in` signal. The amount of errors inserted will be verified by the error counting functionality in the PRBS receiver.

5.6.2 Simulation of the RX FPGA design

The only signals required to test the RX FPGA design, are the data signals transmitted from the TX FPGA, and 8 311 MHz clock signals. The 8 311 MHz clock signals are needed to simulate the clock signals recovered by the demultiplexers, and for simulation purposes these 8 signals will be 8 copies of the clock signal transmitted from the TX FPGA.

In order to simulate the RX FPGA design, a test bench which connects the TX FPGA outputs to the inputs on the RX FPGAs was created. The multiplex and demultiplex stages located between the TX FPGA and the RX FPGAs in the physical implementation of the test module, are not included in the simulation. This is justified as the bits transmitted from the TX FPGA have been verified to be distributed as shown in figure 5.8. As the bits are transmitted correctly from the transmitter, they will be received correctly too, assuming that the demultiplexers on the test module function as intended.

The functionality of the RX FPGA design can be verified by monitoring the input and output signals from the 32 bit receiver and the 8 PRBS receivers. The output signals of interest from the PRBS receivers, are the synchronization signals and the error counts. It is expected that the synchronization signal will change polarity every time the PRBS receiver synchronizes, which will be once every 187th nanosecond for a PRBS with a length of 15 bits (se equation 5.1). The error count output is expected to correspond to the number of errors transmitted from the TX FPGA, and initially no errors will be transmitted in order for the PRBS receiver to synchronize.

The RX FPGA design was simulated using the signals described above, and the waveforms resulting from the simulation can be seen in appendix F.1 to appendix F.5. The designs for the two RX FPGAs are identical, thus only waveforms resulting from the simulation of one RX FPGA design is included. A table describing the signals shown on the waveforms is included in appendix F.

The waveforms in appendix F.1 show that the data outputs of the 32 bit receiver consist of 8 identical PRBSs (`rx_top_1/rxdata(7 downto 0)`) as expected, and that these PRBSs are identical to the PRBS generated in the TX FPGA. These 8 PRBSs are output synchronous to the `rx_top_1/clk80` signal as required. The synchronization signals from the 8 PRBS receivers (`detect_out(7 downto 0)`) change polarity as expected once every 187 ns. The `tx_top/error_set_in` signal is included in the waveforms to indicate when the transmitter starts to transmit errors. As the waveforms show the errors are counted correctly corresponding to the configuration of the `tx_top/error_set_in` signal (se table 5.9). When the TX FPGA starts to transmit errors, the PRBS receivers stop synchronizing to the incoming data. This is expected as a minimum of 1 error is inserted once every 32 bits, which means that the synchronization pattern will never appear (when using a PRBS with a length of 15 bits). The waveforms show that when the transmitter returns to error free transmission, the PRBS receivers start synchronizing again, as intended.

Also included in appendices F.4 and F.5 is waveforms verifying that the 8 DCMs in the RX FPGA design function as intended, and waveforms illustrating how the reset signals are controlled by the locked signals from the DCMs in the RX and TX FPGA designs. Appendix F.3 shows a closeup of the error counts, and as the waveforms show, the receivers are not synchronous (they do not show the same error count at all times). Furthermore the latency caused by the pipelining registers added by Synplify, can be seen as the delay between the edge of the `error_set_in` signal and the edge of the `err_count` signals. Appendix F.2 shows an other simulation of the

PRBS receivers, with no errors transmitted. As shown, the PRBS receivers synchronize as expected (`detect_out` toggles as expected once every 187th nanosecond), and do not register any errors (`err_count` signals "low" after synchronization).

5.7 Implementation

5.7.1 Synthesis

In order to convert the VHDL models into hardware, a synthesis tool is needed. The synthesis of the FPGA designs was done using Synplify Pro 7. Synplify can create hardware that corresponds to the resources available in the Xilinx FPGAs. In order to create the correct hardware, Synplify needs information regarding the FPGAs in which the designs are to be implemented. During the synthesis, the design can be optimized with respect to parameters such as speed and area, and these optimization parameters have to be specified before the synthesis starts. Only the necessary optimizations should be specified, as each optimization adds a considerable amount of time to the synthesis. The synthesis tool needs information about the default frequency used in the design, and furthermore it is possible to specify frequencies for several different clock signals within the design. If no clock frequency is specified for a clock signal, the tool will assume that the signals has the default frequency. Constraints regarding the placement and hardware implementation of the TX and RX macros are included in the VHDL code for the macros. Thus no additional constraints will be needed for synthesis of the macros.

The default frequency used for the synthesis of the FPGA designs was 80 MHz, and in order to make the PRBS receivers function at 80 MHz it was necessary to include the pipelining and retiming optimization options available in Synplify. These options allow Synplify to insert registers and thereby retime some of the functions requested by the receiver logic. This will increase the frequency at which the logic can function, but it will increase the latency of the logic as well. As the receiving circuit will process data continuously, an increase in latency will have no influence.

The synthesis of the FPGA designs was successful, and the resulting log files are included on the CD that came with this report. Unconnected logic is removed by Synplify, and as the synchronization signals and the clock output signals from the macros are not used, logic associated with these outputs is removed. This is indicated in the log files.

5.7.2 Place and Route

In order to place and route the hardware generated by Synplify on the FPGA floor plan, the Xilinx place and route tool has to be used. In order for the place and route tool to place the hardware correctly, some constraints are needed. The locations of the input and output pins used in the design have to be specified, and the frequency of the clock signals has to be specified. The delay between certain critical logic blocks has to be specified as well. The constraints needed for the TX and RX macros have been specified by Xilinx, and are provided together with the macros as an UCF file. Some additional constraints had to be added to these files in order to account for the entire VHDL design. The UCF files used for constraining the TX FPGA design and the RX FPGA designs can be seen in appendix G. The constraints are described by the comments in the UCF file, but some of the constraints need some explanation.

The `TNM` constraint is a basic grouping constraint, and it is used to specify timing groups. In the constraint file it is attached to the clock nets, and is used, together with the `period` constraint, to specify the clock frequency of the clocks used in the design. The groups created with the `TNM` constraint are used as reference for specifying the delay between the synchronous elements in the RX and TX macros. Further information regarding the `TNM` constraint can be found in the Xilinx constraint manual. In the RX FPGA designs, a total of 10 global clock buffers (BUFGs) have been used to distribute the eight 311 MHz clock signals, and the 80 MHz system clock signal. The eight 311 MHz clock signals are connected to the eight RX macros, and these RX macros are located on the same physical edge of the FPGA. There are 16 BUFGs available for distribution of the clock signals, and 8 of these 16 BUFGs are located in the middle of the top edge of the FPGA, and 8 are located in the middle of the bottom edge of the FPGA. The 8 BUFGs on the top and on the bottom edge of the FPGA, are divided into primary and secondary BUFGs. A primary BUFG on one edge faces a secondary BUFG on the opposite edge. When distributing clocks signals in the FPGA, the FPGA should be viewed as consisting of 4 quadrants. Each opposing set of primary/secondary BUFGs share connection to the 4 quadrants, thus they cannot both be connected to the same quadrant. The location selected for the BUFGs are based on the above rule, and the locations of the BUFGs are specified by the `loc` constraint in the constraint file. As it is the DCMs that generate the clock signals for the BUFGs, they have to be placed close to the BUFGs they use. The location of the BUFGs and the DCMs can be determined by viewing the floor plan of the FPGA. Further information about usage of the global clock resources in the Virtex II FPGAs can be found in [12].

Using the constraints included in appendix G, the FPGA designs were successfully placed and routed on the FPGA floor plans. The floor plans for the FPGA designs can be seen in appendix G. The placement of the macros, and their respective I/Os can be viewed on the floor plans. The macros are placed along the edge of the FPGA, and as show, the pins associated with each macro is placed within $\pm$ 2 CLBs distance from the macro.

The Xilinx place and route tool provides precise timing information for the FPGA designs, and using static timing analysis it is possible to verify whether the designs meet all the specified timing constraints. Through static timing analysis it was verified that the FPGA designs met all the specified constraints. The timing reports are included on the CD that came with this report.

In order to verify the functionality of the generated hardware, the floor plan file could be back annotated and simulated. This is done by converting the floor plan file into a VHDL net list together with a SDF file that contains the timing information for the design. Using these two files, a simulation of the hardware routed on the FPGA can be conducted. However, as the FPGA designs are simple and only have few functions, the static timing analysis is considered sufficient to verify that the designs functions at the required frequencies. Furthermore, as the FPGAs are reconfigurable it is possible to correct any errors that should occur in the placement of the hardware, or the synthesis.

5.8 Summary

Based on the requirements for the FPGA designs, a functional block diagram for the TX FPGA design and the RX FPGA design has been constructed. VHDL models for each of the functions specified in the block diagrams have been made. VHDL macros supplied by Xilinx have been used in order to facilitate transmission and reception of data at a rate of 622 Mb/s to and from the FPGAs. The functionality of the TX FPGA design and the RX FPGA design has been verified using simulation. Using Synplify, the VHDL designs have been successfully converted into hardware. This hardware has been placed and routed on the FPGA floor plan, and the constraints needed to place and route the hardware have been described. Using static timing analysis, it has been verified that the FPGA designs function at the required frequencies, and meet all the specified constraints.

Chapter 6

Test module test

In order to test the PCB efficiently, a test plan is needed. The test plan functions as a guide through the tests, and have to contain all the necessary information needed to conduct the tests. A test plan should be divided into a number of test cases, where each test case describes one test. The minimum amount of information that each test case should contain is listed below.

- Case Title
- Case ID
- Equipment and Instruments
- Purpose
- Requirements
- Set-Up and Procedure
- Measurements
- Results

The case *title* gives a general description of the test, and the *case ID* is a unique number that identifies the test case. The *equipment and instruments* needed to perform the test have to be specified, and if possible the ID number (if any) identifying the equipment or the instruments, should be specified too. The *purpose* of the test case describes what the aim of the test is, and the *requirements* specify the requirements for the results in terms of signal levels and timing relations. The *set-up and procedure* contains details relevant to the test setup needed to perform the test. The *measurements* describes how the measurements should be performed, and which test pads or component terminals to use in order to measure the desired signal(s). The *results* contain the test results.

The order in which the tests should be conducted must be specified. The order in which the tests are performed is determined by the dependency between the components in the design. All active components depend on the power supply, so the power supply is usually the first functionality to be tested. In synchronous logic designs, most components depend on a clock signal. Thus after having tested the power supply, the clock distribution should be tested. The origin of the clock distribution test should be the source of the clock, which can be one or several external signal(s), or an oscillator that is part of the design. Once the clock source is identified, the test of the clock distribution should be performed corresponding to the path of the clock signal(s). When the clock distribution has been tested, the data processing components in the design can be tested. The order in which the data processing components are tested, is determined using the same principles as when determining the order for testing the clock distributing components. The tests that have been conducted on the test module are specified in table 6.1. The order in which they appear in the table, represent the order in which they were performed. The general test setup used to test

CASE ID	**Case Title**
001	5 Volt supply test
002	3.3 Volt supply test
003	1.5 Volt supply test
004	622.08 MHz reference clock test
005	FPGA TX clock test
006	MUX clock test
007	DMUX Reference clock test
008	LVDS I/O test (TX FPGA)
009	MUX Clock/Data alignment
010	CML clock and data phases
011	LVDS data and clock (DMUX output)
012	Loop back test

Table 6.1: The table specifies the tests that have been conducted on the test module. The order in which the tests appear in the table, is identical to the order in which they were conducted.

the test module can be seen in figure 6.1. The test plan used to test the test module can be seen in appendix H.

6.1 Power Supply Test

The power supply is tested to make sure that all supplies function correctly before powering up the entire test module for the first time. This is important as components using multiple supplies, can take damage if only some of

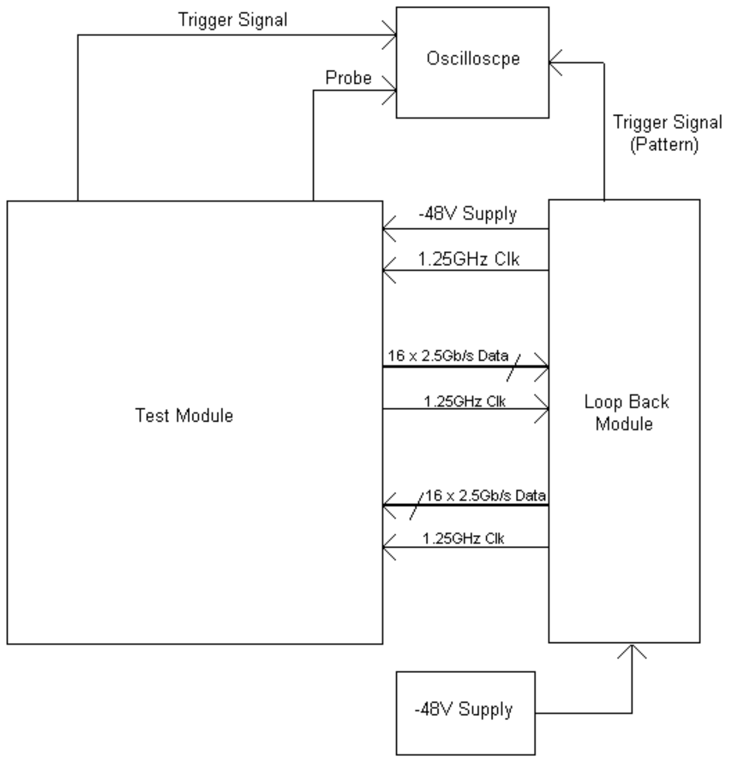
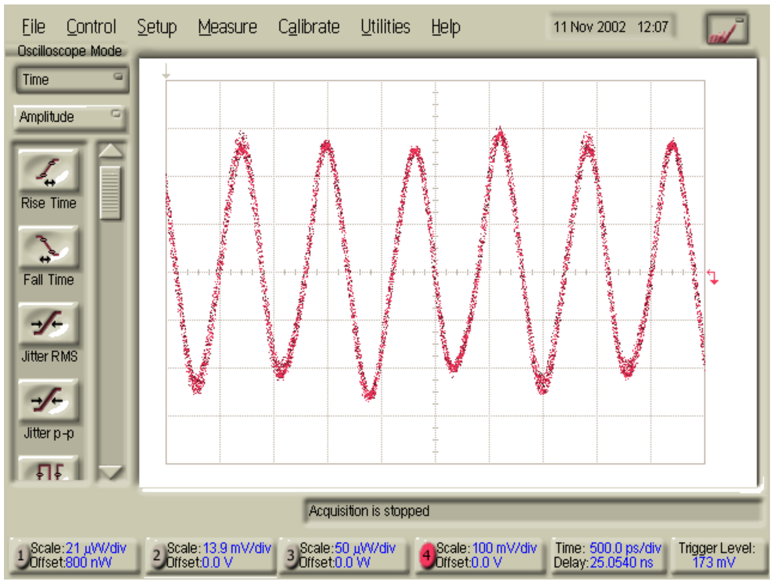


Figure 6.1: General test setup used for testing the test module

the required supplies are turned on. Thus before powering up the entire test module, all fuses connecting the power supply to the rest of the test module were removed. Using high power (25 watt) 1 Ω resistors, all the power supply voltages were loaded to their maximum current while monitoring their output voltage. In order to load the supplies to their specified current, several 1 Ω resistors had to be placed in parallel. All supply voltages stayed within their specified range at all loads.

6.2 Clock Distribution Test

The 622 MHz reference clock oscillator on the test module is not needed for testing the test module, and testing of this oscillator is not included in the initial tests of the test module. The clock signal that the test module uses to generate and transmit data, is generated and transmitted from a 1.25 GHz oscillator on the loop back module. This clock signal is input to a clock divider on the test module. The 1.25 GHz clock signal was measured at the input of the clock divider on the test module, and the waveform resulting from this measurement can be seen in figure 6.2. As expected the clock

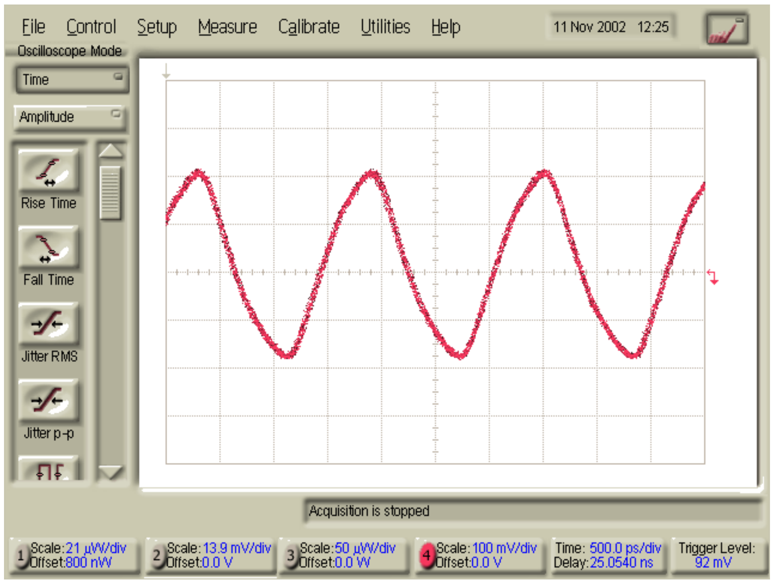


Acquisition is stopped

1 Scale: 21 µW/div Offset: 800 nW
2 Scale: 13.9 mV/div Offset: 0.0 V
3 Scale: 50 µW/div Offset: 0.0 W
4 Scale: 100 mV/div Offset: 0.0 V
Time: 500.0 ps/div Delay: 25.0540 ns
Trigger Level: 173 mV

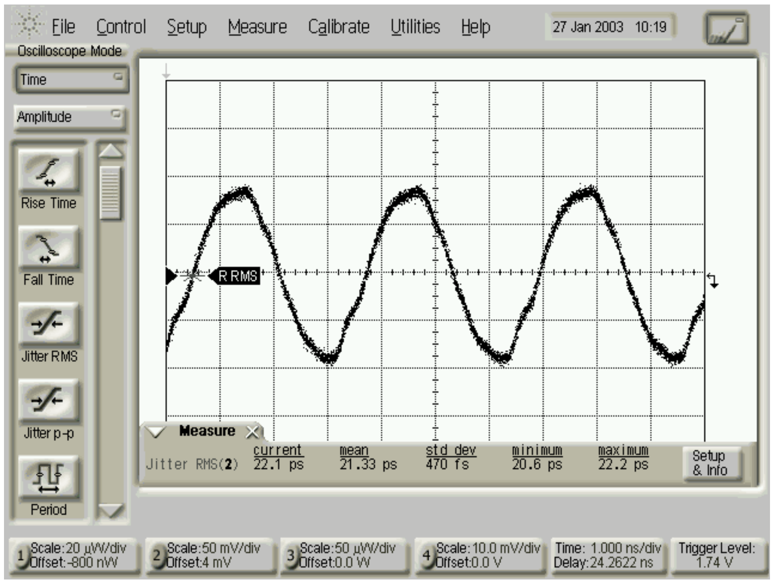
Figure 6.2: 1.25 GHz reference clock signal generated by an oscillator on the loop back module. The waveform shows the 1.25 GHz clock signal at the input of a clock divider (U0045, refer to the schematic in appendix A) on the test module. The oscilloscope settings are 100 mV/div. and 500 ps/div.

signal has a frequency of 1.25 GHz, and the voltage swing is approximately 500 mV single ended as expected (LVPECL VCO on loop back module). The clock divider (U0045) which the 1.25 GHz clock signal interfaces, is expected to divide the clock frequency by two. The output from the clock divider is input to an other clock divider (U0041) which has four outputs. On two of the outputs it provides a clock signal with a frequency equal to the frequency of the input clock signal. On the two other outputs it provides a clock signal with a frequency equal to half the frequency of the input clock signal. Thus this clock divider is expected to generate two 311 MHz clock signals and two 622 MHz signals. One of the 622 MHz clock signals is used as trigger signal for the oscilloscope, and the two 311 MHz signals are used as input clock to the TX FPGA and the PLL clock synthesizer. The 622 MHz trigger signal can be seen in figure 6.3. The frequency of the trigger signal is 622 MHz as expected, and the signal swing is approximately 400 mV single ended. The clock input to the TX FPGA and the PLL clock synthesizer can be seen in figure 6.4 and figure 6.5 respectively. As figure 6.4 shows, the input clock signal for the TX FPGA has a frequency of 311 MHz, and an amplitude of approximately 190 mV single ended (LVDS). As figure 6.5 shows, the input clock signal for the PLL clock synthesizer has a frequency of 311 MHz as intended, and an amplitude of approximately 780 mV single



```
11 Nov 2002  12:25
Acquisition is stopped
1 Scale:21 µW/div   Offset:800 nW
2 Scale:13.9 mV/div Offset:0.0 V
3 Scale:50 µW/div   Offset:0.0 W
4 Scale:100 mV/div  Offset:0.0 V
Time: 500.0 ps/div  Delay:25.0540 ns
Trigger Level: 92 mV
```

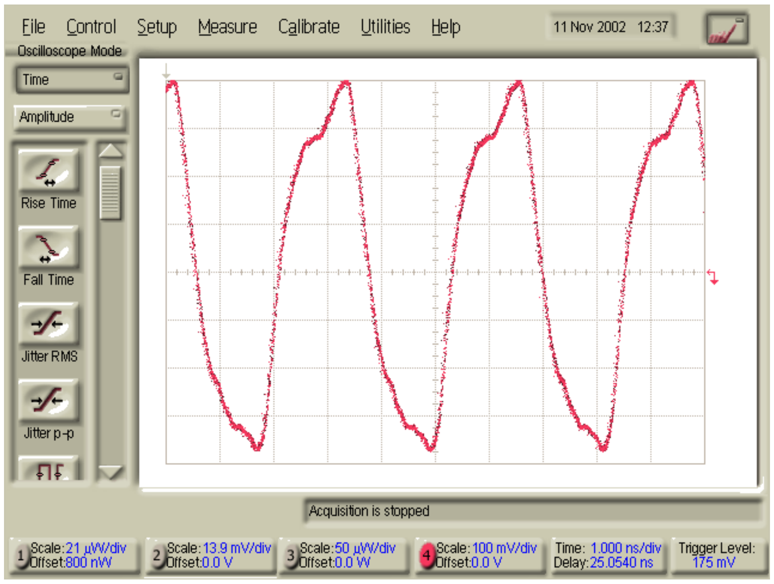
Figure 6.3: 622 MHz clock signal used as trigger signal for the oscilloscope. This clock signal is output from a clock divider (U0041), and makes out one of four outputs from this clock divider. The oscilloscope setting are 100 mV/div and 500 ps/div



```
27 Jan 2003  10:19
               current   mean      std dev   minimum   maximum
Jitter RMS(2)  22.1 ps   21.33 ps  470 fs    20.6 ps   22.2 ps
1 Scale:20 µW/div   Offset:-800 nW
2 Scale:50 mV/div   Offset:4 mV
3 Scale:50 µW/div   Offset:0.0 W
4 Scale:10.0 mV/div Offset:0.0 V
Time: 1.000 ns/div  Delay:24.2622 ns
Trigger Level: 1.74 V
```

Figure 6.4: TX FPGA input clock signal. The oscilloscope settings are 50 mV/div and 1 ns/div.

ended (LVPECL). The amplitude of the trigger signal is a bit low compared to the amplitude of the input clock signal for the PLL clock synthesizer. The



Acquisition is stopped

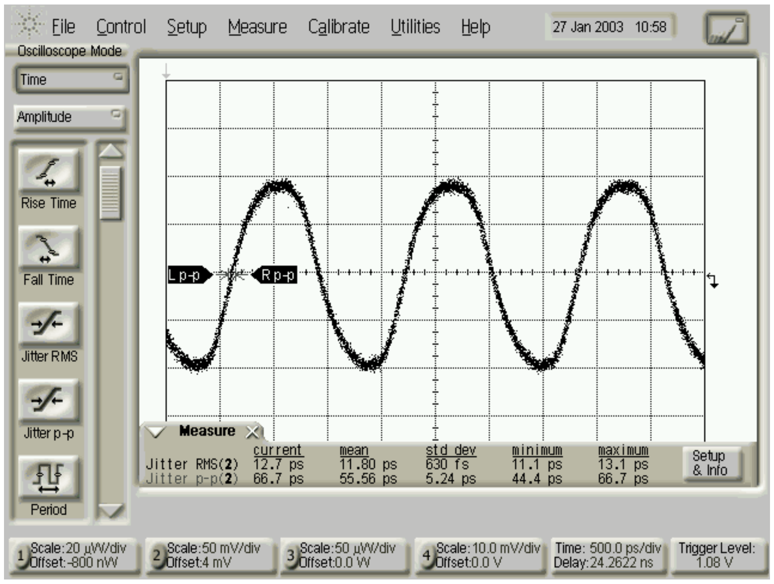
1 Scale: 21 µW/div Offset: 800 nW
2 Scale: 13.9 mV/div Offset: 0.0 V
3 Scale: 50 µW/div Offset: 0.0 W
4 Scale: 100 mV/div Offset: 0.0 V
Time: 1.000 ns/div Delay: 25.0540 ns
Trigger Level: 175 mV

Figure 6.5: 311 MHz input clock for the PLL clock synthesizer. The oscilloscope settings are 100 mV/div and 1 ns/div

two signals originate from the same clock divider, but the frequency of the trigger signal is twice as high as the frequency of the input clock signal for the PLL clock synthesizer. Furthermore the trigger signal is loaded by a 50 Ω coaxial cable (used to connect the trigger signal to the oscilloscope) which will load the clock signal capacitively, and thus increase the rise time of the signal. The higher frequency of the trigger signal combined the capacitive load caused by the coaxial cable, results in a lower amplitude.

The 622 MHz output clock signal from the PLL clock synthesizer is transmitted to a 1:20 clock buffer, that distributes the clock signals for the 17 multiplexers in the MUX stage. The 622 MHz clock signals on the input of the multiplexers were measured, and the phases of these clock signals were compared. In order to compare the phases of these clock signals, the measurements were conducted using the same oscilloscope probe and the same trigger level. In this way the delay through the probe and oscilloscope will be the same for all measurements. The maximum skew between any two of the input clock signals for the multiplexers, was observed to be 40 ps. This number includes differences in PCB traces and the output to output skew on the 1:20 clock buffer. As the output to output skew for the 1:20 clock buffer is 20 ps, a total of approximately 20 ps (40 ps - 20 ps) skew has been added to the clock signals by differences in the lengths of the PCB traces. This is 5 ps more than specified in section 4.6, but considering the uncertainties in the measurements and the jitter on the clock signals, the measured skew is considered acceptable. Figure 6.6 shows one of the

622 MHz clock signals on the input of one of the multiplexers. Figure 6.6



```
                current    mean       std dev    minimum    maximum
Jitter RMS(2)   12.7 ps    11.80 ps   630 fs     11.1 ps    13.1 ps
Jitter p-p(2)   66.7 ps    55.56 ps   5.24 ps    44.4 ps    66.7 ps
```

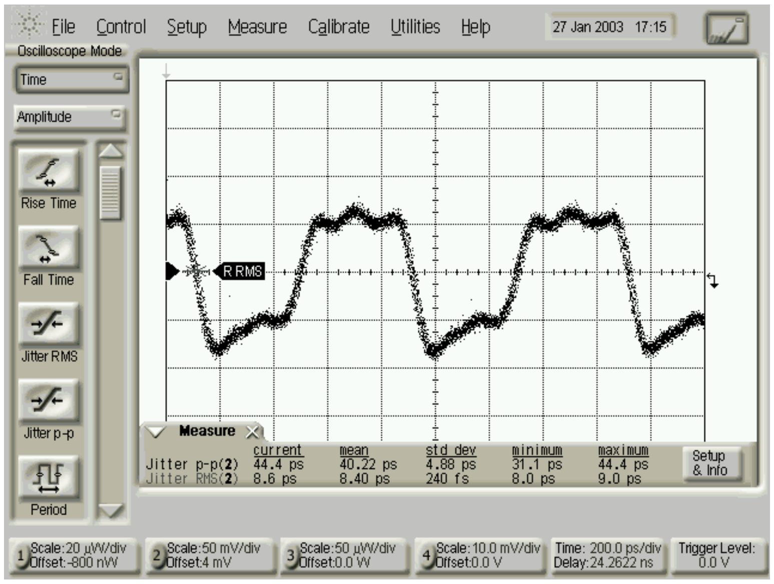
```
1 Scale:20 µW/div    2 Scale:50 mV/div   3 Scale:50 µW/div   4 Scale:10.0 mV/div   Time: 500.0 ps/div    Trigger Level:
  Offset:-800 nW       Offset:4 mV         Offset:0.0 W        Offset:0.0 V          Delay:24.2622 ns      1.08 V
```

Figure 6.6: 622 MHz clock signal at the input of one of the multiplexers in the MUX stage

shows that the input clock signal for the multiplexer has a frequency of 622 MHz as expected, and an amplitude of approximately 200 mV single ended (LVDS). The RMS and peak to peak jitter is indicated in figure 6.6, and has a maximum amplitude of approximately 67 ps (peak to peak). This figure should be seen as a rough indication of the jitter level present on the clock signal. The oscilloscope and the test setup will add jitter to the measurement, and the trigger signal which is input to the oscilloscope has some level of jitter present on it as well. Thus the amount of jitter shown on the measured signals should be viewed as worst case values.

The 1.25 GHz CML clock signal which is demanded by the 40 Gb/s interface, is generated by the 17th multiplexer in the MUX stage. The signal was measured and the resulting waveform can be seen in figure 6.7. As figure 6.7 shows, the frequency of the CML clock signal is 1.25 GHz as intended, and the single ended signal swing is 225 mV. The maximum peak to peak jitter on the 1.25 GHz clock signal is measured to 44.4 ps, but as stated earlier this figure is properly a great deal higher than the actual jitter level. The measured jitter level is below the theoretical maximum allowable limit of 50 ps (see section 4.6). As the actual jitter level is expected to be lower than the jitter level shown in figure 6.7, the measured clock signal is considered acceptable.

The last clock signal remaining (not counting the clock signals recovered by the demultiplexers) is the 39 MHz reference clock signal for the demulti-



```
                current   mean      std dev   minimum   maximum
Jitter p-p(2)   44.4 ps   40.22 ps  4.88 ps   31.1 ps   44.4 ps
Jitter RMS(2)   8.6 ps    8.40 ps   240 fs    8.0 ps    9.0 ps
```

```
1 Scale:20 µW/div    2 Scale:50 mV/div   3 Scale:50 µW/div   4 Scale:10.0 mV/div   Time: 200.0 ps/div   Trigger Level:
  Offset:-800 nW       Offset:4 mV         Offset:0.0 W        Offset:0.0 V          Delay:24.2622 ns     0.0 V
```

Figure 6.7: 1.25 GHz clock signal generated by the 17th multiplexer in the MUX stage. The oscilloscope settings are 50 mV/div. and 200 ps/div.

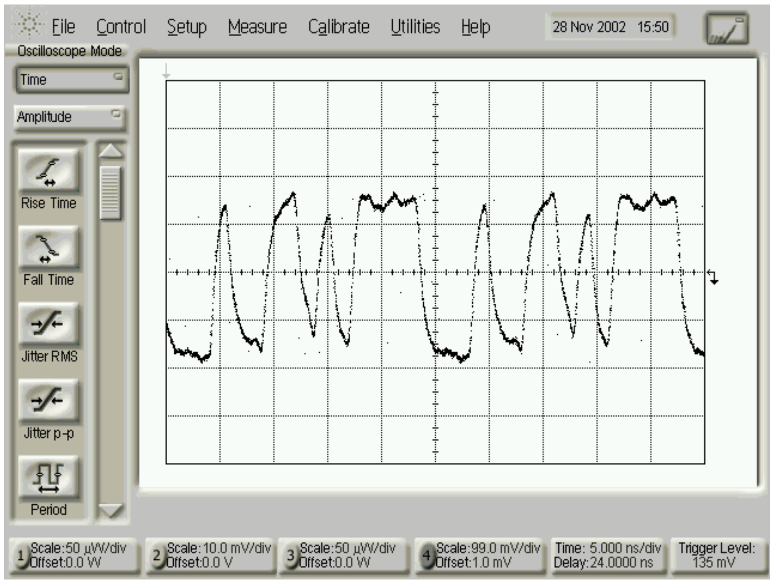
plexers. This signal is not critical timing wise, and the measurement showed that the frequency of the clock signal is 39 MHz as expected, and that the amplitude is approximately 800 mV single ended (LVPECL).

6.3 Data transmission and acquisition test

6.3.1 Data transmission test

The data generation starts in the TX FPGA, thus the first data signals to be measured are the LVDS data signals transmitted from the TX FPGA to the MUX stage. The timing of the 64 622 Mb/s data signals transmitted from the TX FPGA is critical, and according to section 3.2.3, the skew between any two LVDS outputs must not exceed 400 ps. Using the same oscilloscope probe, and the same trigger level, the phase of 16 out of the 64 LVDS data signals was measured at the input of the multiplexers and compared (one signal at each multiplexer). The signal transmitted from the FPGA for this measurement was a PRBS with a length of $2^4 - 1$. As the trigger signal is a 622 MHz clock signal, the oscilloscope image is an eye diagram, and the phase of the rising/falling edges can be compared (it is known that the skew between the data signals wont exceed 1 bit period). A maximum skew of 200 ps between any two of the 16 LVDS signals was measured, which is well within the acceptable limit of 400 ps. Using a pattern trigger signal transmitted from one of the 16 TX macros, it is possible to display the

PRBS pattern on the oscilloscope. The pattern trigger signal is simply a signal that delivers a rising edge once every 15th bit (corresponding to the 15 bit period of the PRBS being transmitted). The waveform resulting from the measurement can be seen in figure 6.8. The PRBS shown in figure 6.8



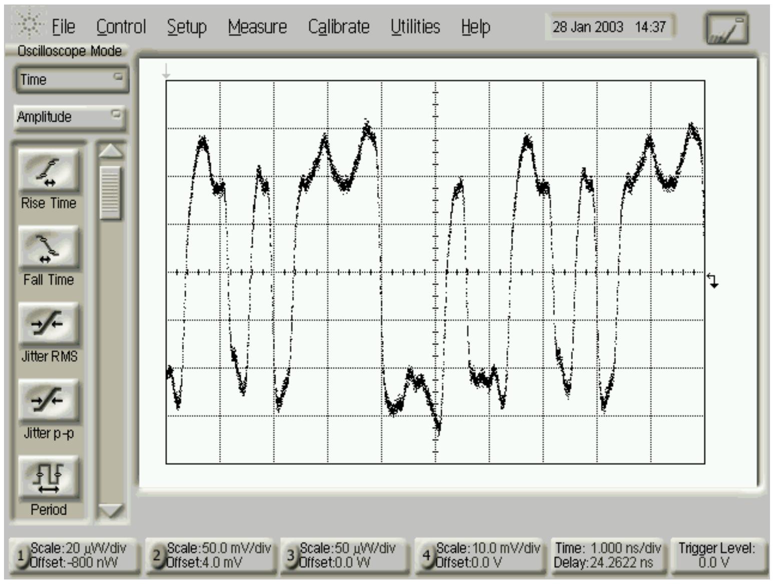
28 Nov 2002 15:50

1 Scale:50 µW/div Offset:0.0 W
2 Scale:10.0 mV/div Offset:0.0 V
3 Scale:50 µW/div Offset:0.0 W
4 Scale:99.0 mV/div Offset:1.0 mV
Time: 5.000 ns/div Delay:24.0000 ns
Trigger Level: 135 mV

Figure 6.8: 622 Mb/s PRBS at the input of one of the multiplexers in the MUX stage. The oscilloscope setting are 99 mV/div. and 5 ns/div.

was observed at the input of the 16 multiplexers in the MUX stage, which verifies that the PRBS generator in the TX FPGA functions correctly. As Figure 6.8 shows the data rate at which the PRBS is transmitted is 622 Mb/s as expected, and the amplitude of the signal is approximately 300 mV single ended. Figure 6.9 shows the PRBS on the output of one of the multiplexers in the MUX stage. Figure 6.9 shows that the PRBS is transmitted at 2.5 Gb/s as expected, and that the amplitude of the signal is approximately 250 mV single ended.

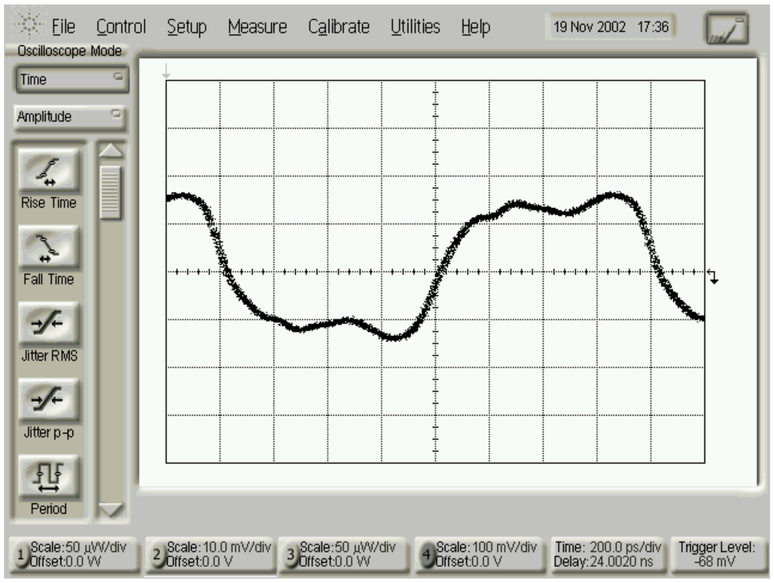
The timing of the 17 2.5 Gb/s signals on the output of the multiplexers in the MUX stage is critical. The device to device skew for the multiplexers is 50 ps, and the maximum skew between the clock signals distributed to the multiplexers was observed to be 40 ps. Thus a maximum of 90 ps ($50ps + 40ps$) skew between any two of the 16 data signals on the MUX stage output is expected. Using the same oscilloscope probe and the same trigger level, the phases of the 2.5 Gb/s signals on the MUX stage output were measured and compared. In order to ease the phase comparison, the same bit pattern was transmitted on all 16 channels. The transmitted bit pattern was `00110011001100110011.....0011`, and this bit pattern results in the waveforms shown in figure 6.10. A maximum of approximately 70



28 Jan 2003 14:37

1 Scale:20 µW/div Offset:-800 nW
2 Scale:50.0 mV/div Offset:4.0 mV
3 Scale:50 µW/div Offset:0.0 W
4 Scale:10.0 mV/div Offset:0.0 V
Time: 1.000 ns/div Delay:24.2622 ns
Trigger Level: 0.0 V

Figure 6.9: 2.5 Gb/s PRBS at the output of one of the multiplexers in the MUX stage. The oscilloscope setting are 50 mV/div. and 1 ns/div.



19 Nov 2002 17:36

1 Scale:50 µW/div Offset:0.0 W
2 Scale:10.0 mV/div Offset:0.0 V
3 Scale:50 µW/div Offset:0.0 W
4 Scale:100 mV/div Offset:0.0 V
Time: 200.0 ps/div Delay:24.0020 ns
Trigger Level: -68 mV

Figure 6.10: 2.5 Gb/s output data from one of the 17 multiplexers in the MUX stage. During the measurement of the phases of the multiplexer outputs, all multiplexers were transmitting the same bit pattern. The oscilloscope settings in the figure are 100 mV/div. and 200 ps/div.

ps skew was oberseved between any two of the 17 2.5 Gb/s output signals from the MUX stage. This is 20 ps better than expected. Furthermore it is possible to reduce this skew even more if necessary by using the loop filter adjusment circuits. The measured skew is considered acceptable as it is below the expected skew, and combined with the measured jitter on the 1.25 GHz CML clock signal, it results in a valid data window of

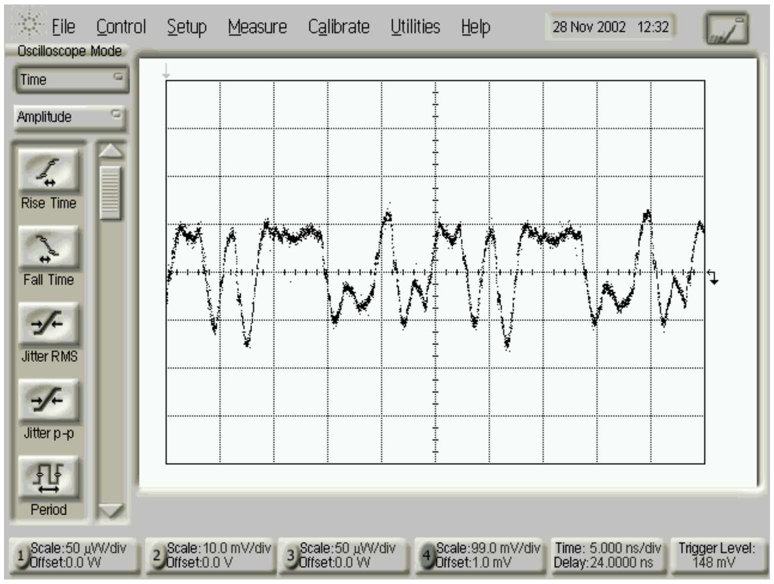
$$T_{valid} = T_{2.5Gb/s} - T_{skew} - t_{rf} - T_{jitter} = 400ps - 70ps - 100ps - 44ps = 186ps \tag{6.1}$$

where T_{valid} is the valid data window, $T_{2.6Gb/s}$ is the bit period of the 2.5 Gb/s data signals, t_{rf} is the rise/fall time for the data signals, and T_{jitter} is the jitter on the 1.25 GHz CML clock signal. T_{valid} is 36 ps larger than the 150 ps demanded by the setup and hold time requirement for the FIFO on the MUX/DeMUX module, and this allows for some difference in trace length between the 34 PCB traces connecting the MUX stage to the back plane. The actual valid data window is expected to be larger, as T_{jitter} is expected to be lower than measured. After having verified the timing of the 17 data signal on the MUX stage output, the TX FPGA was configured to transmit a PRBS with a length of $2^4 - 1$.

6.3.2 Data acquisition test

The 16 2.5 Gb/s data signals transmitted from the multiplexers in the MUX stage, are connected to the demultiplexers in the DMUX stage through the loop back module. The 2.5 Gb/s data signals interfacing the DMUX stage was measured at the input of the demultiplexers. The resulting waveforms were similar to the waveform shown in figure 6.9.

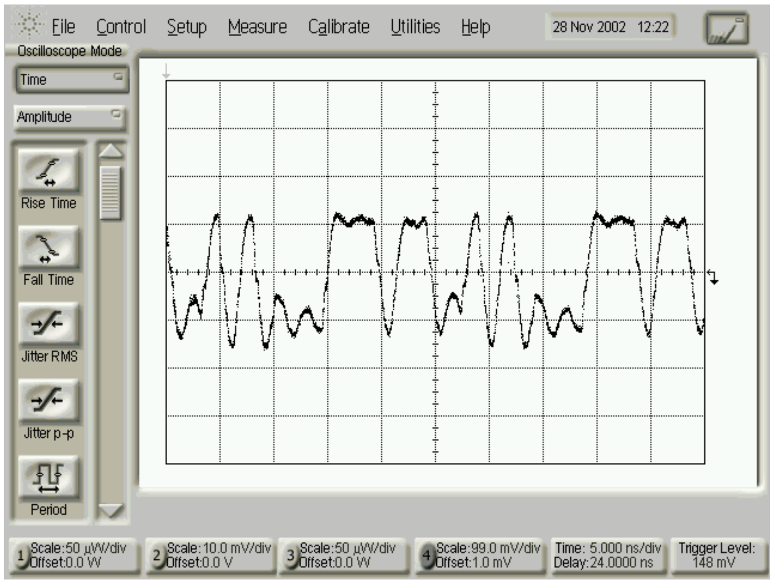
Each demultiplexer recovers one 622 MHz clock signal and four 622 Mb/s data signals. Using the same oscilloscope probe and the same trigger level, the phases of the data signals transmitted from each demultiplexer were measured and compared. A maximum skew of 200 ps was observed between any two of the 4 data signals transmitted from each demultiplexer. A 622 Mb/s PRBS at the input of one of the RX FPGAs can be seen in figure 6.11. As figure 6.11 shows, the data rate of the signal is 622 Mb/s as expected, and the amplitude of the signal is approximately 200 mV single ended (LVDS). Furthermore figure 6.11 shows that some oscillations occur on the signal. The signal showed in figure 6.11 is the positive input of the differential signal transmitted to the RX FPGA. The corresponding negative input of the same differential signal can be seen in figure 6.12. Figure 6.12 shows that the oscillations exist on the corresponding negative input of the differential signal as well. These oscillations are believed to be cross talk from neighbor signals, as the oscillations appear to have the same frequency as the data signal. The maximum amplitude of the oscillation is 100 mV peak to peak, but as the differential signal has an amplitude of 400 mV, the oscillation should not have any influence on the receiver decision circuitry.



28 Nov 2002 12:32

1 Scale:50 µW/div Offset:0.0 W
2 Scale:10.0 mV/div Offset:0.0 V
3 Scale:50 µW/div Offset:0.0 W
4 Scale:99.0 mV/div Offset:1.0 mV
Time: 5.000 ns/div Delay:24.0000 ns
Trigger Level: 148 mV

Figure 6.11: 622 Mb/s PRBS measured at the true input of one of the RX FPGAs. The signal was measured using a pattern trigger signal. The signal is measured through a via under the FPGA. The oscilloscope settings are 99 mV/div and 5 ns/div.

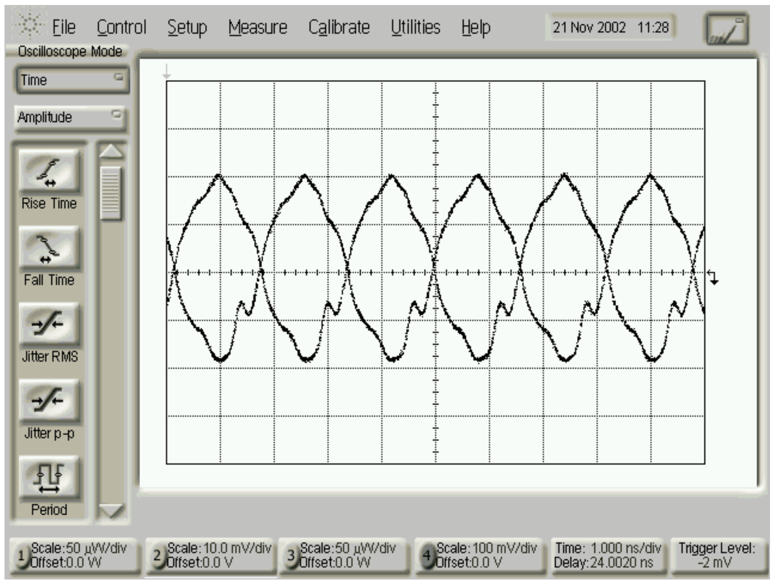


28 Nov 2002 12:22

1 Scale:50 µW/div Offset:0.0 W
2 Scale:10.0 mV/div Offset:0.0 V
3 Scale:50 µW/div Offset:0.0 W
4 Scale:99.0 mV/div Offset:1.0 mV
Time: 5.000 ns/div Delay:24.0000 ns
Trigger Level: 148 mV

Figure 6.12: 622 Mb/s PRBS measured at the inverted input of one of the RX FPGAs. The signal was measured using a pattern trigger signal. The signal is measured through a via under the FPGA. The oscilloscope settings are 99 mV/div and 5 ns/div.

The 622 MHz clock signal transmitted from each demultiplexer is divided by 2 before it is transmitted to the FPGA. The 311 MHz clock signal transmitted from the clock divider to the FPGA was measured at the input of the FPGA. The resulting waveform can be seen in figure 6.13. The waveforms in figure 6.13 appear as an eye diagram, because the 311 MHz clock signal is measured using a 622 MHz trigger signal. The reflections appearing



1 Scale:50 µW/div Offset:0.0 W
2 Scale:10.0 mV/div Offset:0.0 V
3 Scale:50 µW/div Offset:0.0 W
4 Scale:100 mV/div Offset:0.0 V
Time: 1.000 ns/div Delay:24.0020 ns
Trigger Level: -2 mV

Figure 6.13: 311 MHz clock signal at the input of one of the RX FPGAs. The reflection appearing on the signal edge, occurs because the termination resistors terminating the clock signal are placed several centimeters away from the input on the RX FPGA. The oscilloscope settings in the figure are 100 mV/div. and 1 ns/div.

on the edges of the clock signal in figure 6.13, occur because the termination resistor used to terminate the clock signal, is placed at the output of the clock divider (the transmitter). The termination resistor should have been placed as close to the input of the FPGA (the receiver) as possible, but this was not physically possible. The reflection occurs early on the rising edge of the signal, and should not influence the receiver decision circuitry. Figure 6.13 shows that the frequency of the clock signal is 311 MHz as expected, and that the amplitude of the signal is 400 mV single ended.

Having verified that the clock and data signals are recovered and transmitted correctly from the demultiplexers to the RX FPGAs, the PRBS receivers in the RX FPGA can be tested. In order to test the PRBS receivers, the synchronization signals from the PRBS receivers were output on the test pins on the FPGA. Furthermore the timing of the clock and data signals on the input of the FPGA had to be measured, so that the DCMs in the FPGA could be configured to delay the clock signal by the appropriate amount of time, in order to sample the data correctly. First, a PRBS with a period of 15 bits was transmitted from the TX FPGA, thus the synchronization signals are expected to toggle once every 187 ns (se equation 5.1). After having adjusted the delay in the DCMs several times, it was verified that 15 out of the 16 (one channel used for trigger signal) PRBS receivers synchronized to the incoming PRBS. Next, a PRBS with a period of $2^{31} - 1$ bits was transmitted from the TX FPGA, and it was observed that the synchronization signals from the PRBS receivers toggled once every 26th second as expected.

6.3.3 Error count test

In order to verify that the PRBS receivers receive the PRBS correctly, a signal indicating if more than 0 errors are registered by a PRBS receiver, was connected to one of the test outputs on the RX FPGAs. Unfortunately, several of the receivers that synchronized to the incoming PRBS, registered errors as well. However, by fine tuning the delay in the DCMs in the FPGA, it was accomplished to get 4 channels running error free for a period of 8 hours. Fine tuning of the DCM delay is tedious as it requires a new synthesis of the VHDL models and a reconfiguration (place and route and generation of configuration file) of the FPGA. Thus it was not possible to fine tune the delay in all 16 DCMs within this project, but it is believed that all channels will run error free once the DCM delay is correct.

Using one of the functional channels[1] it was verified that the PRBS receiver registers and counts errors correctly. This was done by inserting a certain amount of errors in the PRBS after the receiver had synchronized. The DIP switch on the PCB was used to insert the errors manually. The test outputs on the RX FPGA were configured to activate if, and only if, the error count delivered by the PRBS receiver, was equal to the specified amount of errors transmitted from the TX FPGA. Several different error counts were transmitted from the TX FPGA, and these errors were inserted once every 32 bits on every channel. As the receiver processes 32 bits at a time, the error count delivered on every clock cycle from a receiver, will correspond to the error count transmitted from the transmitter. The test showed that the PRBS receiver registers and counts errors correctly.

6.4 Improvement Suggestion

The fine tuning of the DCMs should be improved. It should be made possible to scan the valid sampling window using the dynamic delay adjustment option available on the DCMs in the FPGA. The dynamic delay adjustment of the DCMs works by incrementing the delay through the DCM on every

[1]One of the channels that had been verified to run error free

rising edge of an user defined clock signal. By making it possible to register the error count every time the delay increases, the valid data window can be defined, and the right delay configuration setting can be output. This will require that the RS-232 interface is operational, in order to output the delay configuration settings. If the PCB is redesigned, the termination resistors terminating the 16 311 MHz clock signals transmitted to the RX FPGAs should be placed closer to the RX FPGAs if possible.

6.4.1 Design errors

Only minor design errors were found, and they were easily overcome. The reference clock signals for the 16 demultiplexers in the DMUX stage was connected to the 78 MHz output of the GD16590 PLL clock synthesizer. These signals should have been connected to the 39 MHz output. This was corrected easily, but must be included if the PCB is to be redesigned. Furthermore the enable signal for the 39 MHz clock signal should be activated, while the enable pin for the 78 MHz clock signal should be deactivated. Unfortunately only 2 instead 3 PROMs for configuring the TX FPGA were included on the PCB. This was overcome by configuring the TX FPGA from a PC after power on.

6.5 Summary

The power supply on the test module has been tested, and all supply voltages stayed within the specified range at all loads. It has been verified that the clock distribution functions as expected. Data has been successfully transmitted from the TX FPGA, and the maximum skew between any two of the 64 LVDS signals on the TX FPGA output, was observed to be 200 ps. The data transmitted from the TX FPGA has been successfully multiplexed by the MUX stage, and the resulting 16 2.5 Gb/s data outputs have been verified to meet the requirements specified by the 40 Gb/s interface specification. The 16 2.5 Gb/s CML data signals have been successfully demultiplexed by the DeMUX stage, and the maximum skew between the 4 data signals output from each demultiplexer, was observed to be 200 ps. Reflections were observed on the edges of the 311 MHz clock signals input to the RX FPGAs, but these reflections should not influence the decision circuitry in the RX FPGA. It has been verified that 15 out of the 16 (one channel used for trigger signal) PRBS receivers in the FPGA synchronize to the incoming PRBS, and 4 of the 16 receivers has been running error free for a period of 8 hours. Furthermore, it has been verified that the PRBS receiver registers and counts errors correctly.

Chapter 7

Conclusion

The design and implementation of a 40 Gb/s PRBS transmitter/receiver module has been described, and the properties of pseudo random binary sequences have been investigated.

Through simulation and test, it has been verified that the PRBS generator functions correctly. Using VHDL macros supplied by Xilinx, it has been proven possible to transmit the PRBS at a rate of 40 Gb/s from a single FPGA. This data rate was accomplished using 64 622 Mb/s LVDS outputs, and measurements have verified that the transmitter FPGA is capable of delivering these 64 signals with a very low skew between them.

The 64 622 Mb/s LVDS data signals transmitted from the transmitter FPGA, have been successfully multiplexed by the 16 multiplexers in the multiplex stage. 15 out of the 16 2.5 Gb/s data signals on the back plane, have been confirmed to constitute identical $2^4 - 1$ PRBSs. The 16th 2.5 Gb/s data signal has been used as a pattern trigger signal, and has enabled displaying of the pattern of the PRBS signals on an oscilloscope. The phase relationship between the 16 2.5 Gb/s data signals on the output of the multiplex stage, has been verified to meet the 40 Gb/s interface specification. Furthermore, measurements have shown that the 1.25 GHz clock signal generated by the multiplex stage, meets the requirements specified in the 40 Gb/s interface specification.

The 16 2.5 Gb/s data signals have been successfully demultiplexed by the 16 demultiplexers in the demultiplex stage. It has been verified that the recovered 622 Mb/s data signals, and the recovered clock signals from the demultiplexers, are transmitted to the receiver FPGAs as intended. The 64 622 Mb/s data signals recovered by the demultiplexers have been measured at the input of the receiver FPGAs, and it has been verified, that each of these 64 data signals constitute identical PRBSs as expected.

Oscillations on the data signals on the receiver FPGA input have been observed, and it is believed that these oscillations are cross talk from neighbor signals. The maximum amplitude of these oscillations was observed to be 100 mV peak to peak. As the total differential voltage swing has been verified to be 400 mV peak to peak, these oscillations will not influence the decision circuitry in the receiver FPGAs. Furthermore, voltage variations on the edges of the 311 MHz clock signals were measured at the inputs of the receiver FPGAs. These voltage variations occur because the termination resistors terminating these clock signals, are placed too far away from the FPGA. These voltage variations occur early on the signal edges, and is believed not to have any influence on the decision circuitry in the receiver FPGAs.

The transmitting and receiving side of the test module have been connected through a loop back module, and through this loop back test, the PRBS receivers have been tested. The loop back test showed that 15 out of the 16 PRBS receivers synchronized to the incoming PRBS. The 16th PRBS receiver was not tested, as the signal interfacing this receiver was configured to be used as a pattern trigger signal. In order to verify that the PRBS receivers received the PRBS error free, a signal indicating if more than 0 errors were registered, was connected to the test outputs of the FPGA. The test showed that the majority of the receivers that had synchronized to the incoming PRBS, registered errors too. These errors are believed to be caused by misalignment of the clock and data signals inside the FPGA, thus leading to incorrect sampling of the input data. The alignment of the clock and data signals was fine tuned on 4 receivers, resulting in error free transmission of a $2^{31} - 1$ PRBS in a period of 8 hours. Fine tuning the alignment of the clock and data signals in the receiver is tedious, and it was not possible to fine tune all 16 receivers within this project. It is believed that all 16 receivers will run error free, once the clock and data signals in all receivers are aligned correctly.

This project has investigated some of the possibilities that the current FPGA technology offers. It has been shown that FPGAs indeed are viable choices for designs, that require high speed transmission and reception of data across a synchronous parallel interface. The design and implementation of the 40 Gb/s PRBS test module has, to a high degree, been successful. The problems remaining to be solved, are believed to be simple, and error free performance of all 16 receivers is expected to be easily obtained.

Bibliography

[1] Ando, "Pulse pattern generator/error detector ap9950/ap9951." Internet, March 2002. Downloaded from : http://www.ando.com/mid/pdfs/AP9950_5120020403E_b2.pdf in January 2003.

[2] S. C. T. AG, "Shf bpg 44e 44gbps pattern generator." Internet, December 2002. Downloaded from : http://www.shf-communication.de/pdf/bpg44e.pdf in January 2003.

[3] X. Inc., "The platform for programmable systems." Internet, 2002. Downloaded from : http://www.xilinx.com/products/virtex2pro/v2p_brochure.pdf in January 2003.

[4] C. Hede, *Pseudo Random Sequences - Coding, Autocorrelation and Spectral Aspects*. PhD thesis, DTU, June 1978.

[5] R. Mutagi, "Pseudo noise sequences for engineers," *Electronics and Communication Engineering*, pp. 79 – 87, April 1996.

[6] K. Dalgaard, "40+ gb/s optical transmitter and receiver design and implementation," Master's thesis, DTU, January 2003.

[7] IEEE, "Ieee standard for low-voltage differential signals (lvds) for scalable coherent interface (sci)," March 1996.

[8] Maxim, "Introduction to lvds, pecl, and cml." Internet, 10 2000. Downloaded from : http://pdfserv.maxim-ic.com/arpdf/AppNotes/hfan10v2.pdf in March 2002.

[9] B. P. Jensen, *Dimensionering og Konstruktion af Digitale Systemer*. BeeGs forlag, 5th ed., 1998.

[10] P. S. (Onsemi), "Interface between lvds and ecl." Internet, Feb 2002. Downloaded from : http://www.onsemi.com/pub/Collateral/AN1568-D.PDF in March 2002.

[11] N. S. X. Inc.), “High-speed data serialization and deserialization (840 mb/s lvds).” Internet, June 2002. Downloaded from http://www.xilinx.com/xapp/xapp265.pdf in March 2002.

[12] X. Inc., “Virtex ii user guide v1.5.” Internet, December 2002. Downloaded from : http://www.xilinx.com/publications/products/v2/handbook/index.htm in January 2003.

[13] ohjensen and mtheist, “Hx hardware document: S4 backplane requirements and design.” Internal Tellabs document, Dec 2001.

Appendix A

Schematic

15 pages

Appendix B

Component data sheets

B.1 Data sheet for the 622.08 MHz reference clock oscillator

2 pages

B.2 Data sheet for the MC100EP32 clock divider

12 pages

B.3 Data sheet for the GD16590 PLL clock synthesizer

12 pages

B.4 Data sheet for the MC100LVEL37 clock divider

8 pages

B.5 Datasheet for the MC100EP195 delay line

16 pages

B.6 Data sheet for the LXT16653 multiplexer

16 pages

B.7 Data sheet for the LXT16642 demultiplexer

14 pages

B.8 Data sheet for the NB100LVEP221 high fan-out clock buffers

12 pages

Appendix C

S4 Backplane information

The test module will be positioned in the ”40G Main” slot, and the MUX/DMUX module in the ”40G Opto” slot, and the opto module in the ”10G Opto” slot.

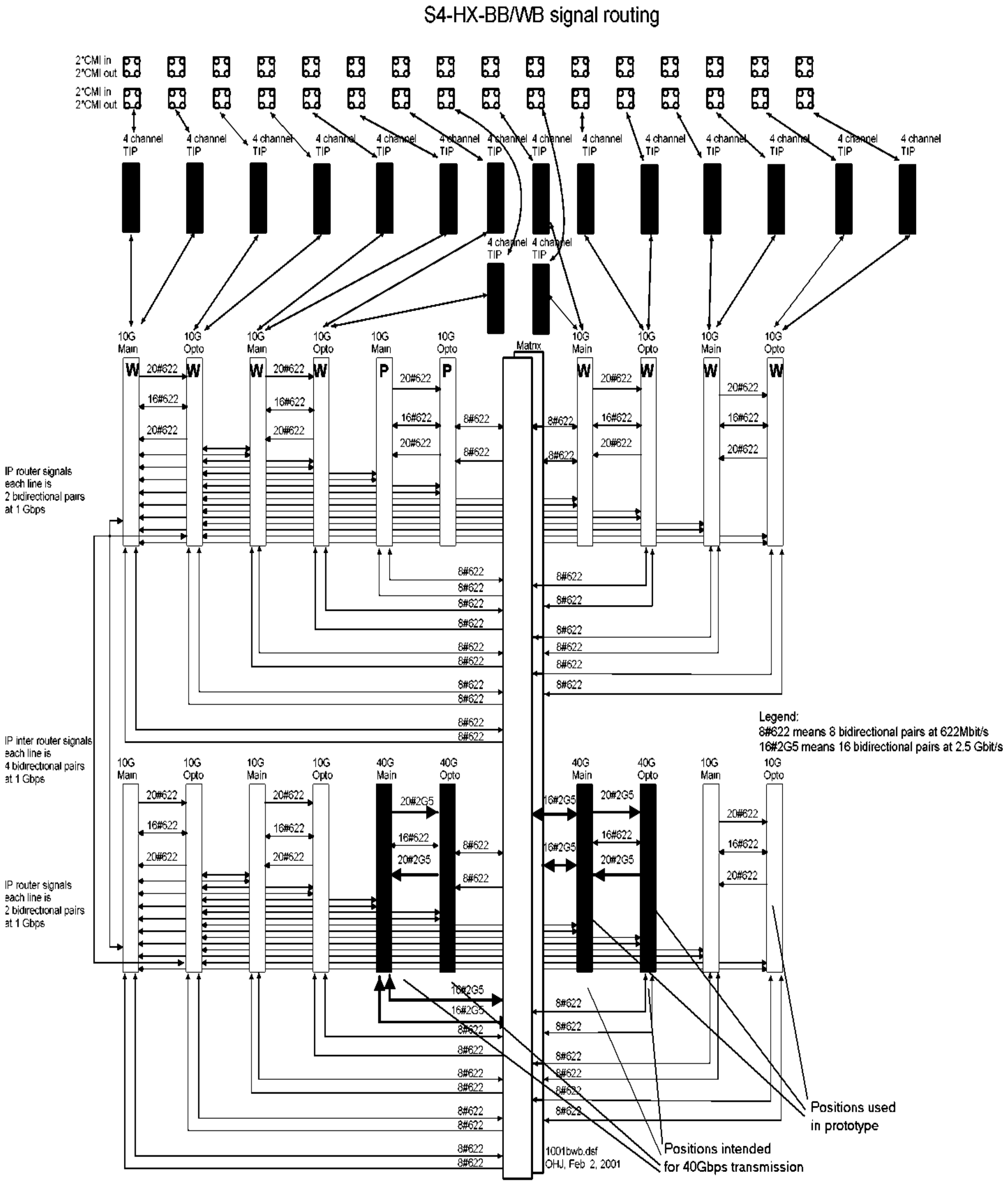

S4-HX-BB/WB signal routing
2*CMI in
2*CMI out
4 channel TIP
10G Main
10G Opto
40G Main
40G Opto
Matrix
W
P
20#622
16#622
8#622
20#2G5
16#2G5
IP router signals each line is 2 bidirectional pairs at 1 Gbps
IP inter router signals each line is 4 bidirectional pairs at 1 Gbps
Legend:
8#622 means 8 bidirectional pairs at 622Mbit/s
16#2G5 means 16 bidirectional pairs at 2.5 Gbit/s
Positions used in prototype
Positions intended for 40Gbps transmission
1001bwb.dsf
OHJ, Feb 2, 2001

Figure C.1: Illustration of the S4 backplane [13]

Appendix D

XAPP 265 application note from Xilinx

13 pages

Appendix E

Results from the TX FPGA simulation

E.1 Waveforms from the simulation of the PRBS generator

E.2 Waveforms from the simulation of the 64 bit transmitter

E.3 Waveforms from the simulation of the delay line configuration settings and the error transmitting functionality

Signal descriptions for the TX FPGA simulation	
Signal Name	**Signal Description**
`*/error_set_in`	Signals for enabling insertion of transmission errors (4 bit)
`*/delay_int`	Signal for controlling the `*/delayline_int` signal
`*/delayline_int`	Delay line configuration signals (11 bit)
`*/tx_top/prbs/reset`	Reset input on PRBS generator
`*/tx_top/prbs/clk`	80 MHz clock input for PRBS generator
`*/tx_top/prbs/prbs_data`	Output from PRBS generator (512 bit)
`*/tx_top/clkin_p`	311 MHz differential input clock for the TX FPGA
`*/tx_top/clkin_n`	311 MHz differential input clock for the TX FPGA
`*/tx_top/rstin`	Reset push button on TX FPGA
`*/tx_top/delayline_out`	Output configuration signals for the delay line (11 bit)
`*/tx_top/clkint`	Internal 311 MHz clock signal (input to DCM)
`*/tx_top/clkdcm`	80 MHz output clock signal from DCM
`*/tx_top/clkx4`	311 MHz global clock signal for 64 bit transmitter
`*/tx_top/clkx4not`	Inverted 311 MHz global clock signal for 64 bit transmitter
`*/tx_top/clkx4dcm`	311 MHz output clock signal from DCM
`*/tx_top/clkx4notdcm`	311 MHz inverted clock signal from DCM
`*/tx_top/clk_locked`	Locked signal from DCM
`*/tx_top/not_clk_locked`	Inverted locked signal
`*/tx_top/clkoe`	Clock output enable for 64 bit transmitter
`*/tx_top/dataoe`	Data output enable for 64 bit transmitter
`*/tx_top/rst_clk`	Clocked reset signal (controlled by `clk_locked`)
`*/tx_top/clk`	80 MHz global clock signal
`*/tx_top/txdata_int`	Connected to output from PRBS generator
`*/tx_top/tx0/clk`	80 MHz clock input on the 64 bit transmitter
`*/tx_top/tx0/clkx4`	311 MHz clock input on the 64 bit transmitter
`*/tx_top/tx0/clkx4not`	Inverted 311 MHz clock input on the 64 bit transmitter
`*/tx_top/tx0/datain`	512 bit data input on the 64 bit transmitter
`*/tx_top/tx0/rst`	Reset input on the 64 bit transmitter
`*/tx_top/tx0/dataoe`	Data output enable on the 64 bit transmitter
`*/tx_top/tx0/dataout`	64 bit output data from the 64 bit transmitter
`*/tx_top/tx0/buffera to bufferp`	32 bit input data for the 16 TX macros
`*/tx_top/tx0/tx0/datain`	32 bit data input for TX macro 0
`*/tx_top/tx0/tx0/dataout`	4 bit output data from TX macro 0

Table E.1: Description of the signals shown in the waveforms resulting from the simulation of the TX FPGA design. The "*/" represents /tst_top_tx_rx/

Appendix F

Results from the RX FPGA simulation

Signal descriptions for the RX FPGA simulation	
Signal Name	**Signal Description**
`*/rx_top_1/datain_p`	32 bit differential data input on RX FPGA 1
`*/rx_top_1/datain_n`	32 bit differential data input on RX FPGA 1
`*/rx_top_1/rxdata`	8 x 32 bit output data signals from the 32 bit receiver
`*/rx_top_1/err_count`	8 x 20 bit error count outputs from the 8 PRBS receivers
`*/rx_top_1/sync_pattern`	8 x 32 synchronization pattern for the 8 PRBS receivers
`*/rx_top_1/detect_out`	8 x synchronization signals from the 8 PRBS receivers
`*/rx_top_1/clk80`	80 MHz global clock signal
`*/rx_top_1/test_output_#`	Output used for test purposes
`*/rx_top_1/rst_clk`	Reset signal controlled by TX DCM lock signal
`*/rx_top_1/rxclkint`	311 MHz input clock signals for the RX DCMs
`*/rx_top_1/rxclkdcm`	311 MHz output clock signals from the RX DCMs
`*/rx_top_1/clkdcm`	80 MHz output clock signal from DCM for clocking data out from the 8 FIFOs
`*/rx_top_1/rxclknotdcm`	Inverted version of `rxclkdcm`

Table F.1: Description of the signals shown in the waveforms resulting from the simulation of the RX FPGA design. The "*/" represents /tst_top_tx_rx/

F.1 Waveforms from the simulation of the 32 bit receiver and the error counting functionality of the PRBS receivers

F.2 Waveforms from the simulation of the PRBS receiver

F.3 Waveforms showing a closeup of the error count output from the PRBS receivers

F.4 Waveforms showing the functionality of the receiver DCMs

F.5 Waveforms showing that the reset signals function correctly

Appendix G

UCF files for the TX and RX FPGA designs

```
############################################################
###########                                 ################
########### UCF file for the TX FPGA design ################
###########    Timespec is for 622 Mb/s     ################
########### Pinout is for a FF1152 package  ################
###########                                 ################
############################################################

#specifying clock frequency of net clk
net clk period = 78 Mhz ;

#Defining Timing group
net clk tnm = clk ;
net clkx4 tnm = clkx4 ;
net clkx4not tnm = clkx4not ;

#Specifying the delay between synchronous elements located in the macros
timespec tstx00 = period clkx4 311 MHz ;
timespec tstx01 = period clkx4not 311 Mhz ;
timespec tstx02 = from clk to clkx4 = tstx00 ;
timespec tstx03 = from ffs(tx0/*syncpl*) to latches(tx0/*xrb) = tstx00*1.34 ;
timespec tstx04 = from latches(tx0/*xrb) to ffs(tx0/*syncn) = tstx00*1.34 ;
timespec tstx05 = from ffs(tx0/*oeint) to ffs = tstx00/4 ;
#timespec tstx06 = from ffs(dataoe*) to ffs(tx0/*oeint) = tstx00/4 ;
timespec tstx07 = from ffs(clkoe*) to ffs(tx0/*oeint) = tstx00/4 ;
timespec tstx08 = from ffs(rst_*) to ffs = tstx00/4 ;
timespec tstx09 = from ffs(tx0/*db*) to ffs(tx0/*) = tstx00/2 ;

#Reset PIN loc

#NET "rstin" loc = "AM14" ;

#Specifying the location of the 16 four bit TX macros
set "tx0/tx0/hset" rloc_origin =  "X0Y120" ;
set "tx0/tx1/hset" rloc_origin =  "X0Y112" ;
set "tx0/tx2/hset" rloc_origin =  "X0Y104" ;
set "tx0/tx3/hset" rloc_origin =  "X0Y96" ;
set "tx0/tx4/hset" rloc_origin =  "X0Y88" ;
set "tx0/tx5/hset" rloc_origin =  "X0Y80" ;
set "tx0/tx6/hset" rloc_origin =  "X0Y72" ;
set "tx0/tx7/hset" rloc_origin =  "X0Y64" ;
set "tx0/tx8/hset" rloc_origin =  "X0Y56" ;
set "tx0/tx9/hset" rloc_origin =  "X0Y48" ;
set "tx0/tx10/hset" rloc_origin = "X0Y40" ;
set "tx0/tx11/hset" rloc_origin = "X0Y32" ;
set "tx0/tx12/hset" rloc_origin = "X0Y24" ;
set "tx0/tx13/hset" rloc_origin = "X0Y16" ;
set "tx0/tx14/hset" rloc_origin = "X0Y8" ;
set "tx0/tx15/hset" rloc_origin = "X0Y0" ;



#PIN LOCATIONS - The locations of the data I/Os are selected
#correspoding to placement of the TX macros
```

```
NET "clkin_n" LOC = "E19"  ;
NET "clkin_p" LOC = "E18"  ;
NET "clktx_locked_out_1" LOC = "F3"  ;
NET "clktx_locked_out_2" LOC = "K8"  ;
NET "dataouta_n(0)" LOC = "E33"  ;
NET "dataouta_n(1)" LOC = "F30"  ;
NET "dataouta_n(2)" LOC = "K24"  ;
NET "dataouta_n(3)" LOC = "E32"  ;
NET "dataouta_n(4)" LOC = "F31"  ;
NET "dataouta_n(5)" LOC = "J27"  ;
NET "dataouta_n(6)" LOC = "K26"  ;
NET "dataouta_n(7)" LOC = "H29"  ;
NET "dataouta_n(8)" LOC = "J29"  ;
NET "dataouta_n(9)" LOC = "H30"  ;
NET "dataouta_n(10)" LOC = "M25"  ;
NET "dataouta_n(11)" LOC = "G32"  ;
NET "dataouta_n(12)" LOC = "J31"  ;
NET "dataouta_n(13)" LOC = "J34"  ;
NET "dataouta_n(14)" LOC = "M26"  ;
NET "dataouta_n(15)" LOC = "J32"  ;
NET "dataouta_n(16)" LOC = "K31"  ;
NET "dataouta_n(17)" LOC = "L29"  ;
NET "dataouta_n(18)" LOC = "L30"  ;
NET "dataouta_n(19)" LOC = "M31"  ;
NET "dataouta_n(20)" LOC = "M32"  ;
NET "dataouta_n(21)" LOC = "N29"  ;
NET "dataouta_n(22)" LOC = "P25"  ;
NET "dataouta_n(23)" LOC = "P30"  ;
NET "dataouta_n(24)" LOC = "P31"  ;
NET "dataouta_n(25)" LOC = "R28"  ;
NET "dataouta_n(26)" LOC = "R27"  ;
NET "dataouta_n(27)" LOC = "P32"  ;
NET "dataouta_n(28)" LOC = "U30"  ;
NET "dataouta_n(29)" LOC = "T29"  ;
NET "dataouta_n(30)" LOC = "U28"  ;
NET "dataouta_n(31)" LOC = "V31"  ;
NET "dataouta_n(32)" LOC = "W32"  ;
NET "dataouta_n(33)" LOC = "V27"  ;
NET "dataouta_n(34)" LOC = "W31"  ;
NET "dataouta_n(35)" LOC = "V28"  ;
NET "dataouta_n(36)" LOC = "AB30"  ;
NET "dataouta_n(37)" LOC = "Y29"  ;
NET "dataouta_n(38)" LOC = "AA31"  ;
NET "dataouta_n(39)" LOC = "W27"  ;
NET "dataouta_n(40)" LOC = "AD31"  ;
NET "dataouta_n(41)" LOC = "Y26"  ;
NET "dataouta_n(42)" LOC = "AB32"  ;
NET "dataouta_n(43)" LOC = "Y25"  ;
NET "dataouta_n(44)" LOC = "AA27"  ;
NET "dataouta_n(45)" LOC = "AB28"  ;
NET "dataouta_n(46)" LOC = "AG33"  ;
NET "dataouta_n(47)" LOC = "AB26"  ;
NET "dataouta_n(48)" LOC = "AD30"  ;
NET "dataouta_n(49)" LOC = "AC27"  ;
NET "dataouta_n(50)" LOC = "AD32"  ;
NET "dataouta_n(51)" LOC = "AB25"  ;
NET "dataouta_n(52)" LOC = "AG32"  ;
NET "dataouta_n(53)" LOC = "AD26"  ;
NET "dataouta_n(54)" LOC = "AD28"  ;
NET "dataouta_n(55)" LOC = "AE27"  ;
NET "dataouta_n(56)" LOC = "AE29"  ;
NET "dataouta_n(57)" LOC = "AF28"  ;
NET "dataouta_n(58)" LOC = "AH32"  ;
NET "dataouta_n(59)" LOC = "AF26"  ;
NET "dataouta_n(60)" LOC = "AG29"  ;
NET "dataouta_n(61)" LOC = "AE25"  ;
NET "dataouta_n(62)" LOC = "AK32"  ;
NET "dataouta_n(63)" LOC = "AD25"  ;
NET "dataouta_p(0)" LOC = "D33"  ;
NET "dataouta_p(1)" LOC = "G30"  ;
NET "dataouta_p(2)" LOC = "L25"  ;
NET "dataouta_p(3)" LOC = "D32"  ;
NET "dataouta_p(4)" LOC = "E31"  ;
NET "dataouta_p(5)" LOC = "J28"  ;
NET "dataouta_p(6)" LOC = "L26"  ;
NET "dataouta_p(7)" LOC = "G29"  ;
NET "dataouta_p(8)" LOC = "H28"  ;
NET "dataouta_p(9)" LOC = "J30"  ;
NET "dataouta_p(10)" LOC = "N25"  ;
NET "dataouta_p(11)" LOC = "F32"  ;
NET "dataouta_p(12)" LOC = "H31"  ;
NET "dataouta_p(13)" LOC = "H33"  ;
NET "dataouta_p(14)" LOC = "N26"  ;
NET "dataouta_p(15)" LOC = "H32"  ;
NET "dataouta_p(16)" LOC = "K30"  ;
NET "dataouta_p(17)" LOC = "M29"  ;
NET "dataouta_p(18)" LOC = "K29"  ;
NET "dataouta_p(19)" LOC = "L31"  ;
NET "dataouta_p(20)" LOC = "L32"  ;
NET "dataouta_p(21)" LOC = "P29"  ;
NET "dataouta_p(22)" LOC = "R25"  ;
NET "dataouta_p(23)" LOC = "N30"  ;
NET "dataouta_p(24)" LOC = "N31"  ;
NET "dataouta_p(25)" LOC = "R29"  ;
NET "dataouta_p(26)" LOC = "T27"  ;
NET "dataouta_p(27)" LOC = "N32"  ;
NET "dataouta_p(28)" LOC = "T30"  ;
NET "dataouta_p(29)" LOC = "U29"  ;
NET "dataouta_p(30)" LOC = "T28"  ;
NET "dataouta_p(31)" LOC = "U31"  ;
NET "dataouta_p(32)" LOC = "V32"  ;
NET "dataouta_p(33)" LOC = "V26"  ;
NET "dataouta_p(34)" LOC = "Y31"  ;
NET "dataouta_p(35)" LOC = "W28"  ;
NET "dataouta_p(36)" LOC = "AA30"  ;
NET "dataouta_p(37)" LOC = "Y28"  ;
NET "dataouta_p(38)" LOC = "AB31"  ;
NET "dataouta_p(39)" LOC = "Y27"  ;
NET "dataouta_p(40)" LOC = "AC31"  ;
NET "dataouta_p(41)" LOC = "AA26"  ;
NET "dataouta_p(42)" LOC = "AC32"  ;
NET "dataouta_p(43)" LOC = "AA25"  ;
NET "dataouta_p(44)" LOC = "AB27"  ;
NET "dataouta_p(45)" LOC = "AC28"  ;
NET "dataouta_p(46)" LOC = "AF33"  ;
NET "dataouta_p(47)" LOC = "AC26"  ;
NET "dataouta_p(48)" LOC = "AE30"  ;
NET "dataouta_p(49)" LOC = "AD27"  ;
NET "dataouta_p(50)" LOC = "AE31"  ;
NET "dataouta_p(51)" LOC = "AC25"  ;
NET "dataouta_p(52)" LOC = "AF32"  ;
NET "dataouta_p(53)" LOC = "AE26"  ;
NET "dataouta_p(54)" LOC = "AE28"  ;
NET "dataouta_p(55)" LOC = "AF27"  ;
NET "dataouta_p(56)" LOC = "AF29"  ;
NET "dataouta_p(57)" LOC = "AG28"  ;
NET "dataouta_p(58)" LOC = "AJ32"  ;
NET "dataouta_p(59)" LOC = "AG26"  ;
NET "dataouta_p(60)" LOC = "AH29"  ;
NET "dataouta_p(61)" LOC = "AF25"  ;
NET "dataouta_p(62)" LOC = "AL32"  ;
NET "dataouta_p(63)" LOC = "AE24"  ;
NET "delayline_out(0)" LOC = "AJ16"  ;
NET "delayline_out(1)" LOC = "AL17"  ;
NET "delayline_out(2)" LOC = "AK13"  ;
NET "delayline_out(3)" LOC = "AD17"  ;
NET "delayline_out(4)" LOC = "AP13"  ;
NET "delayline_out(5)" LOC = "AN13"  ;
NET "delayline_out(6)" LOC = "AG16"  ;
NET "delayline_out(7)" LOC = "AP11"  ;
NET "delayline_out(8)" LOC = "AM12"  ;
NET "delayline_out(9)" LOC = "AF15"  ;
NET "delayline_out(10)" LOC = "AJ14"  ;
NET "error_set_in(0)" LOC = "C22"  ;
NET "error_set_in(1)" LOC = "K20"  ;
NET "error_set_in(2)" LOC = "F22"  ;
NET "error_set_in(3)" LOC = "G23"  ;
NET "delay_set_in(0)" LOC = "D22"  ;
NET "delay_set_in(1)" LOC = "D24"  ;
NET "rx_sync" LOC = "J7"  ;
NET "rx_sync" IOSTANDARD = LVCMOS33;

#Specifying location of clock buffer
INST "clk_bufg" LOC = "bufgmux0s"  ;
INST "clkx4_bufg" LOC = "bufgmux1p"  ;
INST "clkx4not_bufg" LOC = "bufgmux3p"  ;
#Specifying location of DCM
INST "dcm_clk" LOC = "dcm_x2y1"  ;
```

```
####################################################################
##############                                 ####################
############## UCF file for the RX FPGA 1 design ####################
##############   Timespec is for 622 Mb/s        ####################
############## Pinout is for a FG456 Package     ####################
##############                                 ####################
####################################################################

#Specifying phase between input data and clock
net dataina_p(0) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(1) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(2) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(3) offset = in 1.7 nS before rxclkina_p(0) ;

net dataina_p(4) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(5) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(6) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(7) offset = in 1.7 nS before rxclkina_p(1) ;

net dataina_p(8) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(9) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(10) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(11) offset = in 1.7 nS before rxclkina_p(2) ;

net dataina_p(12) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(13) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(14) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(15) offset = in 1.7 nS before rxclkina_p(3) ;

net dataina_p(16) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(17) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(18) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(19) offset = in 1.7 nS before rxclkina_p(4) ;

net dataina_p(20) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(21) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(22) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(23) offset = in 1.7 nS before rxclkina_p(5) ;

net dataina_p(24) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(25) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(26) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(27) offset = in 1.7 nS before rxclkina_p(6) ;

net dataina_p(28) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(29) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(30) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(31) offset = in 1.7 nS before rxclkina_p(7) ;

net dataina_p(*) NODELAY ;
net dataina_n(*) NODELAY ;

#Defining timing groups
net rxclk(*) tnm = rxclk ;
```

```
#net rxclknot* tnm = rxclknot* ;
net rxclkina_p* tnm = rxclkina_p* ;
net rxclkina_n* tnm = rxclkina_n* ;
net clkdcm tnm = clkdcm;


#Specifying the delay between synchronous elements located in the macros
timespec tsrx00 = period rxclk 311 Mhz ;
#timespec tsrx01 = period rxclknot* 311 MHz ;
timespec tsrx02 = from ffs(rx0/*dmuxnb*) to rams = tsrx00/1.34 ;
timespec tsrx03 = from ffs(rx0/*dmuxpb*) to rams = tsrx00/2.00 ;
timespec tsrx04 = from ffs(rx0/*ram_en) to rams(rx0/*) = tsrx00 ;
timespec tsrx05 = from rams(rx0/*) to rams(rx0/*) = tsrx00/4.00 ;
timespec tsrx06 = from ffs(rx0/*syncp*) to latches(rx0/*syncl*) = tsrx00*1.34 ;
timespec tsrx07 = from latches(rx0/*syncl*) to ffs(rx0/*syncn*) = tsrx00*1.34 ;
timespec tsrx08 = from ffs(rx0/*rst) to ffs = tsrx00/2 ;
timespec tsrx09 = period clkdcm 80 Mhz ;


#Specifying location of clock inputs

NET "rxclkina_p(0)" LOC = "D12" ;
NET "rxclkina_n(0)" LOC = "E12" ;
NET "rxclkina_p(1)" LOC = "F13" ;
NET "rxclkina_n(1)" LOC = "F12" ;
NET "rxclkina_p(2)" LOC = "A11" ;
NET "rxclkina_n(2)" LOC = "B11" ;
NET "rxclkina_p(3)" LOC = "C11" ;
NET "rxclkina_n(3)" LOC = "D11" ;
NET "rxclkina_p(4)" LOC = "AB12" ;
NET "rxclkina_n(4)" LOC = "AA12" ;
NET "rxclkina_p(5)" LOC = "Y12" ;
NET "rxclkina_n(5)" LOC = "W12" ;
NET "rxclkina_p(6)" LOC = "V11" ;
NET "rxclkina_n(6)" LOC = "W11" ;
NET "rxclkina_p(7)" LOC = "Y11" ;
NET "rxclkina_n(7)" LOC = "AA11" ;

# Specifying location of data inputs - These location are closely
# related to the placement of the RX macros

NET "dataina_p(31)" LOC = "T2"  ;
NET "dataina_n(31)" LOC = "T1"  ;
NET "dataina_p(30)" LOC = "R4"  ;
NET "dataina_n(30)" LOC = "R3"  ;
NET "dataina_p(29)" LOC = "R2"  ;
NET "dataina_n(29)" LOC = "R1"  ;
NET "dataina_p(28)" LOC = "P6"  ;
NET "dataina_n(28)" LOC = "P5"  ;
NET "dataina_p(27)" LOC = "P4"  ;
NET "dataina_n(27)" LOC = "P3"  ;
NET "dataina_p(26)" LOC = "P2"  ;
NET "dataina_n(26)" LOC = "P1"  ;
NET "dataina_p(25)" LOC = "N6"  ;
NET "dataina_n(25)" LOC = "N5"  ;
NET "dataina_p(24)" LOC = "N4"  ;
NET "dataina_n(24)" LOC = "N3"  ;
NET "dataina_p(23)" LOC = "N2"  ;
NET "dataina_n(23)" LOC = "N1"  ;
NET "dataina_p(22)" LOC = "M6"  ;
NET "dataina_n(22)" LOC = "M5"  ;
NET "dataina_p(21)" LOC = "M4"  ;
NET "dataina_n(21)" LOC = "M3"  ;
NET "dataina_p(20)" LOC = "M2"  ;
NET "dataina_n(20)" LOC = "M1"  ;
NET "dataina_p(19)" LOC = "L2"  ;
NET "dataina_n(19)" LOC = "L3"  ;
NET "dataina_p(18)" LOC = "L4"  ;
NET "dataina_n(18)" LOC = "L5"  ;
NET "dataina_p(17)" LOC = "K1"  ;
NET "dataina_n(17)" LOC = "K2"  ;
NET "dataina_p(16)" LOC = "K3"  ;
NET "dataina_n(16)" LOC = "K4"  ;
NET "dataina_p(15)" LOC = "L6"  ;
NET "dataina_n(15)" LOC = "K6"  ;
NET "dataina_p(14)" LOC = "K5"  ;
NET "dataina_n(14)" LOC = "J5"  ;
NET "dataina_p(13)" LOC = "J1"  ;
NET "dataina_n(13)" LOC = "J2"  ;
NET "dataina_p(12)" LOC = "J3"  ;
NET "dataina_n(12)" LOC = "J4"  ;
NET "dataina_p(11)" LOC = "H1"  ;
NET "dataina_n(11)" LOC = "H2"  ;
NET "dataina_p(10)" LOC = "H3"  ;
NET "dataina_n(10)" LOC = "H4"  ;
NET "dataina_p(9)" LOC = "J6"  ;
NET "dataina_n(9)" LOC = "H5"  ;
NET "dataina_p(8)" LOC = "G1"  ;
NET "dataina_n(8)" LOC = "G2"  ;
NET "dataina_p(7)" LOC = "G3"  ;
NET "dataina_n(7)" LOC = "G4"  ;
NET "dataina_p(6)" LOC = "F1"  ;
NET "dataina_n(6)" LOC = "F2"  ;
NET "dataina_p(5)" LOC = "F3"  ;
NET "dataina_n(5)" LOC = "F4"  ;
NET "dataina_p(4)" LOC = "G5"  ;
NET "dataina_n(4)" LOC = "F5"  ;
NET "dataina_p(3)" LOC = "E1"  ;
NET "dataina_n(3)" LOC = "E2"  ;
NET "dataina_p(2)" LOC = "E3"  ;
NET "dataina_n(2)" LOC = "E4"  ;
NET "dataina_p(1)" LOC = "D1"  ;
NET "dataina_n(1)" LOC = "D2"  ;
NET "dataina_p(0)" LOC = "E5"  ;
NET "dataina_n(0)" LOC = "E6"  ;

# miscellaneous pins
```

```
#Global reset
NET "rstin" loc = "L19" ;
#Error indicator
NET "error_output_ch0" loc = "T22";
NET "error_output_ch0_1" loc = "T20";
NET "error_output_ch0_2" loc = "T18";

#NET "clkrx_locked_out" loc = "T20";
NET "clktx_locked" loc = "K20";

#Sync output to test pad
#NET "sync_out" loc = "T22";
NET "sync_out" loc = "U19";

#Sync output to TX FPGA
#NET "sync_out" loc = "K18";

#NET "power" loc = "U19" ;
#NET "dcmtxlock" loc = "T18";



#Locs representing the origins of the 8 4 bit receiver macros

set "rx0/rx0/hset" rloc_origin = "X0y72" ;
set "rx0/rx1/hset" rloc_origin = "X0y64" ;
set "rx0/rx2/hset" rloc_origin = "X0y56" ;
set "rx0/rx3/hset" rloc_origin = "X0y48" ;
set "rx0/rx4/hset" rloc_origin = "X0y40" ;
set "rx0/rx5/hset" rloc_origin = "X0y32" ;
set "rx0/rx6/hset" rloc_origin = "X0y24" ;
set "rx0/rx7/hset" rloc_origin = "X0y16" ;


#Locs for the 8 block RAMs used in the 8 4 bit receiver macros

INST "rx0/rx0/block_ram" LOC = "RAMB16_X0Y9"  ;
INST "rx0/rx1/block_ram" LOC = "RAMB16_X0Y8"  ;
INST "rx0/rx2/block_ram" LOC = "RAMB16_X0Y7"  ;
INST "rx0/rx3/block_ram" LOC = "RAMB16_X0Y6"  ;
INST "rx0/rx4/block_ram" LOC = "RAMB16_X0Y5"  ;
INST "rx0/rx5/block_ram" LOC = "RAMB16_X0Y4"  ;
INST "rx0/rx6/block_ram" LOC = "RAMB16_X0Y3"  ;
INST "rx0/rx7/block_ram" LOC = "RAMB16_X0Y2"  ;

# Specifying the location of the 8 DCMs
inst "dcm_rxclk0" loc = "dcm_x0y1" ;
inst "dcm_rxclk1" loc = "dcm_x1y1" ;
inst "dcm_rxclk2" loc = "dcm_x2y1" ;
inst "dcm_rxclk3" loc = "dcm_x3y1" ;
inst "dcm_rxclk4" loc = "dcm_x0y0" ;
inst "dcm_rxclk5" loc = "dcm_x1y0" ;
inst "dcm_rxclk6" loc = "dcm_x2y0" ;
inst "dcm_rxclk7" loc = "dcm_x3y0" ;
```

```
# Specifying the location of the global clock buffer used
# The selection of global clock buffers must obey the
# rules for primary/secondary clock buffer access to the
# 4 quadrants of the FPGA
inst "rxclk_bufg0" loc = "bufgmux7p" ;
#inst "rxclknot_bufg0" loc = "bufgmux6s" ;
inst "rxclk_bufg1" loc = "bufgmux6s" ;
#inst "rxclknot_bufg1" loc = "bufgmux4s" ;
inst "rxclk_bufg2" loc = "bufgmux5p" ;
#inst "rxclknot_bufg2" loc = "bufgmux2s" ;
inst "rxclk_bufg3" loc = "bufgmux4s" ;
#inst "rxclknot_bufg3" loc = "bufgmux0s" ;
inst "rxclk_bufg4" loc = "bufgmux3p" ;
#inst "rxclknot_bufg4" loc = "bufgmux7s" ;
inst "rxclk_bufg5" loc = "bufgmux7s" ;
#inst "rxclknot_bufg5" loc = "bufgmux4p" ;
inst "rxclk_bufg6" loc = "bufgmux6p" ;
#inst "rxclknot_bufg6" loc = "bufgmux2p" ;
inst "rxclk_bufg7" loc = "bufgmux5s" ;
#inst "rxclknot_bufg7" loc = "bufgmux0p" ;


#inst "dcm_clk" loc = "dcm_x2y1" ;
#inst "clk_bufg" loc = "bufgmux0s" ;
```

```
####################################################################
##############                                  ####################
############## UCF file for the RX FPGA 2 design ####################
##############   Timespec is for 622 Mb/s        ####################
############## Pinout is for a FG456 Package     ####################
##############                                  ####################
####################################################################

#Specifying phase between input data and clock
net dataina_p(0) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(1) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(2) offset = in 1.7 nS before rxclkina_p(0) ;
net dataina_p(3) offset = in 1.7 nS before rxclkina_p(0) ;

net dataina_p(4) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(5) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(6) offset = in 1.7 nS before rxclkina_p(1) ;
net dataina_p(7) offset = in 1.7 nS before rxclkina_p(1) ;

net dataina_p(8) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(9) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(10) offset = in 1.7 nS before rxclkina_p(2) ;
net dataina_p(11) offset = in 1.7 nS before rxclkina_p(2) ;

net dataina_p(12) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(13) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(14) offset = in 1.7 nS before rxclkina_p(3) ;
net dataina_p(15) offset = in 1.7 nS before rxclkina_p(3) ;

net dataina_p(16) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(17) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(18) offset = in 1.7 nS before rxclkina_p(4) ;
net dataina_p(19) offset = in 1.7 nS before rxclkina_p(4) ;

net dataina_p(20) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(21) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(22) offset = in 1.7 nS before rxclkina_p(5) ;
net dataina_p(23) offset = in 1.7 nS before rxclkina_p(5) ;

net dataina_p(24) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(25) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(26) offset = in 1.7 nS before rxclkina_p(6) ;
net dataina_p(27) offset = in 1.7 nS before rxclkina_p(6) ;

net dataina_p(28) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(29) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(30) offset = in 1.7 nS before rxclkina_p(7) ;
net dataina_p(31) offset = in 1.7 nS before rxclkina_p(7) ;

net dataina_p(*) NODELAY ;
net dataina_n(*) NODELAY ;

#Defining timing groups
net rxclk(*) tnm = rxclk ;
```

```
#net rxclknot* tnm = rxclknot* ;
net rxclkina_p* tnm = rxclkina_p* ;
net rxclkina_n* tnm = rxclkina_n* ;
net clkdcm tnm = clkdcm;


#Specifying the delay between synchronous elements located in the macros
timespec tsrx00 = period rxclk 311 Mhz ;
#timespec tsrx01 = period rxclknot* 311 MHz ;
timespec tsrx02 = from ffs(rx0/*dmuxnb*) to rams = tsrx00/1.34 ;
timespec tsrx03 = from ffs(rx0/*dmuxpb*) to rams = tsrx00/2.00 ;
timespec tsrx04 = from ffs(rx0/*ram_en) to rams(rx0/*) = tsrx00 ;
timespec tsrx05 = from rams(rx0/*) to rams(rx0/*) = tsrx00/4.00 ;
timespec tsrx06 = from ffs(rx0/*syncp*) to latches(rx0/*syncl*) = tsrx00*1.34 ;
timespec tsrx07 = from latches(rx0/*syncl*) to ffs(rx0/*syncn*) = tsrx00*1.34 ;
timespec tsrx08 = from ffs(rx0/*rst) to ffs = tsrx00/2 ;
timespec tsrx09 = period clkdcm 80 Mhz ;


#Specifying location of clock inputs

NET "rxclkina_p(0)" LOC = "D12" ;
NET "rxclkina_n(0)" LOC = "E12" ;
NET "rxclkina_p(1)" LOC = "F13" ;
NET "rxclkina_n(1)" LOC = "F12" ;
NET "rxclkina_p(2)" LOC = "A11" ;
NET "rxclkina_n(2)" LOC = "B11" ;
NET "rxclkina_p(3)" LOC = "C11" ;
NET "rxclkina_n(3)" LOC = "D11" ;
NET "rxclkina_p(4)" LOC = "AB12" ;
NET "rxclkina_n(4)" LOC = "AA12" ;
NET "rxclkina_p(5)" LOC = "Y12" ;
NET "rxclkina_n(5)" LOC = "W12" ;
NET "rxclkina_p(6)" LOC = "V11" ;
NET "rxclkina_n(6)" LOC = "W11" ;
NET "rxclkina_p(7)" LOC = "Y11" ;
NET "rxclkina_n(7)" LOC = "AA11" ;

# Specifying location of data inputs - These location are closely
# related to the placement of the RX macros

NET "dataina_p(31)" LOC = "T2"  ;
NET "dataina_n(31)" LOC = "T1"  ;
NET "dataina_p(30)" LOC = "R4"  ;
NET "dataina_n(30)" LOC = "R3"  ;
NET "dataina_p(29)" LOC = "R2"  ;
NET "dataina_n(29)" LOC = "R1"  ;
NET "dataina_p(28)" LOC = "P6"  ;
NET "dataina_n(28)" LOC = "P5"  ;
NET "dataina_p(27)" LOC = "P4"  ;
NET "dataina_n(27)" LOC = "P3"  ;
NET "dataina_p(26)" LOC = "P2"  ;
NET "dataina_n(26)" LOC = "P1"  ;
NET "dataina_p(25)" LOC = "N6"  ;
NET "dataina_n(25)" LOC = "N5"  ;
NET "dataina_p(24)" LOC = "N4"  ;
NET "dataina_n(24)" LOC = "N3"  ;
NET "dataina_p(23)" LOC = "N2"  ;
NET "dataina_n(23)" LOC = "N1"  ;
NET "dataina_p(22)" LOC = "M6"  ;
NET "dataina_n(22)" LOC = "M5"  ;
NET "dataina_p(21)" LOC = "M4"  ;
NET "dataina_n(21)" LOC = "M3"  ;
NET "dataina_p(20)" LOC = "M2"  ;
NET "dataina_n(20)" LOC = "M1"  ;
NET "dataina_p(19)" LOC = "L2"  ;
NET "dataina_n(19)" LOC = "L3"  ;
NET "dataina_p(18)" LOC = "L4"  ;
NET "dataina_n(18)" LOC = "L5"  ;
NET "dataina_p(17)" LOC = "K1"  ;
NET "dataina_n(17)" LOC = "K2"  ;
NET "dataina_p(16)" LOC = "K3"  ;
NET "dataina_n(16)" LOC = "K4"  ;
NET "dataina_p(15)" LOC = "L6"  ;
NET "dataina_n(15)" LOC = "K6"  ;
NET "dataina_p(14)" LOC = "K5"  ;
NET "dataina_n(14)" LOC = "J5"  ;
NET "dataina_p(13)" LOC = "J1"  ;
NET "dataina_n(13)" LOC = "J2"  ;
NET "dataina_p(12)" LOC = "J3"  ;
NET "dataina_n(12)" LOC = "J4"  ;
NET "dataina_p(11)" LOC = "H1"  ;
NET "dataina_n(11)" LOC = "H2"  ;
NET "dataina_p(10)" LOC = "H3"  ;
NET "dataina_n(10)" LOC = "H4"  ;
NET "dataina_p(9)" LOC = "J6"  ;
NET "dataina_n(9)" LOC = "H5"  ;
NET "dataina_p(8)" LOC = "G1"  ;
NET "dataina_n(8)" LOC = "G2"  ;
NET "dataina_p(7)" LOC = "G3"  ;
NET "dataina_n(7)" LOC = "G4"  ;
NET "dataina_p(6)" LOC = "F1"  ;
NET "dataina_n(6)" LOC = "F2"  ;
NET "dataina_p(5)" LOC = "F3"  ;
NET "dataina_n(5)" LOC = "F4"  ;
NET "dataina_p(4)" LOC = "G5"  ;
NET "dataina_n(4)" LOC = "F5"  ;
NET "dataina_p(3)" LOC = "E1"  ;
NET "dataina_n(3)" LOC = "E2"  ;
NET "dataina_p(2)" LOC = "E3"  ;
NET "dataina_n(2)" LOC = "E4"  ;
NET "dataina_p(1)" LOC = "D1"  ;
NET "dataina_n(1)" LOC = "D2"  ;
NET "dataina_p(0)" LOC = "E5"  ;
NET "dataina_n(0)" LOC = "E6"  ;

# miscellaneous pins
```

```
#Global reset
NET "rstin" loc = "L19" ;
#Error indicator
#NET "error_output" loc = "T19";
#NET "clkrx_locked_out" loc = "T21";
NET "clktx_locked" loc = "K20";

NET "error_output_ch0" loc = "U18";
NET "error_output_ch0_1" loc = "T21";
NET "error_output_ch0_2" loc = "T19";
#Sync output to test pad
NET "sync_out" loc = "U20";

#NET "power" loc = "U20" ;
#NET "dcmtxlock" loc = "T19";


#Locs representing the origins of the 8 4 bit receiver macros

set "rx0/rx0/hset" rloc_origin = "X0y72" ;
set "rx0/rx1/hset" rloc_origin = "X0y64" ;
set "rx0/rx2/hset" rloc_origin = "X0y56" ;
set "rx0/rx3/hset" rloc_origin = "X0y48" ;
set "rx0/rx4/hset" rloc_origin = "X0y40" ;
set "rx0/rx5/hset" rloc_origin = "X0y32" ;
set "rx0/rx6/hset" rloc_origin = "X0y24" ;
set "rx0/rx7/hset" rloc_origin = "X0y16" ;


#Locs for the 8 block RAMs used in the 8 4 bit receiver macros

INST "rx0/rx0/block_ram" LOC = "RAMB16_X0Y9"  ;
INST "rx0/rx1/block_ram" LOC = "RAMB16_X0Y8"  ;
INST "rx0/rx2/block_ram" LOC = "RAMB16_X0Y7"  ;
INST "rx0/rx3/block_ram" LOC = "RAMB16_X0Y6"  ;
INST "rx0/rx4/block_ram" LOC = "RAMB16_X0Y5"  ;
INST "rx0/rx5/block_ram" LOC = "RAMB16_X0Y4"  ;
INST "rx0/rx6/block_ram" LOC = "RAMB16_X0Y3"  ;
INST "rx0/rx7/block_ram" LOC = "RAMB16_X0Y2"  ;

#Specifying the location of the 8 DCMs
inst "dcm_rxclk0" loc = "dcm_x0y1" ;
inst "dcm_rxclk1" loc = "dcm_x1y1" ;
inst "dcm_rxclk2" loc = "dcm_x2y1" ;
inst "dcm_rxclk3" loc = "dcm_x3y1" ;
inst "dcm_rxclk4" loc = "dcm_x0y0" ;
inst "dcm_rxclk5" loc = "dcm_x1y0" ;
inst "dcm_rxclk6" loc = "dcm_x2y0" ;
inst "dcm_rxclk7" loc = "dcm_x3y0" ;

# Specifying the location of the global clock buffer used
# The selection of global clock buffers must obey the
# rules for primary/secondary clock buffer access to the
# 4 quadrants of the FPGA
```

```
inst "rxclk_bufg0" loc = "bufgmux7p" ;
#inst "rxclknot_bufg0" loc = "bufgmux6s" ;
inst "rxclk_bufg1" loc = "bufgmux6s" ;
#inst "rxclknot_bufg1" loc = "bufgmux4s" ;
inst "rxclk_bufg2" loc = "bufgmux5p" ;
#inst "rxclknot_bufg2" loc = "bufgmux2s" ;
inst "rxclk_bufg3" loc = "bufgmux4s" ;
#inst "rxclknot_bufg3" loc = "bufgmux0s" ;
inst "rxclk_bufg4" loc = "bufgmux3p" ;
#inst "rxclknot_bufg4" loc = "bufgmux7s" ;
inst "rxclk_bufg5" loc = "bufgmux7s" ;
#inst "rxclknot_bufg5" loc = "bufgmux4p" ;
inst "rxclk_bufg6" loc = "bufgmux6p" ;
#inst "rxclknot_bufg6" loc = "bufgmux2p" ;
inst "rxclk_bufg7" loc = "bufgmux5s" ;
#inst "rxclknot_bufg7" loc = "bufgmux0p" ;


#inst "dcm_clk" loc = "dcm_x2y1" ;
#inst "clk_bufg" loc = "bufgmux0s" ;
```

G.1 Floor plan view of the FPGA designs

Appendix H

Test Report

20 pages