Clock and Data Recovery over Optical Links and Networks

A MEng Project Final Report

Ammar Bin Shaqeel Ahmed

ammar.ahmed.16@ucl.ac.uk

16080322

University College London

Supervisor:
Dr. Georgios Zervas

Secondary Assessor:
Dr. Domaniç Lavery

May, 2020

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc blandit tellus id lectus egestas, sit amet finibus diam eleifend. Nulla nulla felis, hendrerit vel mi ac, tempus varius libero. Proin lobortis sapien eget malesuada aliquet. Nulla ac eleifend velit. Suspendisse feugiat magna convallis, consectetur leo a, ultrices ex. Aliquam suscipit mi eu tempor porttitor. In facilisis eget massa ac iaculis. Sed dolor lacus, scelerisque in diam vitae, finibus aliquam lorem. Donec eu leo eu lorem sodales varius. Donec dictum mauris sed ornare porttitor. Nunc imperdiet eu elit non hendrerit. Sed velit lectus, lacinia sed dolor vitae, faucibus eleifend felis. Aliquam ac ex enim. Aliquam erat volutpat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque pulvinar sollicitudin mattis.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc blandit tellus id lectus egestas, sit amet finibus diam eleifend. Nulla nulla felis, hendrerit vel mi ac, tempus varius libero. Proin lobortis sapien eget malesuada aliquet. Nulla ac eleifend velit. Suspendisse feugiat magna convallis, consectetur leo a, ultrices ex. Aliquam suscipit mi eu tempor porttitor. In facilisis eget massa ac iaculis. Sed dolor lacus, scelerisque in diam vitae, finibus aliquam lorem. Donec eu leo eu lorem sodales varius. Donec dictum mauris sed ornare porttitor. Nunc imperdiet eu elit non hendrerit. Sed velit lectus, lacinia sed dolor vitae, faucibus eleifend felis. Aliquam ac ex enim. Aliquam erat volutpat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque pulvinar sollicitudin mattis.

Duis vitae malesuada tellus, malesuada tincidunt sem. Nam dapibus mattis lorem, sagittis egestas risus ullamcorper a. Etiam finibus, libero non hendrerit scelerisque, nisl nisi blandit lacus, id laoreet tortor turpis sit amet sapien. Nullam non lacus ut tortor consequat auctor. Vivamus congue, massa in suscipit imperdiet, dui neque ultricies purus, ut fermentum sapien ante vel urna. Sed at blandit ligula. Sed non enim ante. Integer volutpat vestibulum leo, eu malesuada diam lacinia vel. Nulla ultrices leo eu cursus feugiat. Morbi semper nisl ex. Etiam volutpat consectetur sodales. Vestibulum facilisis dapibus enim, ac rutrum massa blandit vehicula. Aenean nec lectus varius, congue risus at, volutpat purus. Duis id rhoncus felis.

Contents

Chapter 1

Introduction

Bandwidth demands in data centers have been doubling every 12-15 months. For data center providers to keep pace with the increased demand (at the same price point) network switches have had to double their capacity while staying at roughly the same cost [1]. However this trend seems to be coming to an end for two reasons. The first is a predicted increase in the rate of growth of demand, due to trends like hardware accelerated programming and dis-aggregated workloads. The second is because electrical switches are predicted to reach a limit due to the physical limits on pin density [2].

For these reasons optical switching is being explored, as it has the potential to overcome many of these problems. Optical switches do not require opto-electrical (OEO) conversion, and hence the number of expensive and power hungry transceivers required is reduced. Furthermore, as buffering is not needed, the latency of the optical switches is much lower. Lastly, they do not use electronics for switching, thus bypassing the aforementioned physical limit [2].

In data centers much of the traffic that is transmitted between servers is in the form of small data packets, with 97.8% of packets being 576 bytes or less [3]. With 100 Gb/s ports this means that switching should take place on the order of hundreds of nanoseconds.

When data is transmitted without a clock signal, the clock has to be regenerated at the receiver before the data can be decoded - this is known as clock and data recovery (CDR). The time taken for the local clock to "lock" to the data stream, adds latency. In optical switches physical links are created between each transceiver-receiver pair. Hence each time the switch is reconfigured, the CDR must re-lock to the new link. This means that the network throughput is limited by the sum of the optical switching time and the CDR locking time - which can be hundreds of nanoseconds in the worst case and tens of nanoseconds in the best case [4]. Assuming an optical switching time of 1 nanosecond, it is evident that the CDR locking time acts as bottleneck that can drastically reduce the throughput [5].

In a source synchronous system the clock is transmitted alongside the data, removing the CDR locking time. This would remove the bottleneck, theoretically increasing the throughput.

Chapter 2

Theoretical Basis

2.1 Background Theory

Here we go deeper into the theory of certain elements of the system.

Bang-Bang CDR

Commonly a serial data stream is sent over a channel without a clock signal. Clock and Data Recovery (CDR) is the process of extracting timing information from a serial data stream, then using it to decode the received data stream. A CDR circuit has two primary functions. The first is to extract a clock based on the input data, and the second is to resample the data.

To extract the clock from the data, a local clock is generated, then is adjusted as "early" or "late" when compared with the incoming data signal [6]. We can think of this as a control system, as shown in Figure 2.1.

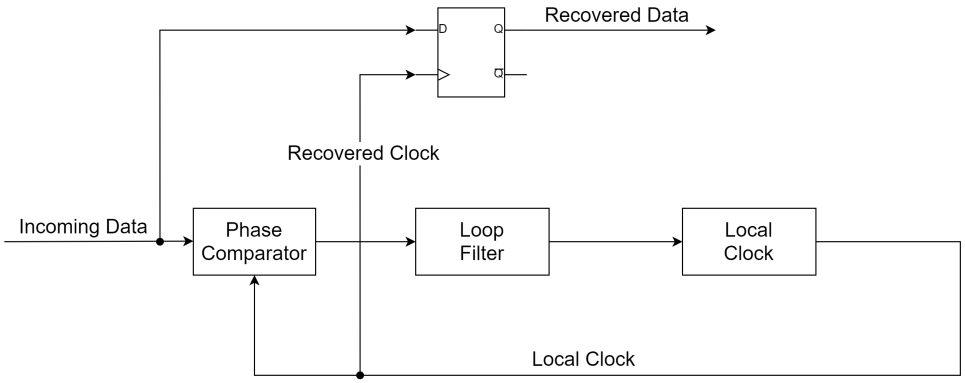

Figure 2.1: Basic CDR design

Phase detectors can be divided into two types, linear (where the output has a linear relationship to the input) and binary or bang-bang phase detectors (where the output is either positive or negative). Binary phase detectors are more commonly used in digital CDR circuits [7]. An example of one is the Alexander detector [8] which gives out a high D0+ and a low D0- if the clock lags and vice-versa if the clock leads, as shown in Figure 2.2.

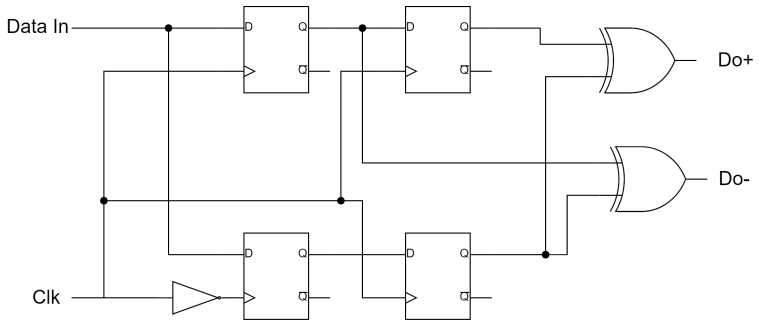

Figure 2.2: Alexander Phase Detector

Pseudorandom Binary Sequence

A pseudorandom binary sequence (PRBS) is a sequence of bits that appears to be random. However as it is generated using a deterministic algorithim, it can be replicated if the inital conditions are the same.

A common practical implementation of PRBS generation uses linear-feedback shift registers. As an example, a PRBS-4 sequence could be generated by using a 4 bit register. We seed the register with a non-zero number, then tap two bits of the register as an input. We then shift the contents of the register, taking the last bit as an output and the new bit as an input, as illustrated in Figure 2.3.

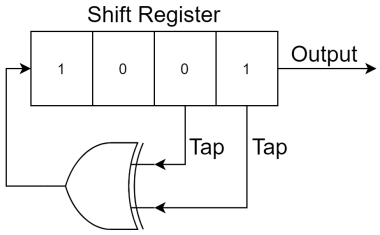

Figure 2.3: Shift Register Implementation

The full operation can be seen in Table 2.1. As 0000 cannot appear (the value of the register would never change) we see that for a register of size N, the bitsequence is $2^N - 1$ bits long.

Source Synchronous System

In a source synchronous system a clock signal is provided alongside the data signal, as shown in Figure 2.4. This has the advantage of not needing a CDR circuit. Furthermore as both the clock and the data come from the same device any jitter will be similar across both signals and can likely be ignored [9]. A downside is that there will be crossing of clock domains at the reciever as the transmitted clock will not be synchronous with the clock domain of the receiving device.

Cycle	Input	Shift Register				Output
0	1	1	0	0	1	1
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	0	1	0	1	1	1
4	1	0	1	0	1	1
5	1	1	0	1	0	0
6	1	1	1	0	1	1
7	1	1	1	1	0	0
8	0	1	1	1	1	1
9	0	0	1	1	1	1
10	0	0	0	1	1	1
11	1	0	0	0	1	1
12	0	1	0	0	0	0
13	0	0	1	0	0	0
14		0	0	1	0	0

Table 2.1: Shift Register Operation

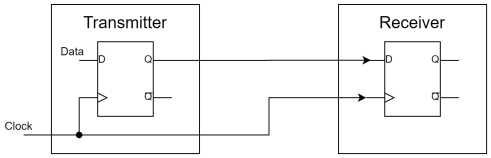

Figure 2.4: Source Synchronous System

Semiconductor Optical Amplifier

Optical amplifiers are devices that can amplify an optical signal without needing to convert it to an electrical one. A silicon optical amplifier (SOA) is one that uses a semiconductor as the gain medium, as light passes through this gain medium it is amplified. SOAs are electrically pumped (do not require the use of another laser) and are of small size.

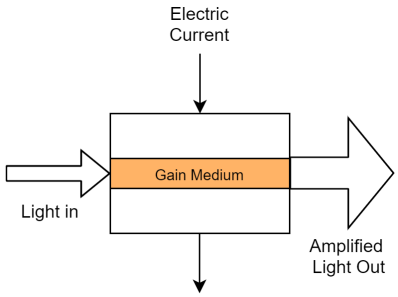

Figure 2.5: Basic SOA Structure

2.2 Literature Review

[5] outlines how CDR circuits are a limiting factor in optical switching and proposes a method of phase caching to overcome this. The data is transferred over the high-speed Xilinx transceivers, and uses a bang-bang CDR. The phase measurments that are being compared are that of the recieved data to the local clock. The PRBS data is pregenerated (written to memory) and is sent in short bursts with a known sequence at the end. When the data arrives it is then written to memory and then processed. The phase caching improved locking time on switching by 12 times.

In [10], [11], and [12], the white rabbit project is discussed. A white rabbit system provides sub-nanosecond synchronisation accuracy. To achieve this, accurate measurements of the link delay between the nodes of the network must be calculated. While instructive, the method is not directly applicable to the project, as in a White Rabbit system, all the nodes are locked to the same frequency. Hence the link delay can be calculated by having a node receive a clock signal from another node, then return the same signal. The link delay can then be calculated by comparing the phase offset of the two signals.

[13] described an optical source synchronous system. It describes how choosing the correct wavelength for the clock can minimise the modal cross-talk. Furthermore, in conjunction with [9] it describes how source synchronous systems are able to track correlated jitter between clock and data channels, and how system performance can be degraded by channel slew between clock and data channels.
[14] further explored reducing the modal crosstalk by proposing an architecture with re-configurable clock and data paths, thus allowing the user to chose the optimal lane for the sensitive clock for each photonic interconnect. This may not be needed however, as each transmitter should have a fixed data characteristic.

[15] and [16] describe fixed latency links. In the event we were unable to bypass the CDR, it may be possible to organise the system to have a fixed latency, then force the CDR to the appropriate fixed phase. Thus the circuit could thus have a much reduced CDR lock time.

[17], [18] describe an Xilinx intellectual property that allows the high speed serial transceivers to be used at much lower data rates. This was initially of interest because it would have been easier to demonstrate a working system with lower data rates. However as this is an extra IP used in conjunction with the transceivers it did not turn out to be useful for the project.

[19] this presentation describes a system where the phase of a transceiver on Xilinx board is kept stable over resets. While this was done on the transmitter side it shows that fixing the phase of the transceiver is possible.

Chapter 3

Proposed System and Objectives

3.1 Proposed System Overview

To demonstrate the efficacy of a source synchronous system we propose a single pseudorandom binary sequence (PRBS) source that optically transmits over two channels to a single reciever. If transmission is alternated the effect is that the reciever would recieve bursts of data from two different channels. If the full PRBS sequence is recieved then the system would be working correctly.
An overview of the system is shown in Figure 3.1.

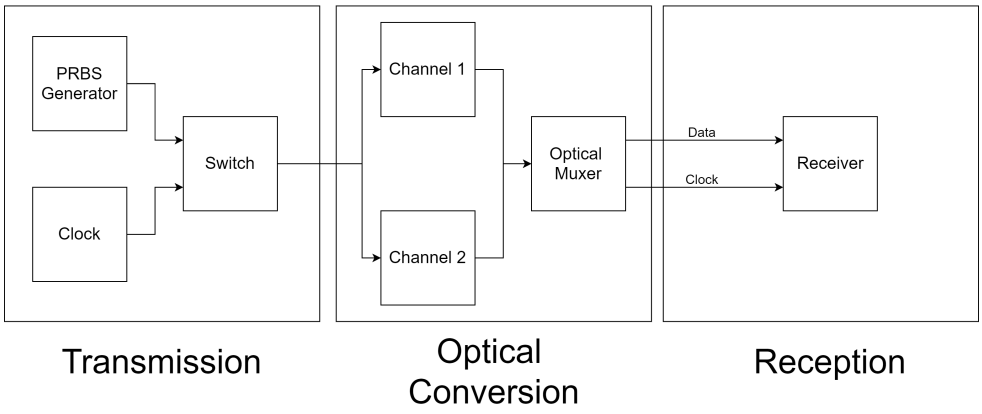

Figure 3.1: Overview of System

3.2 Objectives

The overall objective is to demonstrate successful burst source-synchronous communication for comparison with a system that uses a CDR. Overall we can break down the project to the following sub-objectives:

- Burst mode PRBS transmission over two channels alongside clock
- Convert to optical, then mux the two channels together
- Source synchronous reception of PRBS data

Chapter 4

Implementation and Results

In this section we cover the implementation of the project and the results. As outlined in the Objectives section we can divide the tasks into three main parts: transmission, optical conversion ,and reception. In this project we looked at using a FPGA board for the generation and reception of the PRBS data. Hence the overall design is of a board in a loopback configuration as shown in Figure 4.1.

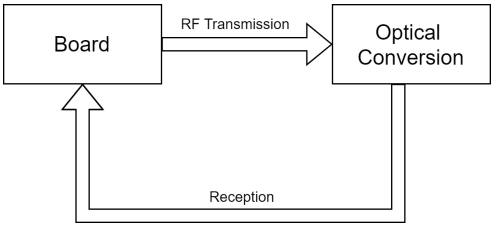

Figure 4.1: Loopback Configuration

4.1 Transmission and Reception

4.1.1 Hardware

To generate and receive PRBS data the VCU118 board was used. The transmission and reception of the data was handled by the onboard high-speed parallel to serial GTY transceivers in conjuction with a Si5345 external clock. To connect with the transceiver the HiTechGLobal FMC-MSMP module was used.

Figure 4.2: VCU118 Board

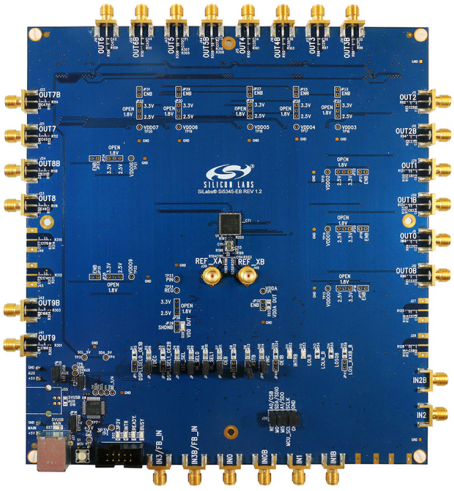

Figure 4.3: Si5345 Clock

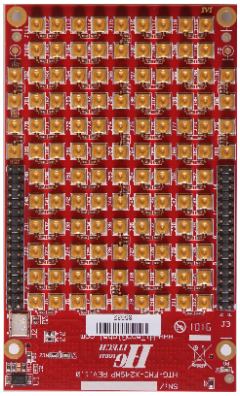

Figure 4.4: FMC-MSMP module

4.1.2 Transceiver Setup

Most of the project took place using the transceivers in a simple RF loopback configuration (no optical transmission).

The setup of the transceivers followed the example design outlined in the user guide to the transceivers [20]. Places where choices were made (as the example design is not specific to a particular board) or the design was deviated from are outlined below.

Selection of Quads

The GTY transceivers in the VCU118 are grouped into four channels or quads. Seven GTY quads on the left side of the device and six GTY quads on the right side of the device. There are 52 transceivers on VCU118 board, in total.

- Four of the GTY transceivers are wired to Samtec Firefly Module Connector
- Four of the GTY transceivers are wired to QSFP1 module connector
- Four of the GTY transceivers are wired to QSFP2 module connector
- Sixteen of the GTY transceivers are wired to the PCIe 16-lane edge connector
- Twenty-four of the GTY transceivers are wired to FMC+ HSPC connector

As the we were using a FMC-HSMP module, we were limited to the transceivers wired to the FMC+ HSPC connector. Furthermore only certain quads can be driven through an external clock - also there are limitations to the number of quads that can be driven with an external clock while still meeting jitter requirments, but as we are only driving two transmitters, it was not an issue. As such we chose Quad 120 and Quad 122 as the quads for testing.

Pin Configuration

For a full pinout diagram, it is necessary to refer to the user guide of the board [21]. However the pin choices made are outlined in the following table.

Transceiver Configuration

The transceivers were programmed to run at a standard rate of 10GB/s, with a reference clock of 156.25 MHz. The free-running clock (used to drive resets) was driven at 125 MHz, sourced from the onboard 300 MHz system clock. The reset functions were bound to the physical push buttons BE23 and BB24. Status indicators (Link Up and Link Down) were bound to LED1 and LED0 respectively.

At the current stage there was no encoding in the setup as we were trying to troubleshoot an issue with the PRBS checking module. However if running the system for a long period of time, encoding would be necessary to ensure the CDR remained locked. The full details of the setup can be found in Appendix A.

4.1.3 Transmitter

In Figure 4.5 we see a block diagram of the transmitter (TX) of the transceiver. Parallel data flows into the transmitter interface, and is seriallised, then finally flows out of the transmitter as high-speed serial data.

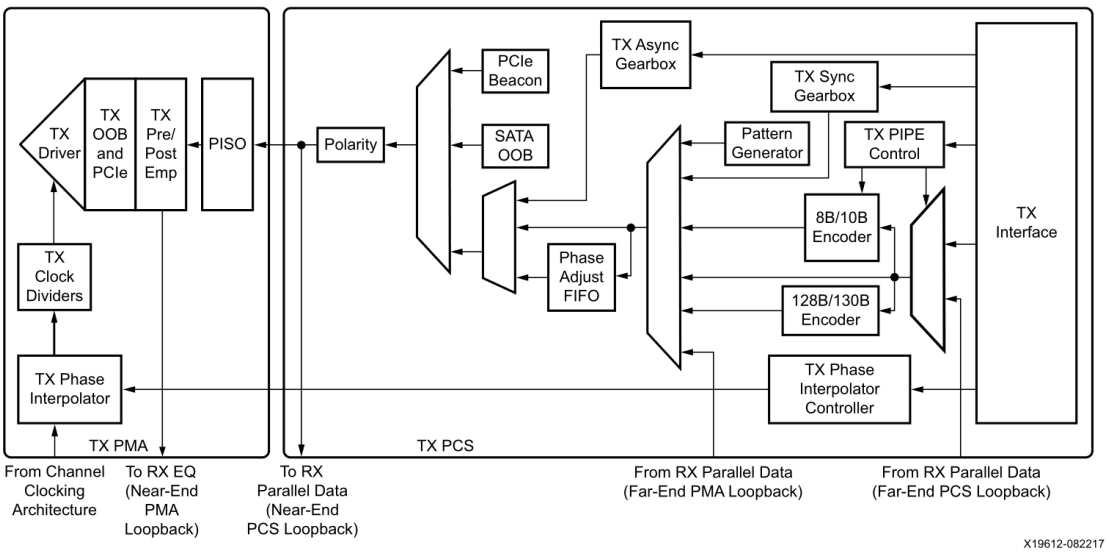

Figure 4.5: Transmitter Block Diagram [22]

We looked to modify the functionality of a basic implementation of the transmitter. In the basic implementation a PRBS generator that generates a sequence through shift registers (as outlined in Section 2.1) is is fed to the transmitter interface.

The PRBS module was mainly unchanged from the default with some minor adjustments. There were two variations of the PRBS generation module that were developed.

Burst Mode over Single Channel

Here we modified the PRBS generation module and set it to output zeros if the enable command was not asserted. In combination we added a 2 bit register inside the wrapper, which on overflow toggles the enable flag. This has the effect of causing the PRBS module to output a sequence interspersed with zeros, as shown in Figure 4.6. We note also that the full sequence is transmitted in burst mode (no data is lost).

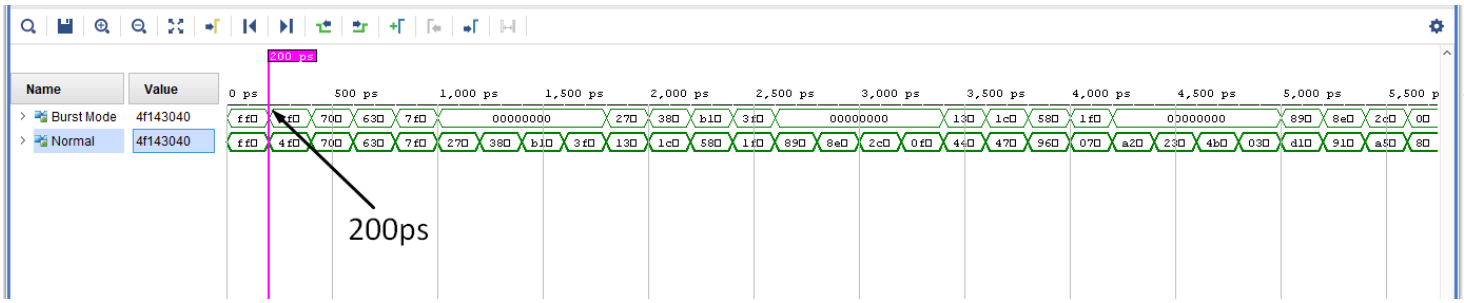

Figure 4.6: Single Burst Mode Behaviour

Comparing Figure 4.7 and Figure 4.8 we see that the recurrence time (in this simulation we use PRBS7, as it is a short sequence) for the normal sequence as compared with the burst mode sequence is about half. This makes sense as the burst mode sequence is operating on a duty cycle of 50% and hence should take roughly double the time to recur.

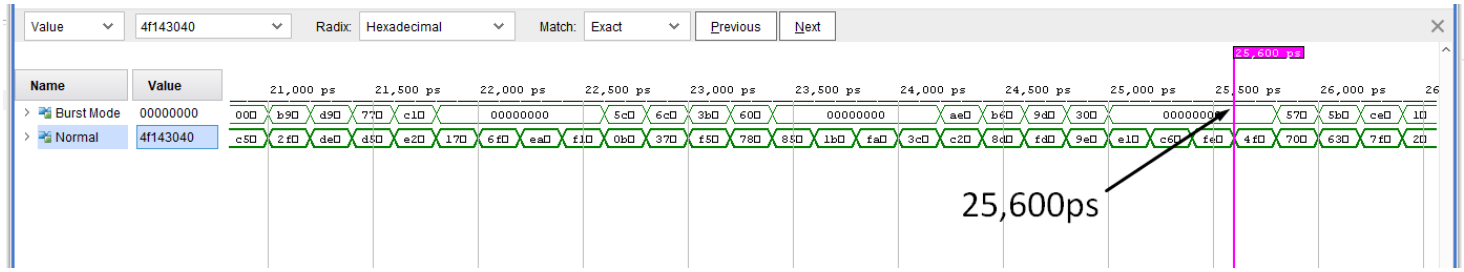

Figure 4.7: Normal Mode Recur Time

Switching Between Two Channels

Here the PRBS wrapper was changed to feed the two different outputs. The PRBS generator module was unchanged. Using a 2 bit register which on oveflow alternated between which of the outputs the PRBS data was sent to, with the other output being sent zeros. This had the overall effect of having the whole sequence be sent over two different channels.

Todo: two channel switch (showing module with two outputs)

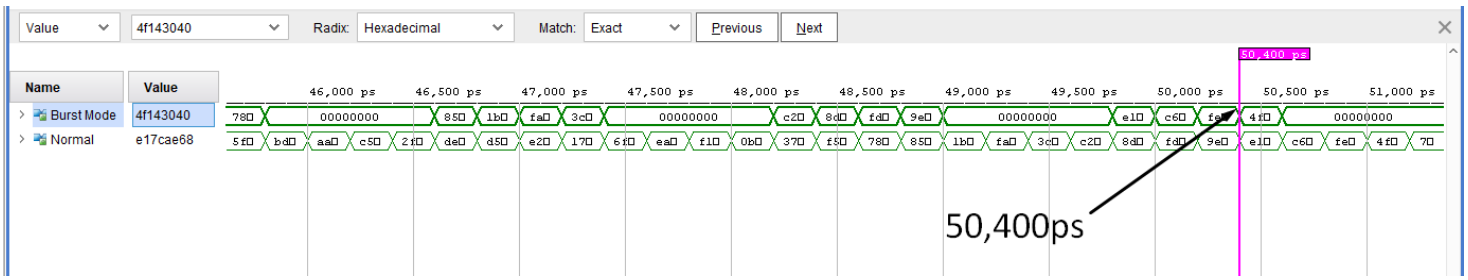

Figure 4.8: Burst Mode Recur Time

4.1.4 Reception

In normal operation the transceiver would parallise the serial data, and then pass the data to the PRBS checking module.

Todo: image of transceiver -> prbs module

Burst Mode Checking

For burst mode checking there are some issues as there are periods when the incoming bitstream is all zeros. The PRBS checker module takes the incoming data as a seed to calculate the next expected sequence. If zeros are provided then this interferes with the module (as the next expected word will be calculated based on zeros). To compensate for this we added a register to the wrapper that would not pass zeros to the checker module. In simulation when the PRBS generation module was connected directly to the PRBS checker this worked correctly, but when passed to through the transceiver, the checker module would throw errors. We were not able to determine why.

Two Channel Checking

In the case where two transmitters were muxed together and were sent to a single receiver, it should not have been necessary to change the behaviour of the PRBS checking module. However we were unable to check this as the lab was closed.

Disabling The CDR

The final step would have been to run the reciever source synchronously. The transceiver did not allow much flexibilty here. We attemped to do this was by disabling the CDR and using the same clock to drive the reciever and transmitter. However the link here was not stable, and we were unable to get phase readouts that may have allowed us to modify the phase of the incoming clock. Overall this was unsuccessful.

4.2 Optical Transmission

This part of the project was not completed as the labs were closed before we were able to test it. However some hardware was prepared, and is described in the following sections.

4.2.1 SOA Board

4 PCBs were prepared. The design consisted of an Inphenix SOA mounted in the center. The electrical component was provided a matched RF line, and the laser was coupled into the sides. A d-type connecter was also provided so that the SOA could be temperature controlled.

4.2.2 Heatsink and Mount

It was also necessary to attach a aluminium substrate underneath the PCB board for heat dissipation.

Chapter 5

Conclusion

Bibliography

[1] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: a decade of clos topologies and centralized control in google's datacenter network," *Communications of the ACM*, vol. 59, no. 9, pp. 88–97, 2016.

[2] H. Ballani, P. Costa, I. Haller, K. Jozwik, K. Shi, B. Thomsen, and H. Williams, "Bridging the last mile for optical switching in data centers," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*. IEEE, 2018, pp. 1–3.

[3] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 78–85.

[4] X. Chen, S. Chandrasekhar, G. Raybon, S. Olsson, J. Cho, A. Adamiecki, and P. Winzer, "Generation and intradyne detection of single-wavelength 1.61-tb/s using an all-electronic digital band interleaved transmitter," in *Optical Fiber Communication Conference Postdeadline Papers*. Optical Society of America, 2018, p. Th4C.1. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2018-Th4C.1

[5] K. Clark, H. Ballani, P. Bayvel, D. Cletheroe, T. Gerard, I. Haller, K. Jozwik, K. Shi, B. Thomsen, P. Watts *et al.*, "Sub-nanosecond clock and data recovery in an optically-switched data centre network," in *2018 European Conference on Optical Communication (ECOC)*. IEEE, 2018, pp. 1–3.

[6] S. Y. Sun, "An analog pll-based clock and data recovery circuit with high input jitter tolerance," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 325–330, 1989.

[7] H. Zhang, S. Krooswyk, and J. Ou, "Chapter 4 - link circuits and architecture," in *High Speed Digital Design*, H. Zhang, S. Krooswyk, and J. Ou, Eds. Boston: Morgan Kaufmann, 2015, pp. 163 – 198. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124186637000046

[8] J. Alexander, "Clock recovery from random binary signals," *Electronics letters*, vol. 11, no. 22, pp. 541–542, 1975.

[9] A. Ragab, Y. Liu, K. Hu, P. Chiang, and S. Palermo, "Receiver jitter tracking characteristics in high-speed source synchronous links," *Journal of Electrical and Computer Engineering*, vol. 2011, p. 5, 2011.

[10] J. Serrano, M. Lipinski, T. Wlostowski, E. Gousiou, E. van der Bij, M. Cattin, and G. Daniluk, "The white rabbit project," *N/A*, 2013.

[11] P. Moreira, P. Alvarez, J. Serrano, I. Darwezeh, and T. Wlostowski, "Digital dual mixer time difference for sub-nanosecond time synchronization in ethernet," in *2010 IEEE International Frequency Control Symposium*. IEEE, 2010, pp. 449–453.

[12] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over ethernet," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009, pp. 1–5.

[13] C. Williams, B. Banan, G. Cowan, and O. Liboiron-Ladouceur, "A source-synchronous architecture using mode-division multiplexing for on-chip silicon photonic interconnects," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 22, no. 6, pp. 473–481, 2016.

[14] C. Williams, D. Abdelrahman, X. Jia, A. I. Abbas, O. Liboiron-Ladouceur, and G. E. Cowan, "Reconfiguration in source-synchronous receivers for short-reach parallel optical links," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[15] K. Chen, H. Chen, W. Wu, H. Xu, and L. Yao, "Optimization on fixed low latency implementation of the gbt core in fpga," *Journal of Instrumentation*, vol. 12, no. 07, p. P07011, 2017.

[16] X. Liu, Q.-x. Deng, B.-n. Hou, and Z.-k. Wang, "High-speed, fixed-latency serial links with xilinx fpgas," *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 2, pp. 153–160, Feb 2014. [Online]. Available: https://doi.org/10.1631/jzus.C1300249

[17] P. Novellini and G. Guasti, *Dynamically Programmable DRU for High-Speed Serial I/O*, Xilinx.

[18] P. Novellini, G. Guasti, and A. Di Fresco, *Clock and Data Recovery Unit based on Deserialized Oversampled Data*, Xilinx.

[19] Eduardo Mendes, "Xilinx transceiver study," URL: https://indico.cern.ch/event/717613/contributions/2948664/attachments/1637690/2613637/hptd_fixed_phase_xcvr_04_18_eduardo_mendes.pdf, 2018.

[20] *UltraScale FPGAs Transceivers Wizard v1.7*, Xilinx.

[21] *VCU118 Evaluation Board*, Xilinx.

[22] *UltraScale Architecture GTY Transceivers*, Xilinx.

Appendix A

Transceiver Settings

The configuration of the Transceiver Wizard can be found here.

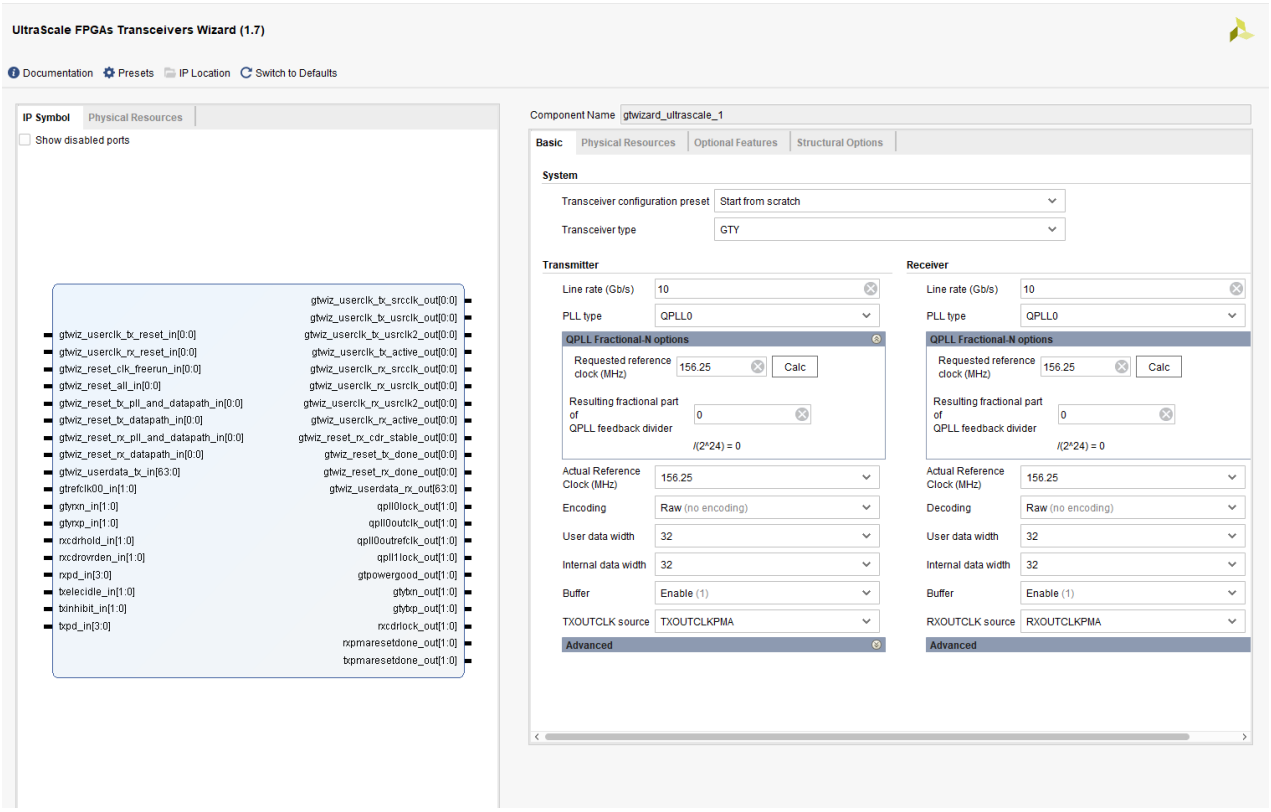

Figure A.1: Transceiver Wizard Settings

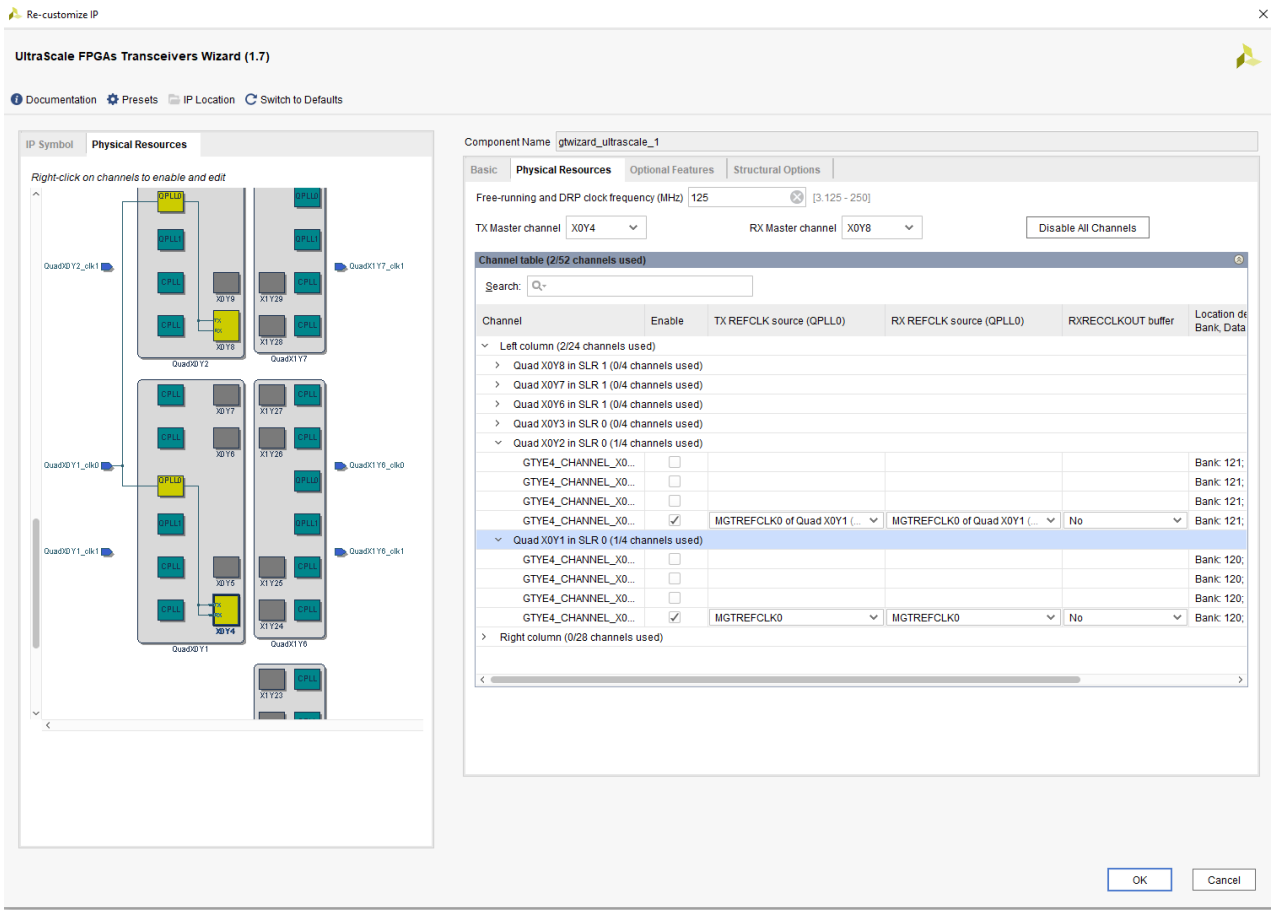

Figure A.2: Transceiver Wizard Settings 2

```
set_property PACKAGE_PIN AN41 [get_ports mgtrefclk0_x0y1_n]
set_property PACKAGE_PIN AN40 [get_ports mgtrefclk0_x0y1_p]

set_property IOSTANDARD DIFF_SSTL12 [get_ports hb_gtwiz_reset_clk_freerun_in_p]

set_property PACKAGE_PIN AY23 [get_ports hb_gtwiz_reset_clk_freerun_in_n]
set_property PACKAGE_PIN AY24 [get_ports hb_gtwiz_reset_clk_freerun_in_p]
set_property IOSTANDARD DIFF_SSTL12 [get_ports hb_gtwiz_reset_clk_freerun_in_n]

set_property package_pin BE23 [get_ports hb_gtwiz_reset_all_in]
set_property IOSTANDARD LVCMOS18 [get_ports hb_gtwiz_reset_all_in]

set_property PACKAGE_PIN BB24 [get_ports link_down_latched_reset_in]
set_property IOSTANDARD LVCMOS18 [get_ports link_down_latched_reset_in]


# LED1 (working correctly)
set_property PACKAGE_PIN AV34 [get_ports link_status_out]
set_property IOSTANDARD LVCMOS12 [get_ports link_status_out]

# LED0 (not working)
set_property PACKAGE_PIN AT32 [get_ports link_down_latched_out]
set_property IOSTANDARD LVCMOS12 [get_ports link_down_latched_out]

# Clock constraints for clocks provided as inputs to the core
------------------------------------------------------------------------------
create_clock -period 8.000 -name clk_freerun [get_ports hb_gtwiz_reset_clk_freerun_in_p]
create_clock -period 6.400 -name clk_mgtrefclk0_x0y1_p [get_ports mgtrefclk0_x0y1_p]

# False path constraints #
------------------------------------------------------------------------------
set_false_path -to [get_cells -hierarchical -filter {NAME =~
*bit_synchronizer*inst/i_in_meta_reg}]
set_false_path -to [get_cells -hierarchical -filter {NAME =~
*reset_synchronizer*inst/rst_in_*_reg}]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*D -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_meta*}]]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*PRE -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_meta*}]]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*PRE -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_sync1*}]]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*PRE -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_sync2*}]]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*PRE -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_sync3*}]]
set_false_path -to [get_pins -filter REF_PIN_NAME=~*PRE -of_objects [get_cells
-hierarchical -filter {NAME =~ *reset_synchronizer*inst/rst_in_out*}]]


set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub] set_property
C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub] connect_debug_port dbg_hub/clk
[get_nets hb_gtwiz_reset_clk_freerun_buf_int]
```

Listing 1: Constraints