Chapter 1

Loopback Test using the GBT Quartus Example

1.1 120 MHz Reference Clock

To be able to conduct a proper loopback test, the Multi-Gigabit Transceiver (MGT) and the Phase-Locked Loops (PLLs) of the GigaBit Transceiver (GBT) example design must have a input clock frequency of 120 MHz. There are a number of ways to achieve this on the Cyclone V board:

- Using an external clock, like a square wave signal generator.
- Using one of the onboard programmable oscillators.
- Implementing a PLL into the design that multiplies the onboard 50 MHz global clock up to the desired frequency of 120 MHz.

The original approach is to use an external signal generator with differential output to generate the reference clock, as shown in the GBT tutorial videos [**?**]. However, at the time of conducting the first tests, there was no available signal generators that could generate a 120 MHz square wave clock for the experiment. Because of this, some time was spent to investigate how to use the internal programmable oscillator as a reference clock.

The approach of implementing an extra PLL into the GBT-example design was also investigated, but attempts of doing so resulted in conflicts between the already implemented transceiver PLLs in the design.

The below sections gives the following descriptions:

- How to configure the onboard oscillator on the Cyclone V GT board for use as reference clock (1.1.1).
- How to configure the $Si5338$ external oscillator for use as reference clock (1.1.2).

1.1.1 Configuring the onboard Oscillator on the Cyclone V Board

To achieve a reference clock of 120 MHz without an external clock, the Field-Programmable Gate Array (FPGA) has an onboard programmable oscillator; the $Si570$ from Silabs. It uses Inter-Integrated Circuit (I2C) for serial communication and can be programmed to output frequencies up to 810 MHz $\pm$ 50ppm. To program the oscillator, Altera provides a dedicated software called "Clock Control". The Clock Control software is part of the Java based "Board Test System" software, included in the Cyclone V kit which can be found at Altera's websites [**?**]. The Cyclone V kit is board specific, so it is therefore important to use the right kit with the right board.

To make use of the Clock Control software has proven to be difficult, mainly because of the software being outdated in relevance to the current version of Quartus (at the time of writing, Quartus 15.0 is the newest edition). The solution was to install an older Quartus (version 13.1) and specify the right paths for the related environment variables. See appendix **??** for a description on how to setup the clock control software using Windows.

1.1.2 Configuring the Si5338 External Oscillator

For an external clock, the $Si5338$ evaluation board was used (see figure 1.1). It has an onboard 25 MHz crystal oscillator in addition to six SMA connectors reserved for external input clocks. For simplicity, the onboard oscillator was used in the thesis experiments. The remaining eight SMA connectors is reserved for differential output clocks. The evaluation board communicates with I2C and connects to the Personal Computer (PC) via Universal Serial Bus (USB) cabling. A dedicated clock builder software [**?**] lets you select clock outputs and type of signaling. For the experiments, a 2.5 V Low-Voltage Differential Signaling (LVDS) clock with 120 MHz was selected, and connected to the Cyclone V GT board using SMA cabling. To enable the SMA connectors as input to the reference clock on the Cyclone V GT board, *CLKSEL* on DIP switch SW4 must be selected ON (default is OFF).

1.2 Testing and Verification of the HDMI Daughter Card

To verify a working Printed Circuit Board (PCB), the High-Definition Multimedia Interface (HDMI) daughter card underwent a series of tests, as summarized in the following sections.

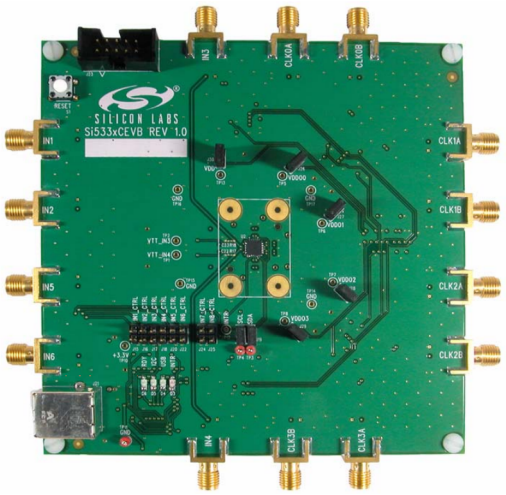


Figure 1.1: The *Si*5338 evaluation board [**?**].

1.2.1 Connectivity Test

Since all the components on the PCB were hand soldered (with the exception of the ground pads underneath the High-Speed Mezzanine Card (HSMC) contact), it was particularly important to check for accidental shorts between pins and/or pads.

Purpose of Test

- Verify that there are no shorts between the pins and/or pads of the PCB.
- Check for current draw to confirm that there are no short on the power-lines.

Experimental Setup

The setup involves the use of a multimeter to probe all pins and check for connection faults according to the schematic. After all shorts have been eliminated, the PCB is to be connected to an external power supply to verify current-draw.

Results

Using a multimeter, all connections were checked and verified that there were no shorts (Some pins on the HSMC were indeed shorted and had to be re-soldered). The PCB was then connected to an external power supply, with a 3.3 V output and a current output limited to 100 mA. It was verified that the current draw, with all HDMI-connections left open, were no more than the current drawn from the power-LED and the 1.25 V voltage divider, i.e around 40 mA.

To confirm connectivity and that there were no further connecting shorts between the pads and/or pins, a simple test-circuit was written in Quartus. Beginning with the transmit-signals: all the relevant LVDS transmit-signals that are physically connected to the HSMC (port A) connector of the FPGA were connected to a given clock signal. The PCB was then connected to the Cyclone V board, and the HDMI connectors were probed with the help of an oscilloscope and verified that there was an output signal on every HDMI-transmitter.

The PCB is now ready for connection with the host FPGA for further testing of the transmitter and receiver signals.

1.2.2 External Loop-back Test for the Fiber-Optic Connector

To verify that the HDMI daughter card SFP-connector for fiber-optic communication is working correctly, an external loop-back test was conducted.

Purpose of Test

- Test the dedicated SFP-connector on the HDMI-daughter card for fiber-optic communication and see that it is capable of sending and receiving information at speeds corresponding the GBT-standard of 4.8Gbit/s.

Experimental Setup

A fiber-optic cable connected from the transmitter to the receiver using a fiber-module, forming an external loop through the cable, was connected to the SFP-connector of the HDMI-daughter card PCB. Using the GBT Quartus-example together with the In-System Source And Probe Editor (ISSP) and SignalTap II, it was possible to perform a pattern check test by comparing the transmitted signals with the received signals. To achieve this test, ISSP was used to setup the pattern generator of the GBT example to count with increments of one (PATTERN SELECT = "1h") and the receiver to receive signals through external cabling (LOOPBACK = '0'). SignalTap II was used to monitor and verify that the receiver line received the same incremented counter that the transmitter sent out.

Results

Running the test resulted in a continuous stream of bits sent from the transmitter to the receiver. Using SignalTap II as a monitor limits to only observe parts of the transmission, but it was sufficient enough to see that the received sum of bits were indeed incrementing by one each time. The results are comparable with the same test using internal loopback (LOOPBACK = '1'), i.e observing that the receiving end is counting by increments of one. This concludes that the SFP-connector on the HDMI-daughter card works as intended at the desired speed of 4.8Gbit/s.

1.2.3 External Loop-back Test for the HDMI Connectors

Purpose of Tests

- Measure the quality of the signal (eye-diagram) at different frequencies up to 300 MHz and see how reflections affects the signal.
- Measure crosstalk between neighboring signal paths.
- Create a test environment using Quartus II in conjunction with SignalTap II to:
  - See if it is possible to sample the received signals at different frequencies up to 300 MHz.
  - Calculate the bit-error rate at different frequencies up to 300 MHz.

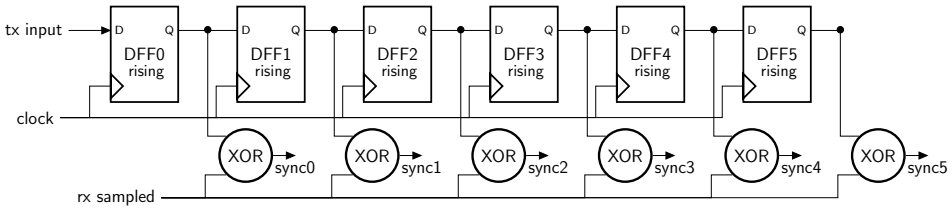
Experimental Setup

Each HDMI-connector has at least one transmitter- and receiver line. To simulate signal transmission over a distance, a HDMI cable was cut in half and the transmit- and receive lines were soldered together, creating an external loop-back. VHDL code was written to simulate a data-stream using a pseudo-random generator to generate random bit-patterns. The data-stream would travel out via one of the transmitters of the FPGA, out through the HDMI-connector, following the cable back into the same HDMI-connector into the receiver input. This would simulate the transmission path to the Versatile Link Demo Board (VLDB) card. The transmitted and received bits would then be compared using xor-logic. A bit-counter register would count each transmitted bit, and a bit-error register would count each time two bits are different in comparison. The bit error rate is thus given by $\frac{bit\ errors}{number\ of\ bits}$.

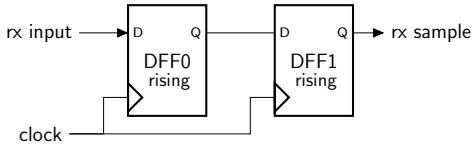
Because of the trace- and cable-lengths and the fact that a signal has a finite propagation time, a delay is introduced between the transmitted and received bit-signals.[1] To cope with the delay differences, a delay chain is added in parallel with the transceiver line: While a bit-signal is travelling through the cable, the same bit-signal is delayed through a series of D Flip-Flops (DFFs). For each clock cycle, the value at the receiver input is compared with the output of each DFF using XOR-logic. The individual outputs of the XORs are connected to Light-Emitting Diodes (LEDs). The LED that has the least amount of toggling is the one XOR, and thus selected DFFs, that is most synchronized with the receiver (ideally, it should be toggle-free), and can be used to count bit errors. A smaller delay-chain was also added at the receiver line to synchronize the incoming bits with the clock. Figure 1.2 a and b illustrates these delay chains.

Cables with different lengths were used to see how this would affect the bit-error rate. This would thus introduce varying delay differences between the transmitter

[1] As mentioned in chapter **??**, a signal propagates through the conductor at a velocity of approximately 15 mm/ns.



(a) Delay chain on the transmitter line.



(b) Delay chain on the receiver line.

Figure 1.2: Transmitter and receiver delay chains.

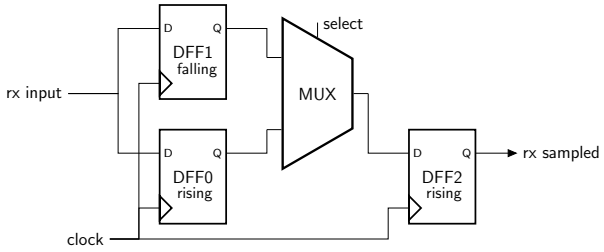


Figure 1.3: Muxed delay chain on the receiver line.

and receiver, and would therefore produce different results comparing the signals in the delay chain. A series of attempts were therefore made to find the most synchronized outputs of the delay-chains at different cable lengths.

Results

During the first samplings using SignalTap II, it was suspected that when running at a 300 MHz clock, the received signal were being sampled when in a metastable state, i.e in the middle of the rising edge of the bit. To compensate for this, a new delay chain was designed on the receiver line (figure 1.3): two DFFs that would trigger on different clock edges (rising and falling), were placed in parallel as the first stage of the receiver delay chain. By using a MUX, one of the DFF outputs were selected to go into a third DFF, and the output of this third DFF was then compared with all outputs in the transmitter delay chain. The "Rising/Falling DFF" tab in table 1.1 states which of the clock edges that caused less toggling on the first delay stage on the receiver.

With the help of SignalTap II it was possible to sample the delay chain signals and produce the results displayed in table 1.1.

Clock [Mhz]	sync#	Signal path [cm]	Rising/Falling DFF	Comments
100	1	110	Rising	Some spikes occur.
200	2	110	Rising	No spikes.
300	4	110	Both	No spikes.
100	1	220	Rising	Some spikes occur.
200	3	220	Rising	Some periodical spikes.
300	5	220	Rising	No spikes.

Table 1.1: SignalTap II measurements on the J10 HDMI connector of the HDMI daughter card. "sync#" is the most synchronized XOR-output in the delay chain. Length is the approximate traveling path (cable and trace) for the signal. The "Rising/Falling DFF" tab shows which edge triggered DFF that worked best as the first stage at the receiver delay chain.

1.2.4 Conclusion and Discussions

When doing the external loopback test for the HDMI connectors (1.2.3), while sampling the LEDs using SignalTap II, some "spikes" occurred at the one LED that seemed toggle-free when observing it, and thus what seemed the most "in sync". The "spikes" were narrower than the 300 MHz clock, and seemed to occur randomly. Whether this is a interference or will affect the error-counter hasn't been confirmed, as the test was aborted due to an accident with the FPGA. The accident caused two of the pins on one of the current regulators on the FPGA board to short, resulting in a burned regulator. This rendered the FPGA useless and halted all remaining tests. Due to time constraints, the FPGA was not repaired in time for the delivery of the thesis.

What remains of this test is to use the most synchronized XOR-output for an error-count measurement over a certain time interval to calculate the bit error rate. Further investigation is also needed to see whether this way of synchronizing the transmitter and receiver produces credible error-counts, or if this approach is altogheter useless.

When measuring a high-speed signal, the way you measure the signal can have a major impact on the result. This was experienced when attempting to measure the LVDS signals using the differential probes. On the first attempt, small clamps connected to each of the differential probes (figure 1.4) were coupled to a differential pair that were running from one end of a hdmi-cable, with a terminating resistor in between; in to a transmitter on the HDMI-daughter card. A clock signal was sent from the FPGA throught the HDMI-daughter card to the end of the hdmi-cable.

The measurement resulted in signals that were heavily distorted by reflections. It was first thought that these reflections might originate from the traces on the FPGA

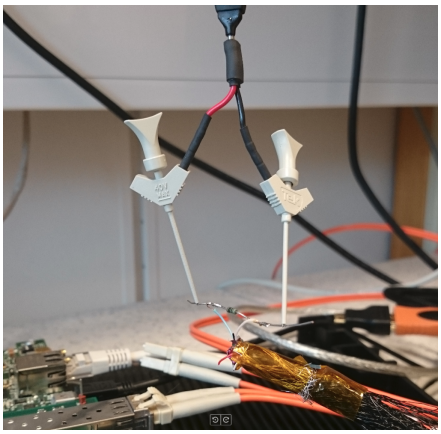
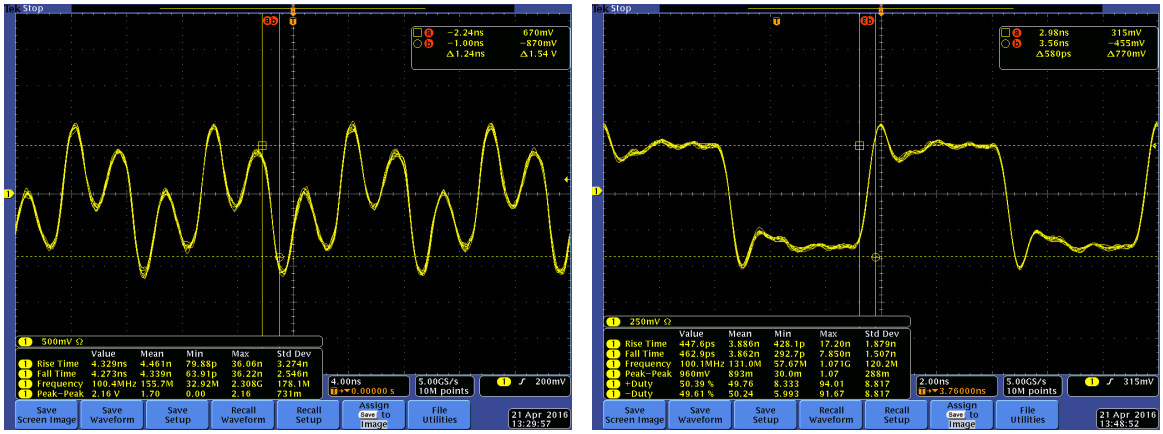


Figure 1.4: Small clamps caused huge reflections when measuring the high-speed signal.

itself, since the same results were produced when sending the clock signals through a different PCB (a GPIO-card) in place of the HDMI-daughter card. However, the theory was quickly rejected when using a completely different FPGA board produced the same reflections. The cause was in fact due to the small clamps that was used to connect the differential probes. By replacing these with small stubs and redo the measure, the result was quite different, as shown in figure 1.5.



(a) Using clamps. **(b)** Using stubs.

Figure 1.5: 100 MHz transmitter signal measured with differential probes.

By doing the same measurements again on different probe-points (viases) and with both the GPIO-card and the HDMI-daughter card on both available FPGAs, the reflections remained somewhat the same. It was concluded that the most notable signal reflections is caused by the measurement setup, and will not affect the digital data when transmitting or receiving signals. However, measuring signal integrity

and creating an eye-diagram could not be done with this measurement setup. This lead to the loop-back test using SignalTap II in an attempt to sample the signals (see chapter 1.3).

1.3 Testing and Verification of the Serial Interface

1.3.1 Hardware Simulation using Testbench in Modelsim

To verify that the hardware design was working properly in terms of correct signal timing between the baudrate, UART and decoder, a testbench was developed. The testbench was made using the Bitvis Utility Library and the design simulation was done using Altera's Modelsim.

Purpose of Tests

- Verify correct timing between the baudrate generator, clock and uart.
- Verify correct timing for the uart decoder.
- Verify that the design is working correctly in simulation.

Bitvis Utility Library

The *Bitvis Utility Library* by Bitvis is an open source VHDL testbench infrastructure library for verification of FPGAs and ASICs [**?**]. It was developed with the aim of simplifying the testbench development process when testing VHDL designs. It was chosen for this test because of the ease of use and fast implementation (see uart_tb.vhd for testbench implementation).

Experimental Setup

The main testbench process goes as follows: Send a request byte to the rx-signal, followed by a register-address. The request can be a read-request (0xDD) or a write-request (0xEE for '1' and 0xFF for '0'), and the address must be between 0x00 - 0xC1. Then, the tx-signal are sampled and compared with the requested address byte and checked for equality. If it is not equal, the test goes to a halt with a corresponding fault-message. The *UART_WRITE_BYTE* and *UART_READ_BYTE* testbench procedures were written with this in mind. To simulate an incoming byte at the receiver, *UART_WRITE_BYTE* inputs a vector of 8 bits as the data-byte. With the period of a transmitted bit, which is equal to $1/baudrate$, it outputs first a start bit, then the data bits and finally the stop bit. The rx-line takes the output of *UART_WRITE_BYTE* as input. *UART_READ_BYTE* takes the tx-line as input, and with the period of a transmitted bit shifts the tx-value into a data vector. Using

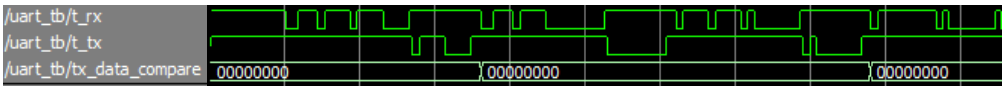


Figure 1.6: Three read operations followed by a write.

the Bitvis *check_value*-function, the data vector is then checked and compared with the address-byte previously sent to the receiver. The *UART_READ_BYTE* procedure must be executed after the *UART_WRITE_BYTE* procedure.

Other test procedures involves checking the constants found in *uart_gbt_pkg.vhd* up against legal values, and also make sure that the tx- and rx-lines are in an idle state at the start of the test, i.e a high state.

Results

Figure 1.6 shows three read operations followed by a write operation. The *tx_data_compare* vector compares the address received with the address transmitted to confirm that the Universal Asynchronous Receiver/Transmitter (UART) decoder does what it is requested to, i.e send out the correct addresses when receiving a read-request and write a bit-value to the GBT register when receiving a write-operation. The data-bit (Most Significant Bit (MSB) of the transmitted signal) is not included in the *tx_data_compare* comparison, nor is the write operation.

1.3.2 Running the PC software design together with the FPGA hardware design

Check if the FPGA design is responding correctly to the bytes sent from the C-program on the PC side.

Purpose of Tests

- Verify that the C-program is in control of the COM port.
- Verify that there is communication between the FPGA UART and the PC COM port.
- Confirm that the FPGA design and C-program is working together the way they should.

Experimental Setup

The Cyclone V Gt FPGA was programmed with the hardware design (**??**) and connected to the user PC using a USB-to-RS232 cable with a 5 V voltage converter between the FPGA and the cable. The software was granted PC admin access so

that it was allowed access to the COM port. The PC COM port itself was configured in device manager (Windows) with a baud rate corresponding to the baud rate set in the hardware design and software.

Results

The software on the PC side was able to communicate with the FPGA via the COM port, and was able to read the registers as well as write to them. In some occurrences the software did not receive all addresses it requested a read on, resulting in some values not being read as often as others. To fix this, the repeat-functionality was implemented into the software (see **??**), and this fixed the problem.

While testing the hardware module together with the send/receive module, when instructing the UART to continuously send data from the GBT registers (by sending read-requests using the software), the UART would stop responding after a random period of time. To make it respond again, the UART had to be hard-reset. The reason why this is happening is not known, but it might be caused by a possible design flaw. A quick fix to this, however, was to simply implement a timer that would reset the UART unit on the condition that no bytes arrives in a given amount of time when it is in the idle state; this time period are larger than the time it takes for two bytes to arrive. This is not a permanent fix, since there is a chance that the UART might reset while a byte is under transmission. This might give an explanation as to why the C-program in some cases did not receive all the data it requested during transmission.

To further investigate this fault, one could use SignalTap II to probe the UART signals and identify and correct irregular behaviour in relevant signals. Due to time constraints, the quick fix was preferred.

Chapter 2

Summary and Conclusion

The future upgrade of the Large Hadron Collider accelerator, the High-Luminosity LHC, has its goal of increasing the beam luminosity by ten times. The GigaBit Transceiver ASICs and transmission protocol was developed with this in mind, and provides a high radiation tolerant, high speed, optical transmission line capable of simultaneous transfer of readout data, timing and trigger signals in addition to slow control and monitoring data. This thesis has had its focus on designing a control interface for the off-detector part of the GBT, the Common Readout Unit, along with a PCB that connects the CRU to the GBTx chip to allow for testing of the chip detector. The PCB was developed with high-speed transmission in mind, allowing for 4.8 Gbit/s optical transmission with minimum reflections using an SFP-contact. In addition, it has 10 HDMI-contacts allowing for e-link interface to the GBTx with a 320 Mbit/s transmission rate. The control interface was written in the C-language, using freely available libraries. A library was also written for the purpose of this thesis, with the GBT control signals in mind (Appendix **??**). A hardware module was implemented in the FPGA to allow the software to read and write information to the GBT probe and source registers. The software consists of two modules: the sending and receiving module, and the ncurses command interface. Both modules work more or less as they should: The send/receive module writes a pattern to the GBT-register in the FPGA and reads it back to the terminal, and the interface module allows you to perform write- and read-commands and monitor the GBT control signals. What remains is to fully integrate the send/receive module into the interface module.

When attempting to measure the signal integrity directly from the traces on the PCB, the measurement equipment caused major reflections, making the measurements unreliable. The reflections remained constant when measuring at different probe points, but varied when changing the probe-tips for the differential probes. External loopback tests were conducted on the HDMIs and SFP-contact using the GBT example design. The SFP-contact had a fiber-optic cable arranged in a loopback manner. Using the GBT example design, a counter was transmitted and received with the same increment at 4.8 Gbit/s, confirming a working SFP-contact.

The HDMI connection were tested by having a HDMI cable cut in half an soldering the transmitter and receiver pairs in a loopback manner. A pseudo-random signal was sent through the cable and a delay chain attempted to synchronize the two signals so that an bit errors could be counted and thus determine the bit error rate. The delay chain was completed, but because of an accident that lead to a damaged FPGA board, the bit error-count test was not conducted. Other remaining tests includes measure of crosstalk between paths and determining the signal integrity using eye-diagram.

The communication hardware on the FPGA side was tested and approved by writing a testbench using the Bitvis Utility Library. Correct timing between the UART and the decoder were confirmed, and the design was confirmed to be working in simulation. The send/receive software module were tested by connecting the COM port of the PC to the FPGA using a USB-to-RS232 cable, and by running the custom hardware on the FPGA, the software was able to communicate with the FPGA and was able to read the GBT probe and source registers as well as write to them.

The hardware module is not yet implemented into the GBT example design itself. During testing, dummy registers were therefore used instead of the actual GBT registers. The module should be integrated by removing the ISSP module in the GBT example design and connecting the leftover probe and source registers signals to the hardware modules dummy register signals.