Chapter 1

External and Internal Loopback Test of the GBT Bank 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) must have a input clock frequency of $120\ \mathrm{MHz}$. 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 $50\ \mathrm{MHz}$ global clock up to the desired frequency of $120\ \mathrm{MHz}$.

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 $120\ \mathrm{MHz}$ 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 setup the onboard oscillator on the Cyclone V GT board for use as reference clock (1.2).
- How to setup the $Si5338$ external oscillator for use as reference clock (1.3).

1.2 Configuring the onboard Oscillator on the Cyclone V Board

To achieve a reference clock of $120\ \mathrm{MHz}$ 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 $810\ \mathrm{MHz} \pm 50ppm$. 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To make use of the Clock Control software has been proven difficult, mainly because of the software being outdated in relevance to the current version of Quartus (at the time of writing, Quartus 15.0 is the newest edition).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) using the Windows Operating System (Linux was also attempted, but without any success) 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.

1.3 Configuring the Si5338 External Oscillator

For an external clock, the $Si5338$ evaluation board was used (figure 1.1). It has an onboard $25\ \mathrm{MHz}$ XTAL 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 $2.5\ \mathrm{V}$ Low-Voltage Differential Signaling (LVDS) clock with $120\ \mathrm{MHz}$ 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).

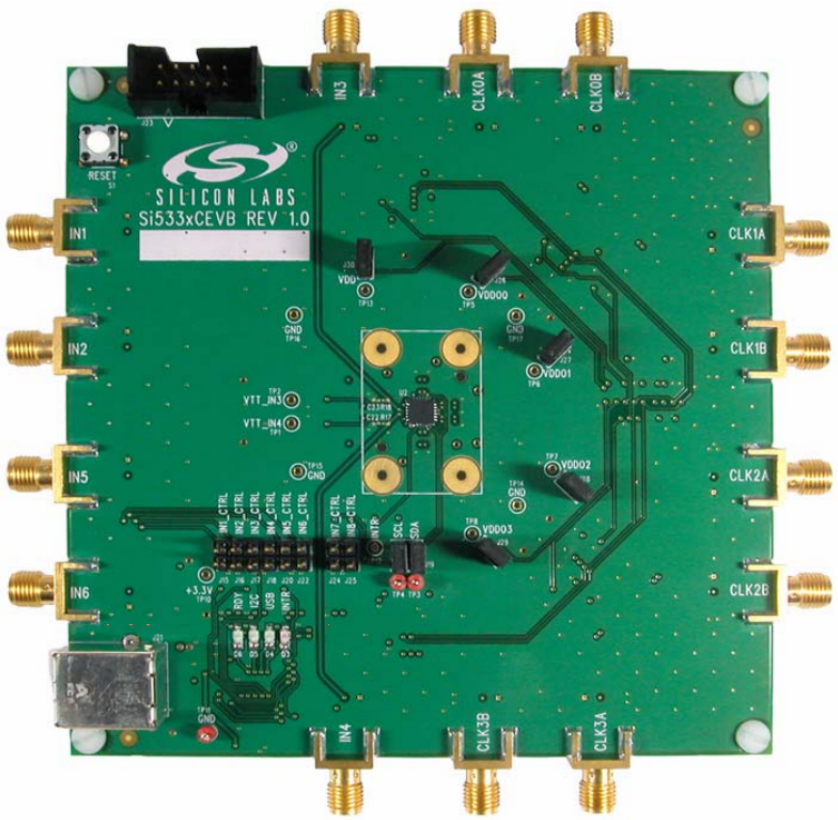


Figure 1.1: The $Si5338$ evaluation board [**?**].

Chapter 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. The sections below summarizes these tests.

2.1 Connectivity Test

Since all the components on the PCB was 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.

2.1.1 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.

2.1.2 Experimental Setup

The setup involves the use of a multimeter to go through 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.

2.1.3 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 $3.3\ \mathrm{V}$ output and a current output limited to $100\ \mathrm{mA}$. 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 $1.25\ \mathrm{V}$ voltage divider, i.e around $40\ \mathrm{mA}$.

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.

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.

2.2.1 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 $4.8\mathrm{Gbit/s}$.

2.2.2 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.

2.2.3 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. This concludes that the SFP-connector on the HDMI-daughter

card works as intended at the desired speed of $4.8\mathrm{Gbit/s}$. Figure **??** shows the resulting incremented sum of bits transmitted and received at the receiving end.

2.3 External Loop-back Test for the HDMI Connectors

2.3.1 Purpose of Tests

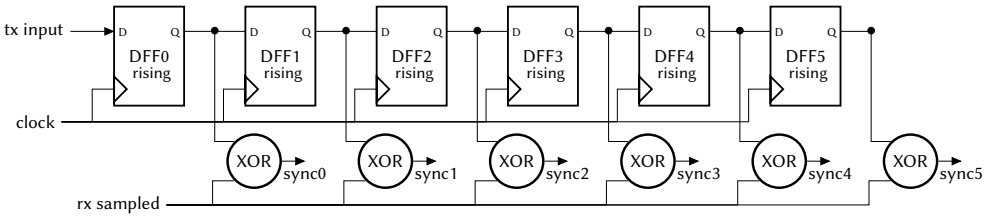
- Measure the quality of the signal (eye-diagram) at different frequencies up to $300\ \mathrm{MHz}$ and see how reflections affects the signal.
- Measure crosstalk between neighboring signal paths.
- Create a test environment using Quartus II in conjunction with SignalTap II to:
  - See if it is possible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

2.3.2 Experimental Setup

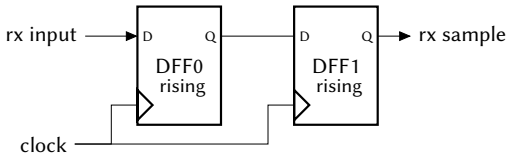
Each HDMI-connector has at least one transmitter- and receiver line. To simulate signal transmission over a distance, a HDMI cable was cut in half and the transmit- and receive lines were soldered together, creating an external loop-back. VHDL code was written to simulate a data-stream using a pseudo-random 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 (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 DFF that has the least amount of toggling is the one 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 2.1 a and b illustrates these delay chains.

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



(a) Delay chain on the transmitter line.



(a) Delay chain on the receiver line.

Figure 2.1: Transmitter and receiver delay chains.

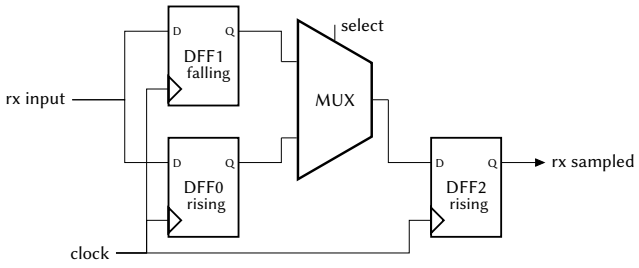


Figure 2.3: Muxed delay chain on the receiver line.

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 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.

To measure the quality of the LVDS signals with an oscilloscope, differential probes were used.

2.3.3 Results

During the first samplings using SignalTap II, it became apparent that, when running at a $300\ \mathrm{MHz}$ 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, 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. This is shown in figure 2.3.

With the help of SignalTap II it was possible to sample the delay chain signals and produce the results displayed in table 2.1.

Clock [Mhz]	In-sync DFF	Signal path [cm]	Rising/Falling DFF	Comments
100	1	110	Rising	Some spikes occur, but are small enough to not be detected. More spikes using a falling edge DFF.
200	2	110	Rising	No spikes.
300	4	110	Both	No spikes.
100	1	220	Rising	Some spikes occur, but are small enough to not be detected.
200	3	220	Rising	Some periodical spikes.
300	5	220	Rising	No spikes.

Table 2.1: SignalTap II measurements on the J10 hdmi connector of the HDMI daughter card. DFF is the most synchronized output in the transmitter 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.

2.4 Conclusion and Discussions

When measuring the quality of 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 2.4) 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 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 2.5.

By doing the same measurements again with both the GPIO-card and the HDMI-daughter card on both available FPGAs, the reflections remained somewhat the same. It is 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. This lead to the loop-back test using SignalTap II in an attempt to sample the signals (see chapter 3).

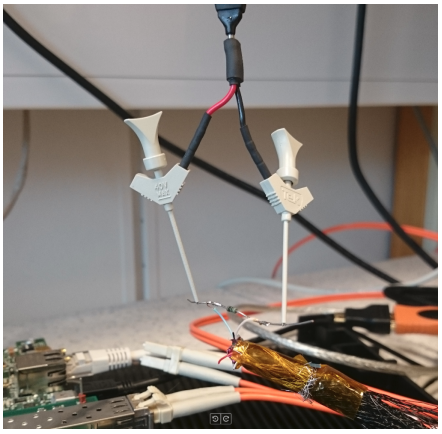
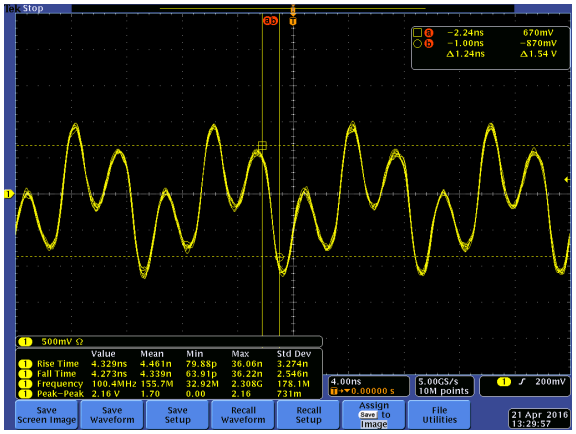
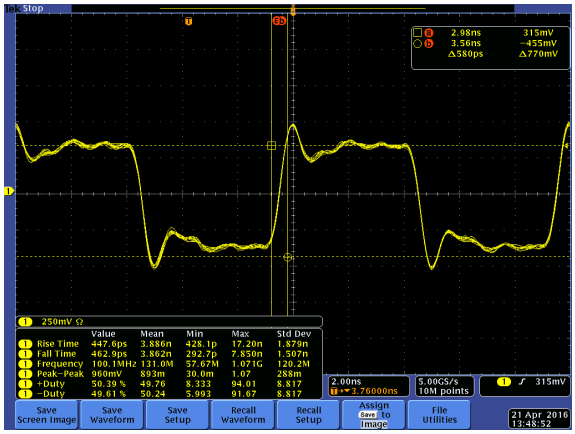


Figure 2.4: Small clamps was first used to measure the high-speed signal.



(a) Using clamps.



(b) Using stubs.

Figure 2.5: 100 MHz transmitter signal measured with differential probes.

Chapter 3

Testing and Verification of the Serial Interface

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.

3.1.1 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.

3.1.2 Experimental Setup

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).

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

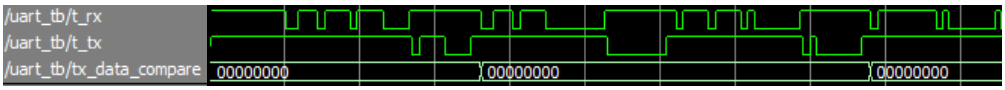


Figure 3.1: Three read operations followed by a write.

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 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.

3.1.3 Results

Figure 3.1 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 comparison, nor is the write operation.

3.2 Connection between COM port and UART using SignalTap II

By using SignalTap II to tap the transmitter and receiver lines, it is possible to verify that the FPGA design is responding correctly to the bytes sent from the C-program on the PC side.

3.2.1 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.

3.2.2 Experimental Setup

3.2.3 Results

While testing the FPGA design, it occured that the UART would hang after a random period of time. The reason as to why this was happening has yet to be found. A quick fix to this, however, was to simply implement a timer that would reset the UART on the condition that no bytes have arrived in a given amount of time when the UART is in the idle state. 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.

3.3 Conclusion and Discussions

Chapter 4

Conclusion and Discussion