Optimizing latency in Xilinx FPGA implementations of the GBT

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2010 JINST 5 C12017

(<http://iopscience.iop.org/1748-0221/5/12/C12017>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 137.138.124.142

The article was downloaded on 11/07/2011 at 09:50

Please note that [terms and conditions apply](#).

TOPICAL WORKSHOP ON ELECTRONICS FOR PARTICLE PHYSICS 2010,
20–24 SEPTEMBER 2010,
AACHEN, GERMANY

Optimizing latency in Xilinx FPGA implementations of the GBT

S. Muschter,^{a,1} S. Baron,^b C. Bohm,^a J.-P. Cachemiche^c and C. Soos^b

^a*Stockholm University,
106 91 Stockholm, Sweden*

^b*Cern,
1211 Geneva 23, Switzerland*

^c*CPPM,
13288 Marseille, France*

E-mail: muschter@fysik.su.se

ABSTRACT: The GigaBit Transceiver (GBT) [1] system has been developed to replace the Timing, Trigger and Control (TTC) system [2], currently used by LHC, as well as to provide data transmission between on-detector and off-detector components in future sLHC detectors. A VHDL version of the GBT-SERDES, designed for FPGAs, was released in March 2010 as a GBT-FPGA Starter Kit for future GBT users and for off-detector GBT implementation [3]. This code was optimized for resource utilization [4], as the GBT protocol is very demanding. It was not, however, optimized for latency — which will be a critical parameter when used in the trigger path. The GBT-FPGA Starter Kit firmware was first analyzed in terms of latency by looking at the separate components of the VHDL version. Once the parts which contribute most to the latency were identified and modified, two possible optimizations were chosen, resulting in a latency reduced by a factor of three. The modifications were also analyzed in terms of logic utilization. The latency optimization results were compared with measurement results from a Virtex 6 ML605 development board [5] equipped with a XC6VLX240T with speedgrade-1 and the package FF1156. Bit error rate tests were also performed to ensure an error free operation. The two final optimizations were analyzed for utilization and compared with the original code, distributed in the Starter Kit.

KEYWORDS: Optical detector readout concepts; Detector control systems (detector and experiment monitoring and slow-control systems, architecture, hardware, algorithms, databases)

¹Corresponding author.

Contents

1	Introduction	1
2	GBT firmware data-flow	1
3	Latency optimization	2
4	Utilization studies	3
5	Test design and measurements	4
6	Conclusion	5

1 Introduction

The GBT firmware code published by the GBT-FPGA group in March 2010 was designed to allow a quick and simple instantiation of the GBT protocol in both Xilinx and Altera FPGAs. The main structure of the protocol was kept as simple as possible to allow every user to adapt and modify the code according to their needs. The study presented below will at first analyze the GBT firmware data-flow in terms of latency without including the platform dependent transceiver. Next we present latency optimizations, which were also analyzed in terms of utilization. Finally we present results of our tests in a real FPGA environment to ensure error free operation.

2 GBT firmware data-flow

Transmitter. Within the GBT-logic, data is shifted along a chain from one component to the next (figure 1). The data first pass through a scrambling unit to provide DC-balanced frames, and then through a Reed Solomon encoder. The frame structure of the data here corresponds to the SLHC frame [1]. To improve error correction capability an interleaving stage was added after the encoder. This complex encoding scheme eventually allows to recover up to 16 consecutive corrupted bits at the receiver stage [6]. The clock domain crossing from 40MHz to 120MHz is realized by a dual port memory module. An internal control unit makes sure that this multiplexing unit operates properly with 120 and 40 bits. In the last stage the data are serialized using a hard-IP transceiver inside the FPGA, resulting in an overall line rate of 4.8Gbps.

Receiver. A received data stream is first deserialized using the hard-IP transceiver. The results are 40 bit words at a frequency of 120MHz. The frequency for controlling the receiving part of the GBT-logic can be recovered from the incoming data stream and then used to clock the rest of the design. Since the Virtex 6 transceiver does not include an alignment block compatible with the SLHC frame this must be done in separate logic blocks, the frame alignment and pattern search units, resulting in 40 bits of parallel data with the header always at the beginning of one of the three words. The rest of the receiver logic chain works in the same way as its counterparts in the transmitter.

Figure 1. Dataflow chain of the GBT-protocol submitted with the Starter Kit, including numbers of estimated latency in terms of clock cycles. “zero cycle” stages include only unregistered combinatorial logic.

3 Latency optimization

In this study, our aim was to minimize the latency contribution from every single component within the chain. The latency was first estimated by looking at event statements embedded in the VHDL code, and later confirmed by simulation. Analyzing the source code distributed with the Starter Kit resulted in a latency of approximately nine 40MHz cycles for the transmitting chain and eight 40MHz cycles for the receiving chain. Figure 1 shows the estimated latency of the different sections. At the end we achieved two different optimizations. Optimization 1 (Opt. 1) used the same components as the Starter kit but optimized for latency, while Optimization 2 (Opt. 2) replaced some components entirely to bring the latency to the lowest possible limit.

Transmitter. Starting at the beginning of the transmission chain the code of the scrambling unit was optimized for latency, resulting in one 40MHz cycle. The largest latency (seven 40MHz cycles) comes from multiplexing the data from 40MHz to 120MHz (Mux). As described previously this is realized with a dual port memory module, where optimizing the corresponding control-unit can improve the timing performance significantly, leading to two 40MHz cycles (Opt. 1). To decrease this latency to one 40MHz cycle the Mux was replaced with a register controlled by a finite state machine working at 120MHz (Opt. 2).

Receiver. Starting from the deserializer the data passes through the frame alignment and pattern search circuits. Although the pattern search circuit is necessary to ensure alignment, there is no need for the data to pass completely through the circuit. Knowing this, the structure of the data flow was modified as shown in figure 2. Going further into the receiving chain revealed that the demultiplexing from 120MHz to 40MHz (Demux) can be optimized in the same way as the previously described Mux, leading to a latency of two 40MHz cycles (Opt. 1) or one 40MHz cycle (Opt. 2).

Summary. The latency for the modified GBT protocol code, excluding the serializer and deserializer, is roughly seven 40MHz cycles for Optimization 1. In detail, three 40MHz cycles are contributed from the modified GBT transmitter code and three 40MHz cycles plus one 120MHz cycle from the modified GBT receiver code. This optimization does not give the shortest latency but should save logic resources since less registers are used. This is investigated in the next section.

Figure 2. Dataflow chain after optimization, including of latency at each stage for both optimizations.

Figure 3. Utilization studies for transmitter (a) and receiver (b).

The same estimation for Optimization 2 results in roughly five 40MHz cycles. The modified GBT transmitter code contributed two 40MHz cycles and the modified GBT receiver code contributed two 40MHz cycles plus one 120MHz cycle. Apart from the low latency this optimization is also platform independent, since it does not use vendor specific hard-IP cores.

4 Utilization studies

It is important to note that changing the GBT protocol code does not significantly increase the resource utilization. The results of this study are shown in figure 3. As a reference the previously described Virtex 6 with 301,440 slice registers and 150,720 slice look-up-tables is used. On the transmitting side one can see that logic utilization is not increased. Instead the optimized code which uses the memory modules strongly decreases the number of slice registers and slightly reduces the number of slice look-up-tables. Replacing the memories with registers increases the number of slice registers, almost to the original value. The number of look-up-tables is also slightly increased compared to the originally code. Summarizing one could say that the optimization does

Figure 4. Schematic of the clock domains within the test design.

not significantly increase the utilization and actually decreases it for Optimization 1. Looking at the utilization of the receiving side gives a slightly different picture. For the first optimization with dual port memory, the number of slice registers is slightly decreased, whereas the number of slice look-up-tables is slightly increased. Comparing this with the final optimization without memory modules one can see that the number of slice registers increases as expected, while the number of slice look-up-tables decreases. This means that using Optimization 1 provides more balance with respect to the number of complete slices used.

5 Test design and measurements

Different optimized designs were tested for reliability. The test system consisted of a Pseudo Random Number Generator (PRNG) to generate a test pattern, a GBT-transmitter, a GBT-receiver, one Gigabit Transceiver and a comparator, all contained in a Virtex 6 FPGA. After generating the pseudo random pattern, it was transferred to the GBT-protocol and the comparator. From the GBT-protocol logic, the signal was transferred to the transceiver. The output of the Gigabit Transceiver was then externally looped back to the input using coaxial cables with a length of 10 cm that directly connected the output of the transceiver to the input. After receiving the data, the GBT-protocol aligned automatically, reconstructed the received data and send it to the comparator, where the generated and received data were compared. The latency of the whole test chain was determined by observing the time difference between the two data streams.

During the test, the transmission line rate was set to 5Gbps rather than 4.8Gbps. This was necessary due to clocking restrictions of a ML605 development board from Xilinx which was used as one clock source. Therefore the frequencies had to be adapted to 41.667MHz and 125MHz. To provide a test system which was close to a real setup, the clock sources of the transmitter and receiver were completely separated using two independent external oscillators. One of these was situated on the ML605 board which was used for the test system while the other was situated on a Xilinx ML507 evaluation board [7]. This reference clock signal was used to supply the serializer (light green area in figure 4). The 125MHz clock signal for the deserializer came from an on board source (grey region in figure 4), and was only used within the hard-IP receiver of the FPGA. A further clock signal with a frequency of 250MHz was recovered from the incoming data stream. Out of this, 41.667MHz and 125MHz clock signals were synthesized to control the receiving part of the GBT protocol (light yellow region in figure 4). The measurements done with unmodified code

and non-optimized transceivers showed a total latency of twenty-one 41.667MHz clock cycles. After optimization the latency of the test design was eight clock cycles for Optimization 1 and six clock cycles for Optimization 2. This latency includes the serialization and deserialization by the hard-IP transceiver, which in simulation was determined to six 125MHz cycles. Taking this into account, the results were consistent with the latency estimated for the two different optimizations. Each was tested more than two days without a single error. This corresponds to a lower BER limit of 2×10^{-15} with a confidence level of 90%.

6 Conclusion

We have optimized the original GBT-protocol code provided by the GBT-FPGA team for low latency. We produced two different optimizations, one using memory modules (Optimization 1: seven 40MHz cycles) and one using registers (Optimization 2: five 40MHz cycles). The estimated latency values were confirmed in measurements with a Virtex 6 FPGA. The optimizations do not introduce additional resource utilization and both optimizations were extensively tested for more than two days without a single error. Soon the optimization should also be available for download from the GBT-FPGA homepage.

References

- [1] P. Moreira, *GBTx specifications*,
<https://espace.cern.ch/GBT-Project/GBTX/Specifications/gbtSpecsV1.2.pdf>.
- [2] B.G. Taylor, *TTC Distribution for LHC Detectors*, *IEEE Trans. Nucl. Sci.* **45** (1998) 821.
- [3] GBT-FPGA project web site: <https://espace.cern.ch/GBT-Project/GBT-FPGA>.
- [4] S. Baron et al., *Implementing the GBT data transmission protocol in FPGAs*, in proceedings of *Topical Workshop on Electronics for Particle Physics*, September, 21–25, 2009 Paris, France.
- [5] Xilinx Virtex-6 FPGA ML605 Evaluation Kit,
<http://www.xilinx.com/products/devkits/EK-V6-ML605-G.htm>.
- [6] G. Papotti, *Architectural studies of a radiation-hard transceiver ASIC in 0.13 mm CMOS for digital optical links in high energy physics applications*, PhD thesis, January 2007, University of Parma, Italy.
- [7] Xilinx Virtex-5 FPGA ML507 Evaluation Platform,
<http://www.xilinx.com/products/devkits/HW-V5-ML507-UNI-G.htm>.
- [8] *HFTA-05.0: Statistical Confidence Levels for Estimating BER Probability*, MAXIM, October 2000,
<http://notes-application.abcelectronique.com/003/3-5321.pdf>.