Contribution ID: **61** Type: **Oral**

Optimizing latency in Xilinx FPGA implementations of the GBT

Thursday, 23 September 2010 11:25 (25 minutes)

The GigaBit Transceiver (GBT) has been developed to provide data transmission and to replace the Timing , Trigger and Control (TTC) system between on-detector and off-detector components in future sLHC detectors. A VHDL version of the GBT-SERDES, designed for FPGAs, has been released in March 2010 as a GBT-FPGA Starter Kit for future GBT users and for off-detector GBT implementation. This code was optimized for resource, as the GBT protocol is very heavy, but not for latency –which will yet be a critical parameter of the GBT when used in the trigger path. We have investigated different methods for minimizing the latency in Xilinx Virtex 5 and Virtex 6 components.

Summary

The GBT firmware code published by the GBT-FPGA group in the beginning of March was designed to allow a quick and simple instantiation of the GBT protocol in both Xilinx and Altera FPGAs. The code was tested for stable operation under various conditions, and optimized to reduce resource usage. However, latency is also a critical parameter, which is especially important when using the GBT in time critical trigger applications. Minimizing the latency is the goal of this investigation. However, to allow a more optimized design we have restricted ourselves to Virtex 5 and Virtex 6 FPGAs from Xilinx.

We have tested different code modifications to verify their reliability. Our test system consists of a pseudo random number generator to generate a test pattern, a GBT-transmitter, a GBT-receiver, one Gigabit transceiver and a comparator, all contained in a Xilinx FPGA. The output is externally looped back to the input.

The pseudo random pattern is both transferred to the GBT-protocol and the comparator. From the GBT-protocol logic, the signal is directly transferred to the transceiver. There the loop back is realized with a coaxial cable that connects the output of the transceiver with the input. After receiving the data, the GBT-protocol aligns automatically, reconstructs the received data and sends it to the comparator, where the generated and received data could be compared. We determined the overall latency and utilization to get a rough overview about the used resources and the possibilities for improvements.

The initial result we measured was 21 clock cycles. We also run some simulation to get a better understanding of origin of the latency. This showed that the main part of the latency was due to dual port memories used to derive three times 40 bits from one time 120 bits and the other way around. On one hand one could reduce latency by removing the dual port memories, but on the other hand they are useful to ensure a proper crossing between two clock domains, that are not synchronous. For this reason we decided to analyze and optimize both possibilities, with and without dual port memories. The first optimization results showed a 40% latency improvement, from 21 clock cycles down to 12 clock cycles by using the dual port memories, optimizing the counters and the rerouting the data path. By removing the dual port memories we were able to reduce the latency further, down to 10 clock cycles. Under simple test conditions both systems were running stably and without error for more than two days.

There are nevertheless still some more facts to consider, such as errors induced by meta-stability that could result from a different clock management like the one used in our test design. For this reason we have planned to make some more tests regarding to other possible layouts for the clock management and phase differences between the clock domains.

Primary author: MUSCHTER, Steffen Lothar (Stockholm University)

Co-authors: BOHM, Christian (Stockholm University); SOOS, Csaba (Cern); CACHEMICHE, Jean-Pierre (Universite d'Aix - Marseille II); BARON, Sophie (Cern)

Presenter: MUSCHTER, Steffen Lothar (Stockholm University)

Session Classification: Programmable Logic, design tools and methods

Track Classification: Programmable Logic, design tools and methods