Implementing the GBT data transmission protocol in FPGAs

S. Baron [a], J.P. Cachemiche [b], F. Marin [b], P. Moreira [a], C. Soos [a]

[a] CERN, 1211 Geneva 23, Switzerland, [b]CPPM, 13288 Marseille, France

sophie.baron@cern.ch, cachemi@cppm.in2p3.fr, marin@cppm.in2p3.fr, paulo.moreira@cern.ch, csaba.soos@cern.ch

Abstract

The GBT chip [1] is a radiation tolerant ASIC that can be used to implement bidirectional multipurpose 4.8Gb/s optical links for high-energy physics experiments. It will be proposed to the LHC experiments for combined transmission of physics data, trigger, timing, fast and slow control and monitoring. Although radiation hardness is required on detectors, it is not necessary for the electronics located in the counting rooms, where the GBT functionality can be realized using Commercial Off-The-Shelf (COTS) components. This paper describes efficient physical implementation of the GBT protocol achieved for FPGA devices on Altera and Xilinx devices with source codes developed in Verilog and VHDL. The current platforms are based on Altera StratixIIGX and Xilinx Virtex5.

We will start by describing the GBT protocol implementation in detail. We will then focus on practical solutions to make Stratix and Virtex transceivers match the custom encoding scheme chosen for the GBT.

Results will be presented on single channel occupancy, resource optimization when using several channels in a chip and bit error rate measurements, with the only aim to demonstrate the ability of both Altera and Xilinx FPGAs to host such a protocol with excellent performances. Finally, information will be given on how to use the available source code and how to integrate GBT functionality into custom FPGA applications.

I. GBT PROTOCOL PRESENTATION

A. Introduction

The general architecture of a high-speed optical link implemented using the GBT chipset and FPGA is represented in Figure 1.

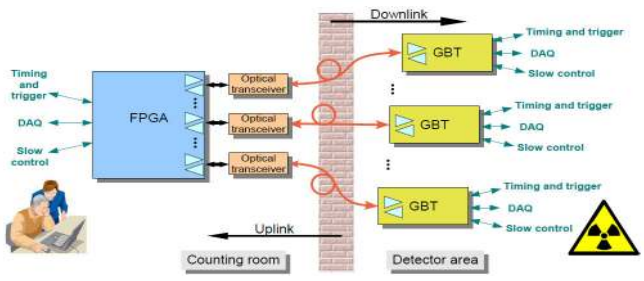

Figure 1: GBT optical link implementation scheme

Logically the link provides three "distinct" data paths for: Timing and Trigger, Data Acquisition and the Slow Control. In practice, the three logical paths do not need to be physically different and are merged. The aim of such architecture is to allow a single link to be used simultaneously for data readout, timing and trigger distribution, readout and experiment control. The link establishes a point-to-point optical bidirectional connection (using two optical fibers).

The GBT chipset [2] is under development to match such architecture. It targets high-speed (3.36Gb/s) data transmission between the detectors and the counting room.

As illustrated in Figure 1, such a link is implemented by a combination of custom and Commercial Off-The-Shelf (COTS) components. In the counting room, the receivers and transmitters will be implemented using COTS components and FPGAs while, embedded on the detectors, the receivers and transmitters will be implemented by the GBT chipset and Versatile Link Components [3]. This architecture clearly distinguishes between the counting room and front-end electronics specificities: that is, the on-detector front-end electronics works in a hostile radiation environment requiring custom made components while the counting room electronics operates in a radiation free environment allowing the use of COTS components. Moreover, the availability of FPGAs with up to 48 Hard-IP serializer blocks would allow concentrating data from several front-end sources into a single module in the counting room facilitating data merging and leading to compact systems.

The study presented below will focus on proving the usability of COTS components and FPGAs to implement the GBT protocol in counting rooms [4].

B. GBT Protocol

Due to the beam luminosity planned for SLHC, the high speed data transmission link will be exposed to high Single Event Upset rates. SEUs are a major impairment to error free data transmission. To deal with this, the GBT line coding adopts a robust error correction scheme that will allow correction of bursts of errors caused by SEUs. A significant fraction of the channel bandwidth must therefore be assigned to the transmission of a Forward Error Correction (FEC) code.

The code to be used must provide a high level of protection, since errors occurring during transmission can also occur as burst errors and not only as isolated events. Because of this, a double interleaved Reed-Solomon correcting code was chosen. The code is built by first scrambling the input data to provide DC-balancing of the frame, and then interleaving two Reed-Solomon encoded words (using 4-bit

symbols), each capable of correcting a double symbol error (Figure 2). The interleaving operation allows increasing the correction capability of errors up to 4 symbols.

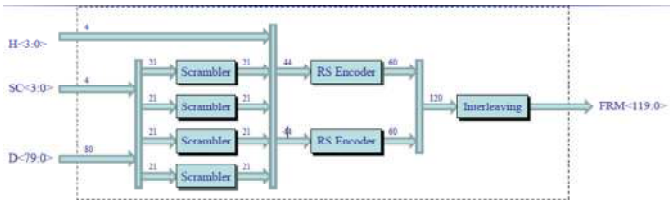

Figure 2: GBT encoding scheme

This in practice means that a sequence of up to 16 consecutive incorrectly-received bits can be corrected. This correction technique requires an extra field of 32 bits in the frame to protect the 88 transmitted bits (including data, header and slow control), resulting in a code efficiency of 73%.

The frame (sketched in Figure 3) is composed of 120 bits that are transmitted during a single SLHC bunch crossing interval (25 ns) resulting in a line data rate of 4.8 Gb/s. Of these, 4 bits are used for the frame Header (H) and 32 used for Forward Error Correction (FEC). This leaves a total of 84 bits free for data transmission corresponding to a user bandwidth of 3.36 Gb/s. In these 84-bits, 4 are always reserved for the Slow Control (SC) field (see 'Slow control channel') and 80-bits are reserved for data (D) transmission. Among the 4-bit of slow control, 2 are reserved for GBT control and 2 are user defined. The 'D' field use is not pre-assigned and can be used indistinguishably for Data Acquisition (DAQ), Timing Trigger & Control (TTC) or Experiment Control (EC) applications [5][6].

This makes 2+80 = 82 bits of data available to the user for a frame of 120bits, giving a payload of 68%.

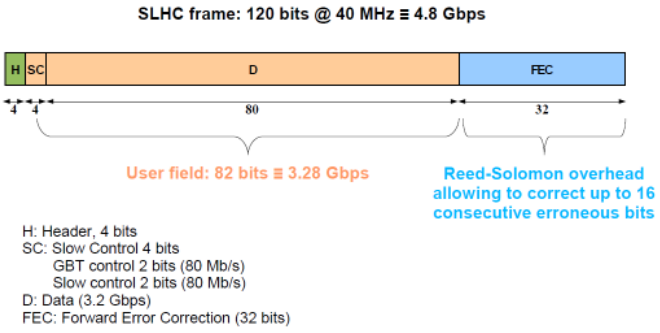

Figure 3: GBT frame

II. GBT Protocol Implementation In FPGAs

A. FPGAs constraints

In the same way it is done in the GBT ASIC, the DC-balance of data transmitted over the optical fiber is ensured by the FPGA by scrambling the data contained in the SC and D fields. For forward error correction the scrambled data and header are Reed-Solomon encoded before nibble interleaving, and serialization. The line encoding/decoding process is represented in Figure 4.

No problem was encountered to configure the hard-IP transceivers of the FPGAs, as the portability of this protocol was carefully checked during the specification phase of the GBT. In particular, the ability of Stratix and Virtex transceivers to transmit 120 bits at a frequency of 40 MHz was ensured at that time.

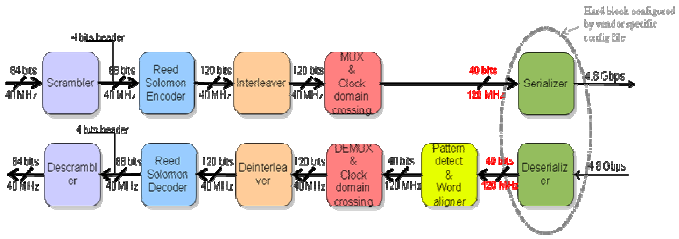

Figure 4: Block diagram of a full GBT link in an FPGA

However, these transceivers provide neither specific encoding schemes like the one we selected nor flexible word alignment functions. This is mainly due to the fact that they target the most common telecommunication protocols. We had thus to implement in user logic all the encoding and decoding blocks, as well as a customized pattern detection and word alignment block (see Figure 5).

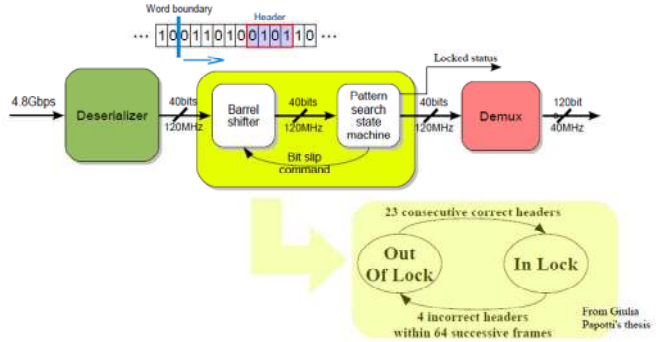

Figure 5: Frame alignment procedure in FPGAs

At power on or after a loss of synchronization, the receiver starts a frame-lock acquisition cycle to find the frame boundaries, that is, to acquire frame synchronization.

The frame-lock acquisition mode operates as follows. In the StratixIIGX, the transceiver hard-IP word aligner block cannot be bypassed. It is thus configured to lock on an arbitrary pattern. Once completed the process is not repeated, except at power on or upon a command from the pattern detection state machine. For all the other devices, we bypass the word aligner inside the transceiver.

The parallel output of the receiver feeds the custom pattern detection and word aligner blocks, which take control of the frame alignment process: for each received frame the four bits in the header position are checked for header validity. Because the header pattern can be found in the data, 23 consecutive frames must contain a valid header before the frame is considered locked (the probability of false boundary detection is then reduced below 10^{-20} as demonstrated in [5]). Otherwise, the frame is shifted by one bit and the valid header checking procedure is repeated. After frame-lock is achieved, the

receiver switches to the frame-tracking mode, which maintains frame synchronization even in the presence of headers corrupted by noise or single event upsets.

The phase tracking mode must thus be tolerant to a low rate of detection of invalid headers. Provided that frame synchronization is maintained, the detection of a corrupted header will not introduce a transmission error since the header field is also protected by the forward error correction code transmitted with the frame. A corrupted header will thus be corrected and properly identified by the Reed-Solomon decoder. The frame tracking mode operates as follows: after a successful frame-lock acquisition cycle has been executed the receiver enters the frame-tracking mode. In this mode the receiver strives to maintain frame synchronization. It checks the validity of the headers and counts the number of invalid headers received in 64 consecutive frames after the first invalid header has been detected. If the number of invalid headers received in 64 consecutive frames is bigger than 4 then the receiver re-enters the frame-lock acquisition mode. Otherwise the receiver resets the count of invalid frames and remains in the frame-tracking mode.

B. *Resource Usage*

The full serializer-deserializer, as described above, was implemented both in a StratixIIGX and in a Virtex5FXT. Besides the transceivers and PLLs, which do not consume any resources as they are hard-coded, a single link consumes 1542 ALMs (Adaptative Logic Modules) for the StratixII and 1481 Slices for the Virtex5.

The table 1 shows the number of links which can be implemented in a selection of StratixIIGX and of Virtex5FXT devices, taking into account the available transceiver blocks and logic elements.

Max usable/available nb of channel	Altera Stratix II GX	Logic cells usage in %
8/8	EP2SGX30D	92%
12/12	EP2SGX60D	78%
16/16	EP2SGX90E	69%
20/20	EP2SGX130G	59%
24/24		

Table 1: Maximum GBT links for StratixIIGX

Max usable/available nb of channel	Xilinx Virtex 5	Logic cells usage in %
3/8	XC5VFX30T	87%
10/16	XC5VFX100T	93%
13/20	XC5VFX130T	94%
20/24	XC5VFX200T	96%

Table 2: Maximum GBT links for Virtex5FXT

Differences of occupancy between Table 1 and Table 2 emphasize the different policies used by Altera and Xilinx in term of ratio between the number of logic cells and the number of transceivers. However, these numbers should be used with care. It is obvious that the occupancy of logic cells is too high if one tries to use all the available transceivers of a chip for GBT protocol implementation. This is tempered by the fact that a design using GBT links will not dedicate all its links to GBT transceivers: some links must be left to output processed data and therefore occupancy will be lower. However, as a back-end FPGA has to dedicate a significant part of its logic to other tasks, optimization of the resources used by the decoding block is a must.

C. *Optimization*

An analysis of the resource usage per block for a single link (see Figure 6) quickly shows that more than half of the logic elements are used by the Reed-Solomon decoder.

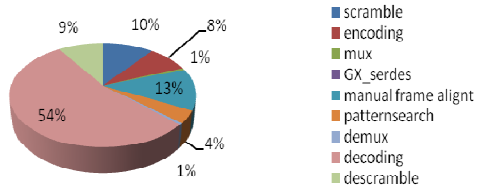

Figure 6: % of ALMs/Slices of one GBT link used by each functional block

It was thus natural to study optimization schemes, particularly for designs hosting several GBT links in one device. The first possibility is to share one decoder block between several links, multiplying its operating frequency by the same factor. The Reed-Solomon decoding algorithm is a large combinatorial circuit, and the maximum operating frequency achieved was 134MHz for the StratixIIGX, applying all the timing optimization constraints available. This allowed to share one decoder block between 3 links.

An analysis of the resources used for 12 links implemented in a StratixIIGX type EP2SGX90 was carried out with and without optimization.

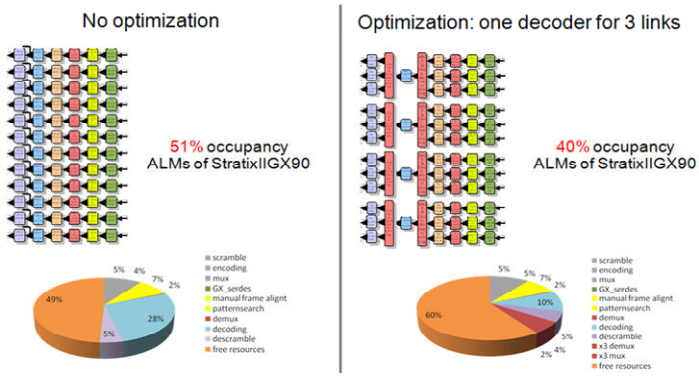

Figure 7: Effect of optimization by 3 on 12 links implemented on a EP2SGX90

As shown in the Figure 7, the device occupancy dropped from 51% of ALMs to 40% thanks to the optimization. Indeed, the fraction of the resources used by the decoder blocks dropped from 28% down to 10%. However, 7% of new logic elements were added due to the resource consuming multiplexers and de-multiplexers required to share the decoder.

This implementation was tested on a PCIe SIIGX development kit with three optimized links using loopback cables mounted on the HSMC connectors. It ran several days without a single error being detected.

The next step for optimization could be to pipeline the decoder algorithm to increase the clock frequency. The drawback of this implementation, beside its complexity, is that it increases the decoding latency.

III. Measurements

A. Setups and equipment

Two evaluation boards were used to implement the GBT protocol on FPGAs. The ML523 (hosting a Virtex5FXT type XC5VFX100T) for Xilinx [8], the PCIe SIIGX Development Kit (hosting a StratixIIGX type EP2SGX90) for Altera [7], both powered by the power supply given in the kit (See Figure 8).

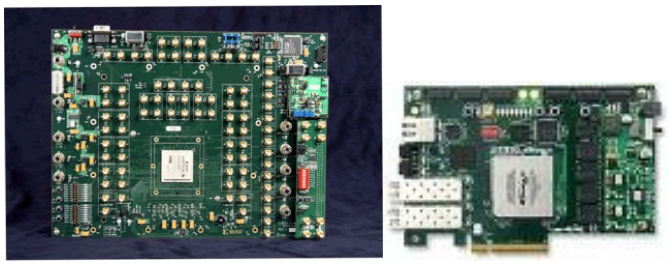

Figure 8: Evaluation platforms. ML523 from Xilinx (left) and PCIe SIIGX from Altera (right)

The reference clock was generated by the J-BERT 4903A from Agilent on differential SMA cables.

For all the qualitative measurements, the very same SFP+ 1300nm optical transceiver module from MergeOptics was used (mounted and dismounted from one board to another). The optical patch cords were 50cm long.

The jitter measurements were made at the optical receiver level with the Lecroy SDA100G sampling scope equipped with 10 GHz optical sampling head.

B. Platform testing

Various platforms and technologies were tested by implementing the GBT protocol in both Altera and Xilinx chips presented above. As described on the Figure 9, a generator instantiated in the Virtex5 was sending parallel data (80 bits @ 40 MHz, either constant words or flying bits) to the encoder and serializer.

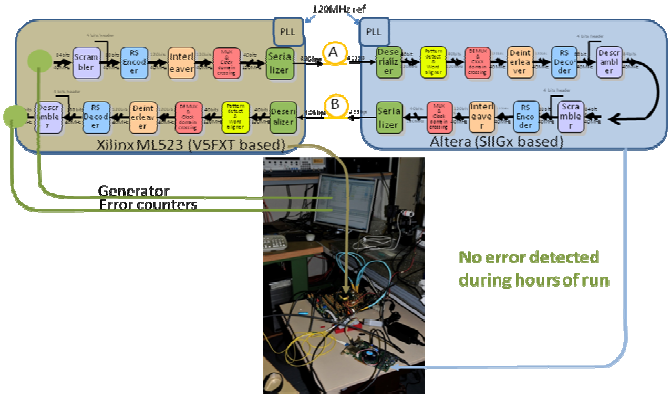

Figure 9: Test setup based on two platforms

The signal (that looks like a PRBS due to the scrambling) was transmitted by an SFP+ to the receiver in the StratixII over a short optical fibre (A). After full decoding (and remote monitoring of the decoded values), the data were encoded back, serialized again and transmitted using another SFP+ module and an optical fibre (B) back to the Virtex5, where it was decoded and compared to the generated words.

We let the system run during several hours without counting any error. Besides providing us an opportunity to implement the GBT protocol on both main technologies, this test allowed us to check the compatibility between the GBT-ASIC protocol and its VHDL translation: the Virtex5 had the Reed-Solomon encoder and decoder implemented in Verilog (the direct copy of the GBT protocol implementation in the ASIC), whereas the StratixII encoder and decoder were implemented in VHDL.

C. Jitter performances

Using the same setup, we measured the jitter out of the two optical fibres A and B in Figure 9. For each of the results below, the SFP+ module transmitting the optical signal was the same (it was successively mounted on A and B fibres to test Xilinx and Altera devices).

As presented in Figure 10, Xilinx and Altera platforms both showed excellent performances. The eyes were widely open, and the total jitter of the order of 80ps PP and 5ps RMS.

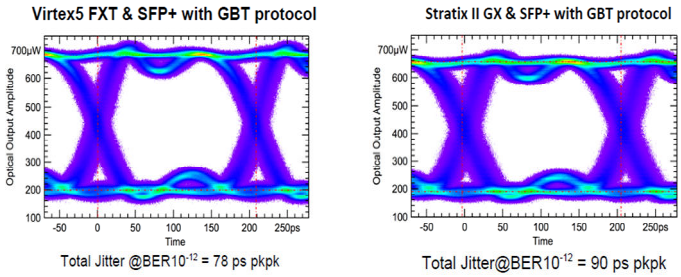

Figure 10: Eye diagrams for Xilinx Virtex5 FXT (left) and Altera StratixIIGX (right)

IV. Source Code Availability

Reference designs of the GBT protocol will be made available before the end of 2009 for both Altera and Xilinx FPGAs. They will be presented as a firmware-based starter kit, downloadable on request via the CERN SVN repository. This starter kit will include the source code for both implementations, and, as much as possible, for various types of devices (StratixII and IV GX, and Virtex5 and 6 FXT) and various flavors of optimization. It will also include documentation.

Basic support will be provided on how to use and optimize the implementation.

V. Conclusion

With this study, we proved that the GBT protocol can indeed be implemented with success both in Altera and Xilinx FPGA chips. The scheme proposed in the introduction where

GBT ASICs are used in detector areas and FPGAs in counting rooms is thus a valid prospect, and the developed code will now be used as a basis to test the GBT serdes chip once it becomes available.

A firmware-based starter kit will be made available upon request to the users. It will be progressively completed by several implementation flavors for StratixIV and Virtex6, and new optimization techniques like a pipelined Reed-Solomon decoder are being considered.

VI. References

[1] GBT project home page:
https://espace.cern.ch/GBT-Project

[2] P. Moreira, GBTx specifications:
https://espace.cern.ch/GBT-Project/GBTX/Specifications/gbtxSpecsV1.2.pdf

[3] F. Vasey, "Versatile Link", ACES 2009 workshop, 3-4 March 2009, CERN, Geneva:
http://indico.cern.ch/contributionDisplay.py?contribId=37&sessionId=22&confId=47853

[4] GBT-FPGA project web site:
https://espace.cern.ch/GBT-Project/GBT-FPGA

[5] 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, University of Parma, Italy, January 2007.
http://papotti.web.cern.ch/papotti/tesi.pdf.

[6] G. Papotti, "An Error-Correcting Line Code for a HEP Rad-Hard Multi-GigaBit Optical Link", 12th Workshop for LHC and future Experiments (LECC 2006), Valencia, Spain, 25-29 September 2006, pp.258-262.
http://indico.cern.ch/contributionDisplay.py?contribId=30&sessionId=19&confId=574

[7] Documentation on Altera PCI express Development Kit, StratixIIGX Edition:
http://www.altera.com/products/devkits/altera/kit-pciexpress_s2gx.html

[8] Documentation on Xilinx Virtex5 FXT ML523 RocketIO GTX characterization Platform:
http://www.xilinx.com/products/devkits/HW-V5-ML52X-UNI-G.htm