EPJ Web of Conferences **214**, 01002 (2019) https://doi.org/10.1051/epjconf/201921401002
CHEP 2018

Front-End Electronics Control and Monitoring for the LHCb Upgrade

Joao Barbosa[1,*], *Federico* Alessio[1,**], *Luis* Cardoso[1,***], *Clara* Gaspar[1,****], and *Paolo* Durante[1,†]

[1]Online Team, EP-LBC, CERN

Abstract. The LHCb experiment, one of the four operating in the LHC, will be enduring a major upgrade of its electronics during the third long shutdown period of the particle accelerator. One of the main objectives of the upgrade effort is to implement a 40MHz readout of collision data. For this purpose, the Front-End electronics will make extensive use of a radiation resistant chipset, the Gigabit Transceiver (GBT), for readout as well as for slow control, monitoring and synchronization. At LHCb, the tools to operate the front-end electronics are developed by a central team and distributed to the users. This contribution describes the architecture of the system that implements the slow control and monitoring of all Front-End electronics using the GBT chipset, namely the GBTx and GBT-SCA. The system is implemented in 3 layers starting with an FPGA based electronic board that interfaces the GBT chipset directly through optical fibers. The second layer is composed by a PCIe driver and a number of processes to operate these boards. The user operates the system in the third layer which is the WinCC OA SCADA that is interfaced with the Front-Ends via a message broker called DIM. The requirements of the system as well as the design and integration of each layer are discussed in detail. The results of the firmware implementation in hardware and operational tests are shown and the overall performance of the system is discussed.

1 Introduction

The LHCb experiment is one of four major experiments operating at the Large Hadron Collider and it consists of a high precision detector that focuses in the detection of decays of particles containing B mesons. During the years of 2019 and 2020, LHCb will be commissioning the new detector and infrastructure that resulted from the upgrade effort [1]. Most of the on-detector and back-end electronics will be replaced. The first-level event triggering will not be performed, and so the readout electronics will have to sustain the full 40 MHz bunch-crossing rate and relay the event information to the Software Trigger. In view of this requirement, the experiment is using the GBT chipset [2] and its related components developed by the EP-ESE group at CERN, for data readout as well as for fast control and slow

*e-mail: joao.barbosa@cern.ch
**e-mail: federico.alessio@cern.ch
***e-mail: luis.granado@cern.ch
****e-mail: clara.gaspar@cern.ch
†e-mail: paolo.durante@cern.ch

control tasks. Furthermore, to perform the slow control tasks, the GBT-SCA ASIC is used to configure and monitor the Front-End electronics (FEE). In LHCb, the FEE are controlled via FPGA-based electronics boards with bidirectional optical links, interfaced to the software control system via a PCIe bus. These boards are commonly referred to as PCIe40 [3], but also as one of the three logic flavours provided for the Upgrade, namely: SODIN, SOL40 and TELL40. While the TELL40 is responsible for the readout of the Front-End event fragments into the DAQ infrastructure, the SODIN plays the role of readout supervisor, distributing the LHC clock, generating synchronous and asynchronous commands to the Front-End and dispatching event data to the event building stage. The SOL40 is an interface board that is responsible for interfacing both TFC (Timing and Fast Control)[4] and ECS (Experiment Control System)[5] with the Front-End boards. It relays timing and clock information sent by SODIN into the optical link to the FEE, while also using the same GBT data frame to give the ECS capabilities to both control and monitor the FEE over the same link. An overview of the role of the three flavours of PCIe40 boards from the ECS and TFC point of view is illustrated in Figure 1. The data acquisition schematic is not evident in this figure.

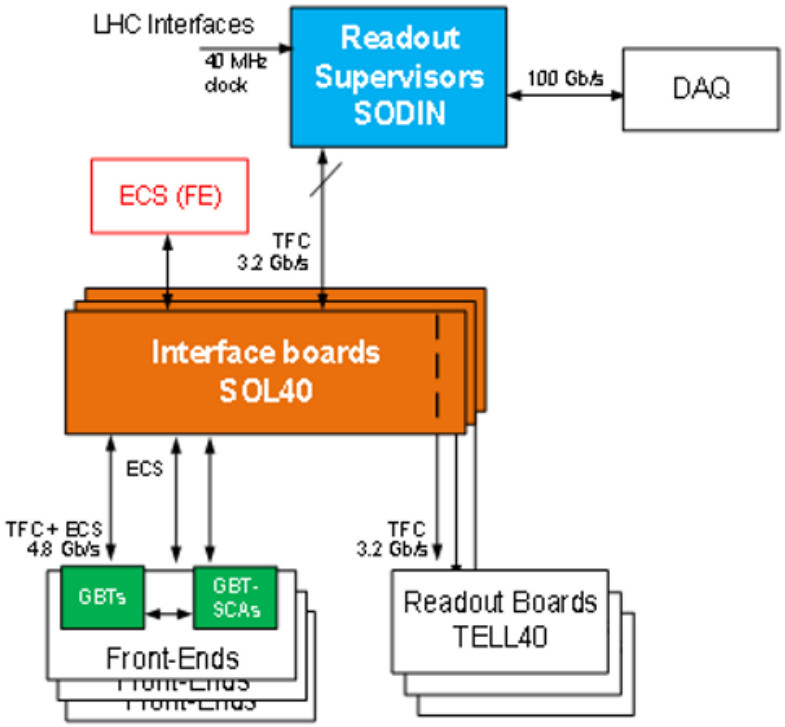

Figure 1. The TFC, ECS and Data acquisition architecture of the PCIe40 boards.

It is estimated that the LHCb experiment will contain around 2500 Master GBTs and GBT-SCAs and around 10000 FE chips. These are controlled from the Run Control system, based in WinCC-OA (the SCADA software used in the experiment), by using as intermediary a software server based on DIM [10] as well as the firmware that manages the FEE configuration and monitoring in the SOL40 board. The firmware consists of a technology agnostic VHDL core, developed to support all buses and protocols of the GBT-SCA chips, while the software is a DIM server that unrolls Run Control requests into firmware instructions. Con-

sequently the firmware unrolls these instructions into GBT-SCA instructions. The interplay of these components is illustrated in Figure 2. This document discusses the architecture of the whole FEE configuration as a whole, explaining how the development of both software and firmware were done in conjunction to so that the CPU could be less strained and most of the low level operations were done in firmware. The overall performance of the system is discussed.

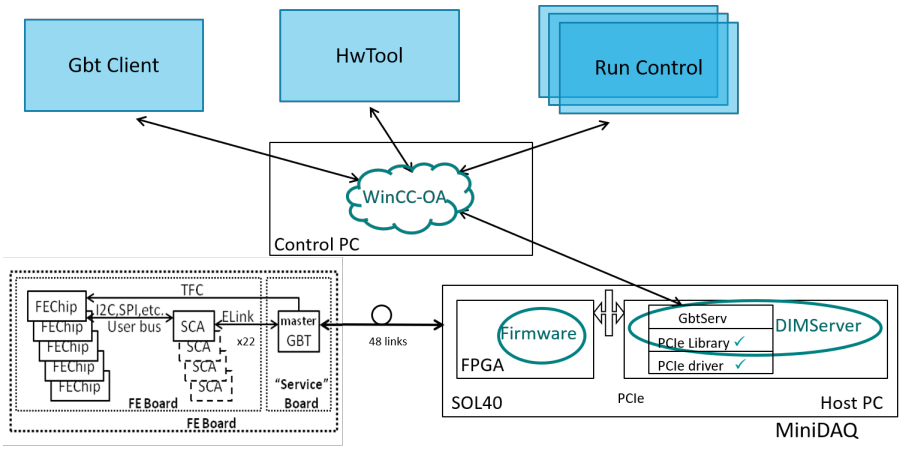

Figure 2. Front-End configuration and monitoring diagram. On the top is the representation of the user panels where the users interact with the detector. In the center is represented the control PC that operates the SCADA this. On the lower right corner is the PCIE40 board (SOL40 flavour) in its host server. On the lower left is the custom front-end board that is different for each part of the detector, but as a rule use the GBT chipset to connect to the SOL40 boards.

2 Firmware architecture

The control and monitoring of the Front-Ends is done by driving the GBTx and GBT-SCA chips, located in the Front End boards, through the optical links in the SOL40 board. Within this board's firmware, the SOL40-SCA core is the one responsible for sending ECS commands to the GBT-SCA chip. The GBT-SCA in turn, will interface directly the devices to be controlled and monitored through a set of serial protocols (I2C, JTAG, SPI), general purpose interfaces (GPIO) and ADC/DAC channels. The GBTx chip can be configured either through its I2C port or directly through the fiber by using the IC (Internal Control) field of the GBT protocol word. The SOL40 firmware contains also a block to control the GBTx IC field, denominated SC-IC, and it is shared by all the links for resource usage optimization reasons. The commands of both these blocks are then mapped to the correct e-links inside each link's SOL40 core block. A simple schematic of the SOL40 firmware architecture is presented in Figure 3.

Each one of the SOL40 cards can connect to 48 GBT front-ends using a different bidirectional optical link. In turn, these GBT chips are able to connect to up to 32 different GBT-SCA chips. The number of actual Front-End devices to be controlled and monitored by each GBT-SCA, and also the number of GBT-SCAs controlled by each GBTx is highly dependable on each Front-End Board architecture. The main requirements for the SOL40-SCA core are that it must accommodate this potentially high number of devices while also simplifying the job of the software running on the PCIe40 host server, allowing it to perform complex operations

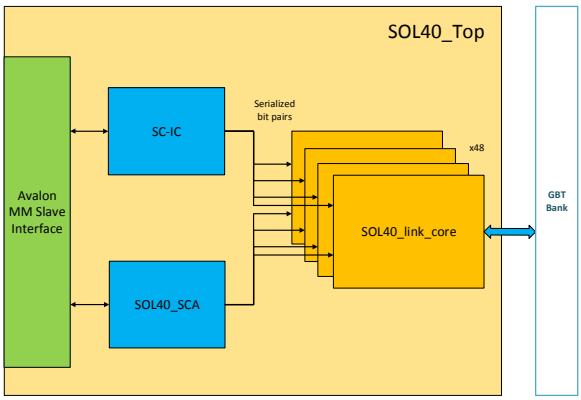

Figure 3. SOL40 firmware flavour architecture.

in each GBT-SCA (eg. configuring a FPGA through JTAG) as fluidly as possible. Since we can not pipeline the commands to the GBT-SCA, nor reduce their round trip time which is bound by the HDLC encoding and the length of the optical fibres, the firmware minimizes the time between two consecutive commands by already having implemented which sequence of commands to send for each control operation ordered by the software.

2.1 The SOL40-SCA core

The SOL40-SCA core is the firmware block in charge of translating commands sent by the software layer into GBT-SCA commands to be sent over the e-links. It consists of a block that scales with both the number of GBT links that are being used as well as with the number of GBT-SCAs served in each of the links. There are 4 layers with different functions in the SOL40-SCA:

- The Interface Layer implements registers for the control system to perform operations on the core through the Avalon Bus.
- The Buffer Layer holds information of the commands to be sent to the GBT-SCAs and also of their replies until the Control System reads them through the Interface Layer
- The Protocol Layer is responsible for reading the commands stored in the Buffer Layer, written in a generic format (ECS format), transforming it into one or several SCA commands, routing them to the chosen SCA through the HDLC encoding layer.
- The HDLC encoding layer is responsible for encapsulating the SCA payload into the HDLC protocol [7] and sends these frames through the GBT link. This layer consists of the firmware block GBT-SC [9].

A schematic of the architecture of the SOL40-SCA block can be seen in Figure 2.1. An explanation for the Buffer layer and the Protocol layer, which are the core functionality of SOL40-SCA, will be explained in the next in section 2.1.1 and 2.1.2.

2.1.1 The Buffer Layer

The buffer layer essentially constitutes a place to temporarily hold command data before it is consumed by the Protocol layer, or to hold reply data while the Control System does not collect it through the Interface Layer. For the command data, going towards the protocol

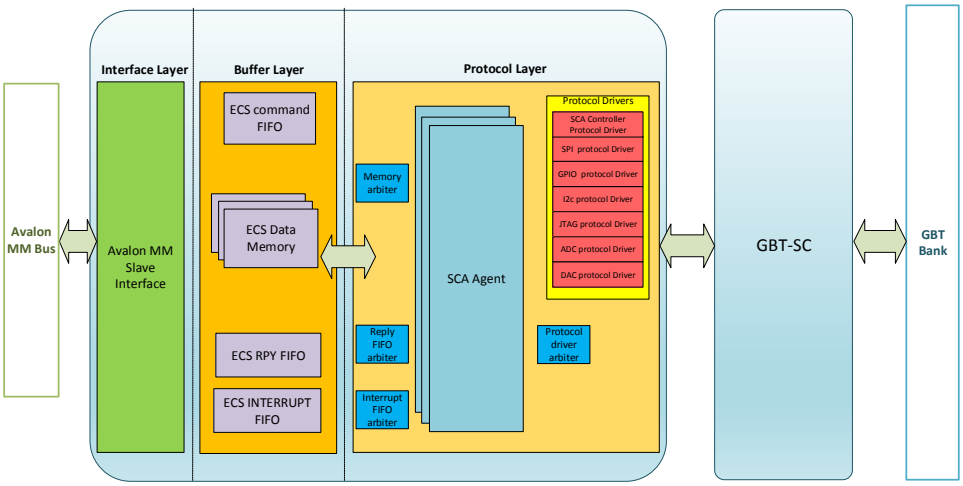

Figure 4. SOL40-SCA firmware block architecture.

layer, a FIFO and a memory is put in place on a per GBT link basis. The command specifications are stored in the FIFO, written to by the interface layer and read by the protocol layer. Each slot of the FIFO contains a 128 bit word that encodes the specifications of an ECS command. This command can be a simple command (corresponding to one single GBT-SCA operation) or a composite command (corresponding to a sequence of commands to a single GBT-SCA). These command structures contain specifications of which SCA they address, as well as which channel and command in the SCA to execute. Moreover they carry protocol specific configurations such as clock speed, register addresses for the FEE or signal sampling specifications, as well as the address of the data to be sent over to the SCA, which is to be stored in the memories (one memory per GBT link). Additionally, these commands carry an identifier that is attributed by the software layer so that later the retrieval of the reply is simplified. The number of SCA commands that can potentially be unrolled from one ECS command is proportional to the size of the instantiated data memory, and up to the limit of representation of the length field in the command word transmitted over the FIFO. The most common operation is to configure the GBTx chips that are handling the data readout into the TELL40s. Each one of these chips has 366 registers of 1 byte each. To configure these with one ECS command, the firmware has to unroll it into 132 GBT-SCA commands and the data to be sent will occupy 105 words of 32 bits in the command data memories of the buffer layer (This number is due to the fact that for each 14 bytes sent over I2C to a GBTx, there have to be 2 bytes of address in the I2C payload).

On the reply side of this mechanism, the reply data to be read by the Control System is stored in a separate memory, denominated the reply memory. There is also one of these memories per GBT link. The existence of a reply is signaled in a single reply FIFO for all the links. This reply FIFO is composed of the identifier of the command (that was chosen by the software layer upon sending it) and by a status word to signal whether everything went as expected with the command. Once the software picks up the reply, it will, through the identifier code, retrieve the reply data from the right address in the reply memory.

2.1.2 The Protocol Layer

The protocol layer is the one orchestrating the consumption of commands and retrieval of replies. On a top level view, it is composed of a protocol driver block, which encodes a sequence of SCA commands for each ECS composite command code, and the sca manager block. The sca manager block itself is composed of several "SCA agents", one per physical GBT-SCA, that hold information on the progress of a given ECS command execution. Initially, the ECS commands are read from the command FIFOs in the buffer layer, with the help of one arbiter managing the SCA agents in one GBT link. Once the command is stored in the SCA agent, it starts unrolling it with the help of the protocol driver block, which is accessed through the means of a global arbiter for all the SCA agents. The protocol driver handles one sca agent request per clock cycle. Once the SCA agent is ready it will issue an access request signal to the protocol driver arbiter and wait till it's granted. It will be busy until the reply for that GBT-SCA comes around and then another request is sent immediately, waiting only one clock cycle in case the protocol driver is not handling any other SCA at the time. This means then that the protocol layer outputs up to one command per clock cycle to the GBT-SCAs even if many sca agents are requesting to send commands at the same time. Typically each SCA agent waits 130 to 160 clock cycles to obtain a reply from the simplest commands sent. Commands that trigger the SCA chip to execute I2C, SPI or JTAG transfers will typically last a maximum of 1000 clock cycles. During this period the protocol driver has time to service many sca agents.

When a command is sent, the SCA agent latches a few values sent by the protocol driver, that will let the agent know what to do when the reply comes. It can either do nothing and just execute the next command, or it can use the reply data and write it to the reply memory, it can save the reply data as status to be sent when the ECS command is finished, or it can be signaled that that reply is the last one expected so it can send the summary status over to the reply FIFO in the buffer layer. Once this happens the cycle starts again and the agent will pick up another command to execute from the command FIFO, if one is available.

This architecture aims at being able to handle several big ECS commands at the same time in different GBT-SCAs, and the only bottleneck is the size of the memories used. It was also thought of in a way that allowed the software to send several commands to the firmware without necessarily having to wait for its reply. This aspect is granted by the identifier word that is present in the command word and the reply word, allowing the software to store a certain command structure and come back to it when the identifier comes back in the reply FIFO. This grants the software layer more versatility on this management process. This is complemented by the fact that the ECS commands can be as long as the memories allow, thus freeing the software of micromanaging these operations. The software can then relay the data received from the control system directly into the SOL40 firmware, without having to divide it in tranches, as long as the memory size in the buffer layer allows for it.

On a parallel functionality, the SCA agents also receive instructions from the operations FIFO, located in the buffer layer, which will issue commands to reset, connect or disable a certain SCA. These commands have priority over any other and can be executed at any time. This means that if an SCA agent will always choose to send the operation FIFO command before the regular GBT-SCA command.

They will also recognize special replies coming from the GBT-SCA that are encoded with a special transaction ID. These consist of configurable interrupts coming from the FEE. These interrupts are stored in a FIFO in the buffer layer, with an identification of the link and SCA index they originated from, as well as the content of the interrupt word states what pins in the GBT-SCA triggered it.

2.2 FPGA occupancy and design performance

One of the drivers of this design was that the SOL40 firmware needs to have a reasonable occupancy in the Arria10 FPGA in order for timing problems not to appear. In this regard, the most populated configuration that is needed for LHCb is to have a SOL40 board with the 48 links populated with 5 SCAs each. This configuration yields a 77% occupancy of the FPGA and an error rate of under 0.01% in the SCA communications. On any given ECS command, the throughput achievable through any of the I2C, JTAG or SPI ports of the SCA reaches 500 KB/s.

2.3 The Software Layer

The software layer is composed of three main components: the PCIe driver that allows processes to interact with the SOL40 firmware, the DIM server processes, denominated GbtServer, and the SCADA layer from where the detector is controlled. In this text we will put emphasis on the GbtServer layer as this is the one that directly interfaces the SOL40-SCA firmware block, through the PCIe driver. The design of the firmware was partly also driven by the goal to unburden the GbtServer of micromanaging GBT-SCA commands, allowing it to focus on other tasks. The designed architecture foresees the use of one GbtServer per SOL40 board or one GbtServer per Gbt Link. These options are interchangeable, and allow for more versatility in case there are too many FEE chips under each GBT link for one process to manage the whole 48 links of a SOL40 board. The GbtServer is tasked with:

- Interfacing the SCADA system through the DIM protocol.
- Implementing part of the Hardware Tool [8] used in LHCb to address the FEE by name instead of by address at the SCADA level.
- Carrying out monitoring on FEE registers subscribed in the Hardware Tool.
- Implementing the ECS commands that will instruct the firmware to carry out certain GBT-SCA command sequences.
- Orchestrating the use of the SOL40-SCA resources, namely the memory and FIFO usage, in a way that exploits the most out of the firmware block. For instance it might be more profitable to send ECS commands with less data if then one can fit several of these commands in the command data memory of the SOL40 to be executed in parallel.

Because the SOL40-SCA registers the FEE replies in a FIFO and they are uniquely identified, the GbtServer is able to send commands and receive replies on separate threads, and consequently it does not have to wait for the reply of a command to send another to the same SCA. This allows for several long operations to be carried on different GBT-SCAs simultaneously, which leads to the implementation of one thread per GBT-SCA to carry out these operations. These are fed by the DIM specific threads which handover instructions from WinCC-OA to the SCA specific threads, and on the lower level side, there is one thread in charge of interfacing the SOL40-SCA firmware to send commands and manage the memories in it, and another that has the sole purpose of retrieving replies from the reply FIFO and reply memories and handing these over to the SCA specific threads.

Long operations on a given single SCA are slightly faster when the memories in the SOL40 increases. The difference goes from 480 KB/s if the memory is only 256 words long, compared to 500 KB/s for a 4096 word long memory. This small difference can be explained by the fact that the time it takes to execute a command is dominated by two factors: writing the data into firmware memory, and having the firmware consume this data and send the commands to the GBT-SCA. These two are not pipelined in the current design and that poses a limitation to the speed gain. Tests for parallelization with the full fledged detector are needed to find the bottlenecks of this design but unfortunately there are not enough Front-Ends available yet.

3 Conclusion

The LHCb experiment is replacing the majority of its Front-End Electronics in the detector, and equiping the new ones with the GBT chipset to control the FEE chips. The functionality of these GBT-SCA chip, its numbers in the future detector, and the complexity of the new Electronics, are what drives the development of the FEE control infrastructure. To be able to configure and monitor the diversity and quantity of FEEs in the future LHCb detector, the software and firmware were designed together so that their interplay would be as smooth and fast as possible. The new design offloads micromanaging the GBT-SCA protocols to the firmware and tasks the software of managing the firmware resources instead and feeding it with new commands, the latter being as long as possible, due to the heavy parallelization carried out by the buffer layer and protocol layer in the SOL40-SCA block. With the new architecture, the firmware scales in way that allows for the control of the most crowded sub-detectors with a modest amount of SOL40 boards. The chosen design for the firmware also allows for the Gbtserver software processes to act in a much more parallel way by not necessarily waiting for a reply of a command before sending the next. This architecture is under the final stages of development and is going to be commissioned during the years of 2019 to 2021 with the installation of bigger sub-detector prototypes in the experiment.

References

[1] G. Liu, and N. Neufeld . *DAQ architecture for the LHCb upgrade.* Journal of Physics: Conference Series. **Vol. 513.** No. 1. (2014) IOP Publishing.

[2] P. Moreira, K. Wyllie, B. Yu, A. Marchioro, C. Paillard, K. Kloukinas, T. Fedorov et al. *The GBT Project.* Proceedings of Topical workshop on electronics for particle physics, TWEPP 2009, Paris, France, (2009), pg. 342

[3] M. Bellato, G. Collazuol, I. D'Antone, P. Durante et al. *A PCIe Gen3 based readout for the LHCb upgrade.* Journal of Physics: Conference Series. **Vol. 513.** (2014) No. 1. IOP Publishing.

[4] F. Alessio and R. Jacobsson. *Timing and fast control for the upgraded readout architecture of the LHCb experiment at CERN.* IEEE Transactions on Nuclear Science **60.5** (2013): 3438-3445.

[5] F. Alessio O. Callot, L. Cardoso, B. Franek, M. Frank, J. C. Garnier, C. Gaspar et al. *The LHCb Run Control System*, Real Time Conference (RT), 2010 17th IEEE-NPSS, Lisbon, (2010), pp. 1-6.

[6] F. Alessio, C. Caplan, C. Gaspar, R. Jacobsson, and K. Wyllie , et al. *A generic firmware core to drive the Front-End GBT-SCAs for the LHCb upgrade.* Journal of Instrumentation (2015) **10.02**: C02013.

[7] *International Standards Organization, Telecommunications and information exchange between systems - HDLC procedures*, ISO/IEC 13239:2002

[8] L. Cardoso, C. Gaspar, J. Barbosa and F. Alessio. *Controlling DAQ electronics using a SCADA framework.* Real Time Conference (RT), (2016) 20th IEEE-NPSS, Padua

[9] J. Mendez, P. Leitao, S. Baron, A. Caratelli. *New slow-control FPGA IP for GBT based system and status update of the GBT-FPGA project.* Topical Workshop on Electronics for Particle Physics, Santa Cruz, Ca, United States Of America, (2018), pp.083

[10] C. Gaspar, M. Dönszelmann, and P. Charpentier . *DIM, a portable, light weight package for information publishing, data transfer and inter-process communication.* Computer Physics Communications **140.1-2** (2001): 102-109.