Available on CMS information server

CMS CR -2022/273

The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

07 December 2022 (v4, 19 January 2023)

RPC Phase-2 Link System Remote Programming for the CMS HL-LHC Upgrade

Behzad Boghrati for the CMS Muon Collaboration

Abstract

The high luminosity LHC upgrade of the CMS RPC electronics system for the low-eta region includes the RPC link system upgrade and back-end electronics. The link system, as an off-detector electronics, comprises control boards and link boards. The electronics will be installed in the Compact Muon Solenoid (CMS) tower racks in the underground experimental cavern (UXC), and during the experiment run, there will be no access to these cards. Therefore, any further Field Programmable Gate Array (FPGA) firmware modification could only be carried out by remote programming techniques. Moreover, FPGA firmware must be designed to detect failures and automatically keep flashbacks of the original configuration memory. This work will explain how the remote programming procedure is implemented into the link system and how the RPC back-end electronics will handle it.

Presented at *RPC2022 XVI Workshop of Resistive Plate Chamber and Related Detector at Cern in Sept. 2022*

RPC Phase-2 Link System Remote Programming for the CMS HL-LHC Upgrade

M. Ebrahimi[a],
on behalf of the CMS Muon Group

[a]*School of Particles and Accelerators Institute for Research in Fundamental Sciences (IPM) P.O. Box 19395-5531 Tehran Iran.*

Abstract

The High-Luminosity LHC upgrade of the Compact Muon Solenoid (CMS) RPC electronics system for the low-η region includes the RPC link system upgrade and back-end electronics. The link system, as an off-detector electronics, comprises control boards and link boards. The electronics will be installed in the CMS tower racks in the underground experimental cavern (UXC), and during the experiment run, there will be no access to these cards. Therefore, any further Field Programmable Gate Array (FPGA) firmware modification could only be carried out by remote programming techniques. Moreover, FPGA firmware must be designed to detect failures and automatically keep flashbacks of the original configuration memory. This work will explain how the remote programming procedure is implemented into the link system and how the RPC back-end electronics will handle it.

Keywords: Phase-2 Upgrade, RPC Link System, RPC back-end electronics, Slow Controller, Remote Programming, FPGA
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

The present Resistive Plate Chambers (RPC) cover the pseudorapidity (η) region below 1.9 in the CMS experiment [1]. RPC chambers use between three and eighteen on-detector frontend electronics boards (FEB) [2]. The fired RPC strip signals are amplified, discriminated, shaped, and transferred to low-voltage differential signals (LVDS) by the FEB. The LVDS signals are transmitted to the off-detector electronics known as the RPC link system through twisted pair round cables. In the link system, a time-stamp is assigned to every incoming hit signal, and together with the strip number, they are sent to the next trigger layer. This system will be functional until the end of LHC Run3 data taking, when they will be replaced by brand new electronics for the High-Luminosity LHC (HL-LHC) "Phase-2" upgrade [3].

The new link system [4], similar to the legacy link system [5, 6], contains 216 control boards (CB) and 1376 link boards (LB). Every CB and three/six/nine LB are shaping a link system card pack and installed into a customized VME crate called Link Board Box (LBB). Figure 1 shows the final prototype of the new link system card packs. The electronics have been designed using the Xilinx Kintex-7 FPGA device and coded with a high-level hardware description language (VHDL). Using this FPGA family, it is feasible to achieve the main goals of the link system upgrade, which are to improve the timing resolution of the RPC muon hit to 1.56 ns and to meet the HL-LHC requirements by enhancing the data transmission bandwidth to 10.24 Gbps.

The RPC architecture for the Phase-2 upgrade is shown in Fig. 2. The new link system will be installed in the CMS underground cavern tower racks and run in a radiation environment. As a result, the link board and control board firmware must be mitigated for radiation effects such as bit flip in the digital flip-flops, SRAM and configuration memory, known as a single event upset (SEU). Moreover, there will be no access to these electronics during the physics run. In case of necessity for FPGA firmware modification, it is neither feasible to access UXC nor to plug a programmer into every single board. The only bridge between the link system in the cavern and the control room on the surface is the RPC back-end electronics (BEE), as shown in Fig. 2. Therefore, using the RPC BEE and developing an FPGA remote programming technique for the link board and control board is essential.

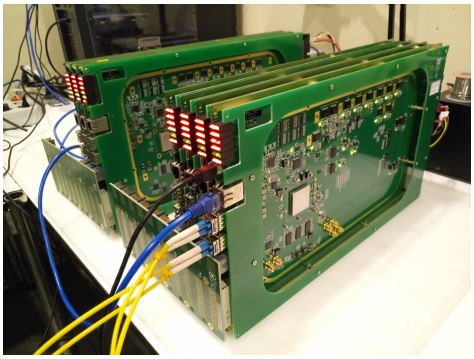

Figure 1: RPC Phase-2 link system card pack prototype

2. The link system remote programming concept

The main idea for remote programming is to program the link system FPGA through the RPC back-end electronics (Fig.

Email address: ebrahimi@ipm.ir (M. Ebrahimi)

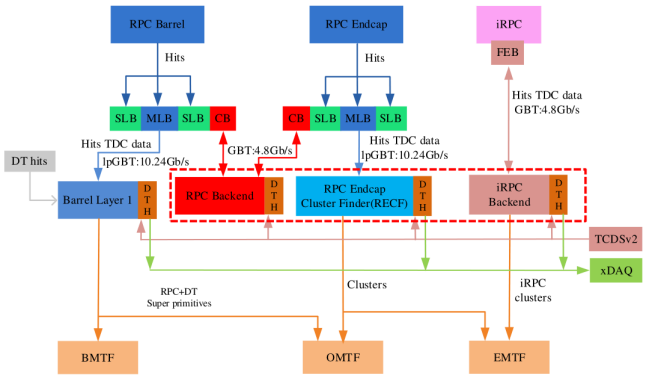

Figure 2: RPC Phase-2 upgrade architecture and Layer-1 trigger.

3). In this way, each FPGA receives the new configuration from BEE in bytes format, called updated bitstream. It stores them in a specific location on the external flash memory, known as the update image area. After transferring all bytes to the updated image area, back-end electronics inform the FPGA that the file transmission is completed. In the next step, FPGA reboots and reconfigures itself from the updated image area.

To secure FPGA not booting from a corrupted updated image file, the FPGA must be able to reliably fallback and configure from a well-proven old working version called the golden image area. Typically, the golden image area is always kept the same to ensure its known good condition for all cases. In Xilinx FPGA, conventional remote update solutions use the FPGA built-in MultiBoot and Fallback features. The MultiBoot feature enables the FPGA to selectively load a bitstream from a specified address in flash memory. If the FPGA detects a configuration error, the Fallback feature resets the FPGA. It retries the configuration from address zero of the flash memory that keeps the beginning address of the golden image area [7].

The main drawback of this method is that it takes twice the standard configuration time to detect a failure in the updated image area and reconfigure FPGA with the reserved golden bitstream. Indeed, this programming solution only relies on the FPGA try-and-recover method, which can degrade the system's reliability. To have a reliable and robust programming solution and quick FPGA configuration time, we assigned the programming operation through the programming algorithm for the bitstream update process. Part of this programming algorithm is implemented in BEE, and other parts are deployed on the control and link boards. In this solution, after verifying that the updated bitstream is correctly received and stored in the flash memory, the critical switch word is set to ON, which means FPGA should jump to the updated bitstream area. Otherwise, the critical switch word stays on its default OFF value that conducts FPGA to load its configuration from the golden image area.

3. Remote programming handler in the RPC back-end electronics

As already mentioned, the only bridge between the link system and the control room is the RPC back-end electronics

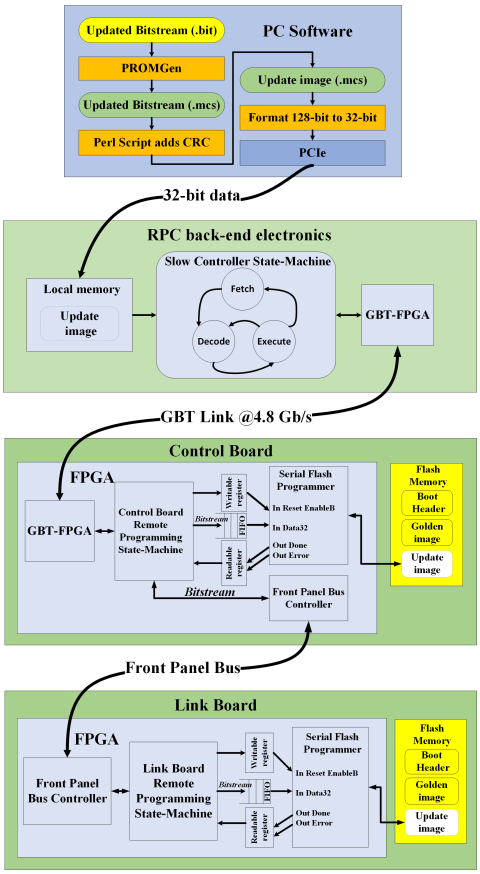

Figure 3: Slow Control and Remote Programming Chain. From the top, The PC software and update bitstream file generator, RPC back-end electronics, control board, and link board.

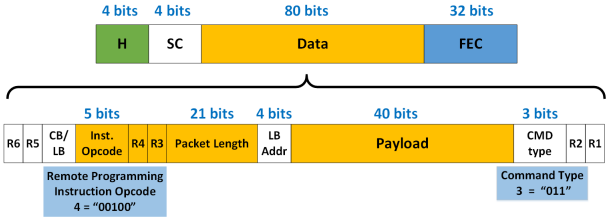

Figure 4: GBT link 120-bit frame format (top), 80-bit remote programming payload carried out by the GBT link user data field (bottom).

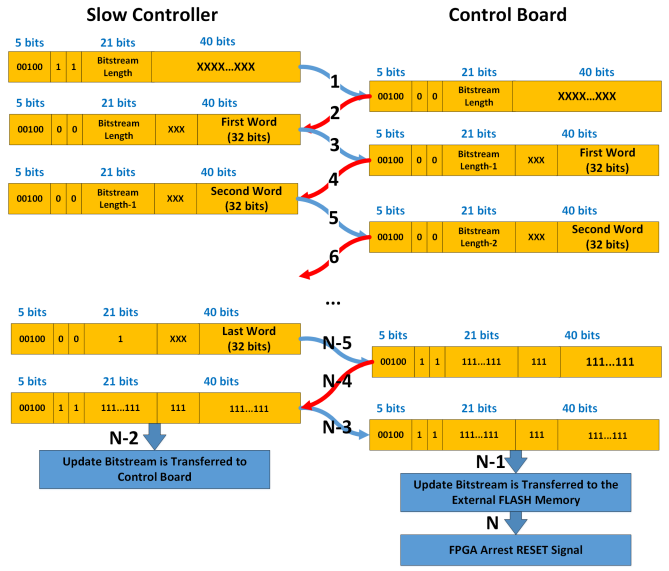

Figure 5: GBT frame exchanges between Slow Controller (left) and Control Board (right).

(Fig. 3). These systems are connected via optical links. The radiation-mitigated firmware in the control board and link board is implemented into the FPGA. Three parts of the firmware mitigation solution in the control board are using triple modular redundancy, internal scrubbing, and a robust data transmission line coding that allows the correction of bursts of errors caused by SEU.

The CERN standard GBT link encoding is deployed in the control board FPGA to keep the data transceiver SEU tolerant. In practice, a sequence of up to 16 consecutive incorrectly received bits can be corrected by the GBT link encoding. In the GBT link protocol, every 88-bit transmitted data requires an extra 32-bit code, known as Forward Error Correction (FEC), for error recovery at the receiver side. The GBT link frame (Fig. 4) comprises 120 bits, including 4 bits of the frame header (H), 4 bits of slow control (SC), 32 bits used for FEC, and 80 bits for user data. This frame is transmitted on every bunch crossing (25 ns), resulting in a line data rate of 4.8 Gbps.

The same line encoding and interface must be implemented in the RPC back-end electronics, which firmware is previously tested and validated on the KC705 evaluation board. This firmware is called "slow controller". Three kinds of functionalities or commands have been developed for the slow controller. Command type A is dedicated to reading and writing the link system and FEB parameters; type B reads the link system histograms, diagnostics, and data monitoring; command type C is for remote programming. These command types are supervised by the Slow Controller State Machine (SC-SM).

Regarding remote programming, the main scope of this work, when a command of type C is issued, the following steps are executed by the slow controller and the control board state machines:

1. At the beginning, an 80-bit remote programming payload (Fig. 4, bottom) carried out by the GBT link (Fig. 4, top), and containing
   (a) remote programming instruction opcode "00100" (5-bit),
   (b) reserved bits R3 and R4 (2-bit) "00",
   (c) bitstream length (21-bit),
   (d) 40 bits don't-care ('X') data payload,
   (e) reserved bits R1, R2, R5, and R6 with "0" content,
   (f) control or link board selection bit (CB/LB) set to "0",
   (g) link board address (4-bit) set to "0000", and
   (h) type C command (3-bit) set to "011"
   is sent continuously by the back-end electronics to the control boards (Fig. 5, top-left).
2. Once the control board receives the GBT frame and its 80-bit remote programming payload, it erases the update area in the flash memory, and responds to the back-end electronics with an identical as a received frame, except for R3 and R4 2-bit changes from "11" to "00" to inform that the remote programming command is received, and it is ready to receive the rest of the bitstream file.
3. When the back-end electronics receives a reply from the control board, it sends the length of the bitstream and the first data word (32-bit) that should be stored in the first memory address in the flash update area.
4. After the control board receives a new GBT frame, it stores the bitstream in the external flash memory. In response, it decrements the bitstream length and sends it back to the back-end electronics.
5. This procedure will continue until the last word of the bitstream, which bitstream length reaches the one; this state is shown in Fig. 5 by index N-5. Once the control board receives the last bitstream word, it replaces all fields of frames with ones and sends the GBT frame back to the back-end electronics.
6. By receiving the last GBT frame with all fields set to one, the back-end electronics realizes that the control board has received all the words of the bitstream file. Therefore, it sends the last received frame again to the control board to inform that the data transmission is completed (Fig. 5, index N-3), and then it goes to the idle state (Fig. 5, index N-2).
7. Once the control board receives the GBT transmission completion frame, it arrests the reset pin of the FPGA for reloading its configuration memory from the new update bitstream file (Fig. 5, index N).

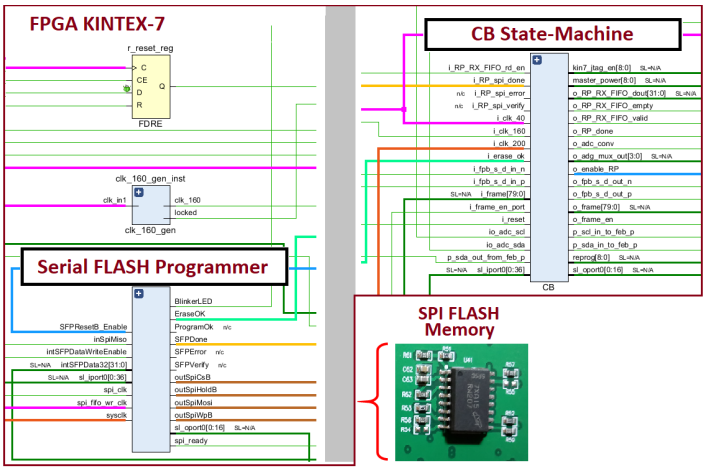

Figure 6: Schematic representation of the firmware of the control board state machine (CB-SM) and the Serial Flash Programmer (SFP) in the control board.

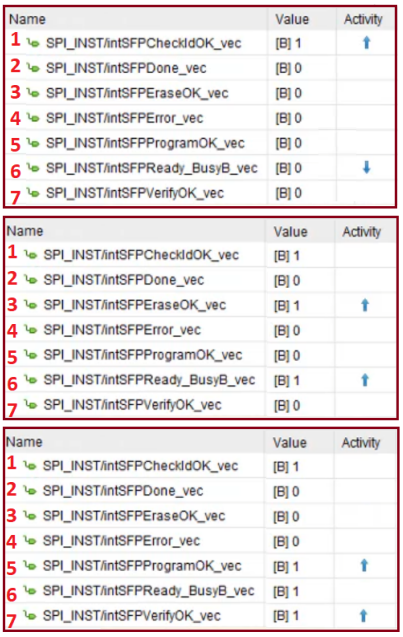

Figure 7: SFP controller output status flags. On the top, the SFP controller is erasing the flash update area. For this reason, the Ready/Busy flag (row 6) shows logic low. In the middle, the flash is erased (row 3), and the SFP controller is ready (row 6) for a new command. At the bottom, the flash is programmed (row 5) and verified for zero error (row 7), and the SFP controller is ready for further requests (row 6).

4. The link system remote programming firmware

The control board firmware comprises

1. the CB state machine,
2. the GBT link transceiver,
3. the serial flash programmer (SFP),
4. the RPC FEB controller,
5. the clock manager and jitter cleaner,
6. the ADC controller, and
7. the front panel bus controller.

In remote programming, like other slow control commands, the frames and bitstreams are received by the GBT link receiver and processed by the CB state machine. The CB-SM manages the SFP to erase, verify, and reprogram the external flash memory update area with the newly received bitstream. Figure 6 illustrates the schematic representation of the firmware of the CB-SM, SFP controller, and connection of the external flash memory to the SFP interface ports.

In the control board, once the remote programming command is received, the CB-SM enables the SFP controller by setting the Reset/Enable input to logic low and waits until the SFP Erase-ok flag is set to logic one (Fig. 7, top). After completing this state, the SFP Erase-ok, check ID, and Ready status flags are set to the logic one (Fig. 7, middle) to show that the update area of the flash memory is erased and SFP is ready to receive the new update bitstream file. The CB-SM checks the status of these flags, and when it realizes that all of them are active, it gives the green light to the slow controller to send the new update bitstream file.

In practice, we prefer to minimize the programming time for which, instead of sending one frame from the slow controller to the CB, a burst of 1024 frames is sent. In the control board, the GBT link receiver buffers the received data to the FIFO memory. Once something is written into the memory, its empty flag changes and informs the CB-SM that new data is available. In the next step, the CB state machine reads the FIFO memory and transfers each word to the flash update area. When the entire update bitstream is transferred to the CB flash memory, the corresponding cyclic redundancy check (CRC) value calculated by the PC software is written on the last location of the flash update area.

Finally, the SFP controller checks the entire update area, calculates the CRC value of the received area, and compares it with the primary CRC value to ensure that the new update area is error-free. The SFP controller changes the critical switch word to the ON value to complete the programming process. Additionally, it sets the program-ok flag to logic one (Fig. 7, bottom) to inform the CB state machine that programming the flash update area with the new update bitstream had been successful. Once the CB-SM realizes that the flash update area is programmed correctly with a new bitstream file, it arrests the FPGA reset pin to force it for reconfiguration. As a result of this warm boot, the FPGA reads the first location of the flash memory, and due to the critical switch word being ON, it jumps to the update area and reloads the FPGA configuration memory with the new update area.

The remote programming of the link board FPGA is identical. In this case, the update bitstream is forwarded to the destination link board through the front panel bus controller and the front panel PCB board. The data transmission between these electronics is under the supervision of the front panel controllers, which are implemented in FPGA firmware.

5. Conclusion

The new Phase-2 RPC link system will be installed in the CMS UXC tower racks and run in a radiation environment. During the HL-LHC run period, it is mandatory to be able to access these electronics remotely and, in case of necessity, update their firmware. This work explains the firmware and the link protocol for the RPC link system and back-end electronics. This firmware has been tested on the last version of the link system prototypes and the RPC back-end electronics emulator, called the slow controller. The total time needed for remote programming of an FPGA is less than a few seconds.

6. Acknowledgements

We would like to acknowledge the enduring support for the Upgrade of the CMS detector and the supporting computing infrastructure provided by the following funding agencies: FWO (Belgium); CNPq, CAPES and FAPERJ (Brazil); MES and BNSF (Bulgaria); CERN; CAS, MoST, and NSFC (China); MINCIENCIAS (Colombia); CEA and CNRS/IN2P3 (France); SRNSFG (Georgia); DAE and DST (India); IPM (Iran); INFN (Italy); MSIP and NRF (Republic of Korea); BUAP, CINVESTAV, CONACYT, LNS, SEP, and UASLP-FAI (Mexico); PAEC (Pakistan); DOE and NSF (USA).

References

[1] CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3 (2008) S08004. doi:http://dx.doi.org/10.1088/1748-0221/3/08/S08004.

[2] M. Abbrescia, et al., New developments on front-end electronics for the cms resistive plate chambers, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 456 (1-2) (2000) 143–149. doi:http://dx.doi.org/10.1016/S0168-9002(00)00980-3.

[3] CMS Collaboration, The Phase-2 Upgrade of the CMS Muon Detectors, CERN-LHCC-2017-012 CMS-TDR-016 (2018).

[4] B. Boghrati, et al., CMS phase-2 upgrade of the RPC Link System, JINST 16 (2021) C05003. doi:http://dx.doi.org/10.1088/1748-0221/16/05/C05003.

[5] K. Bunkowski, et al., Diagnostic tools for the rpc muon trigger of the cms detector-design and test beam results, IEEE Transactions on Nuclear Science 52 (6) (2005) 3216–3222. doi:http://dx.doi.org/10.1109/TNS.2005.860174.

[6] K. Bunkowski, et al., Synchronization methods for the pac rpc trigger system in the cms experiment, Measurement Science and Technology 18 (8) (2007) 2446–2455. doi:http://dx.doi.org/10.1088/0957-0233/18/8/020.

[7] R. Kuramoto, Quickboot method for fpga design remote update, Xilinx Application Note: 7 Series FPGAs XAPP1081 (v1.1) (2013) 1–41.