Remote Configuration of the ProASIC3 on the ALICE Inner Tracking System Readout Unit

Shiming Yuan
on behalf of the ALICE Collaboration

Shiming.Yuan@uib.no

University of Bergen

22^{nd} Virtual IEEE Real Time Conference
21 October 2020

Outline

- Overview of ALICE Inner Tracking System (ITS)
- ALICE ITS readout chain
- Motivation for remote configuration of ProASIC3 (PA3) on ITS Readout Unit (RU)
- Configuration data path in GBT
- Configuration data path on RU
- The GBT-SCA interface
- `DirectC`
- `pa3jtag`
- Summary

ALICE, ALICE Inner Tracking System

- A Large Ion Collider Experiment
- Higher luminosity of heavy-ion collisions in Run 3 of LHC
- Higher data rates in ALICE $\Longrightarrow$ detectors and read-out electronics replaced
- Inner Tracking System (ITS): the innermost sub-detector
- Upgraded ITS, ALPIDE (**AL**ice **PI**xel **DE**tector, a tailored CMOS monolithic active pixel sensor) $\Longrightarrow$ New ITS Readout System

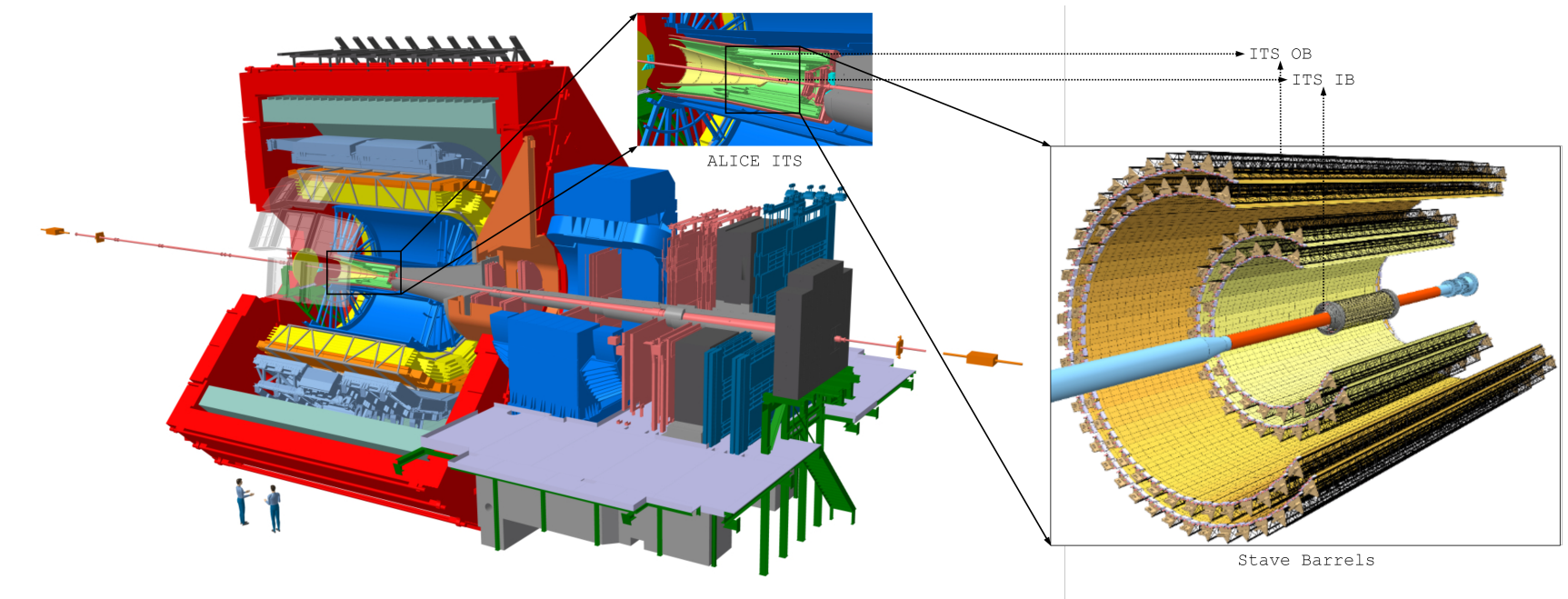


Figure: ALICE and ITS

Overview of ITS readout chain

- ALPIDE sensors are controlled and read out by Readout Units (RUs)
- 192 RUs ship data from front-end sensors to Common Readout Units (CRUs)
- RU main FPGA: data streaming, trigger handling, control of ALPIDEs, interfaces Power Unit
- RU Auxiliary FPGA (PA3) configures and scrubs the main FPGA
- Radiation environment for RU: anticipated high-energy hadron flux (above 20 MeV): 1 kHz/cm^2 $\Longrightarrow$ SEU in SRAM-based main FPGA

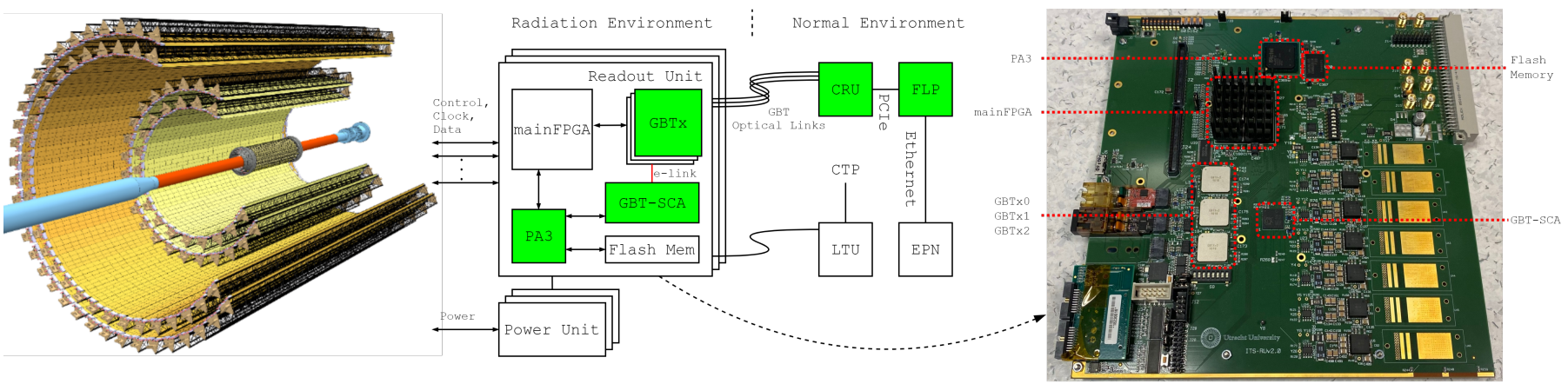
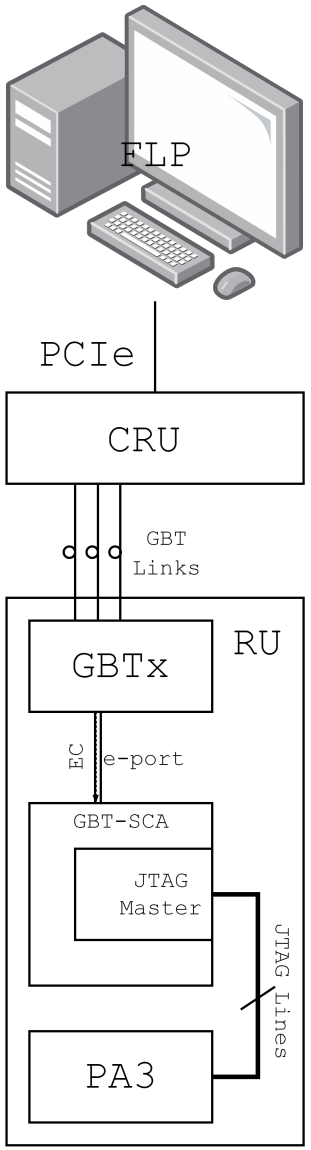


Figure: ALICE ITS Readout System: Sensor Staves, RU, CRU, and FLP

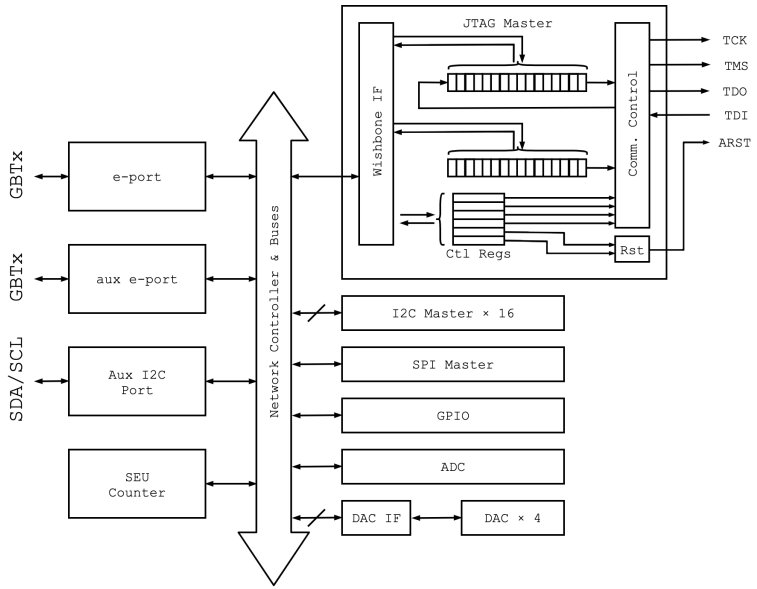
Motivation for remote configuration of PA3

- SEU mitigation techniques of Readout Unit:
  - TMR in main FPGA
  - ⋆ Scrubbing: PA3 reconfigures the main FPGA continuously; A scrub cycle is ≈1.7s
- Foreseen circumstances where reprogramming of the PA3 is needed
  - Correct possible issues
  - Add new functionality
  - Configure PA3s with special builds (e.g. fault injection support) to qualify the robustness of new versions of the main FPGA design
- Minimum hardware components of remote configuration of PA3
- Infrequent operation



Configuration data path in GBT

- RoC module of ALICE O^2: high level interface for accessing and controlling the CRU
- In GBT Frame, 2-bit External Control field in the 4-bit Slow Control
- On ITS RU, EC is passed to GBT-SCA, the GBTx is transparent between CRU and GBT-SCA
- Two 32 $\times$ 4 buffer registers (TDO/TDI, TMS) in JTAG master channel



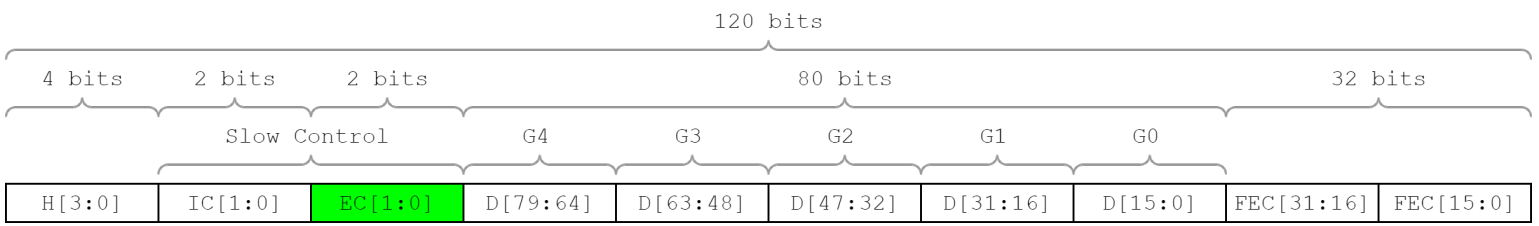


Figure: GBT frame composition

JTAG configuration chain on RU

- TMS (Test Mode Select) pins are connected in parallel
- TDO (Test Data Out)/TDI (Test Data In) pins are connected in series
- Instruction Registers (IRs) and Data Registers (DRs) are connected in a long shift register from TDO to TDI of GBT-SCA
- PA3 IR length: 4; main FPGA IR length: 6
- DR length depends on the instruction in IR, when in by-pass mode, DR length is 1

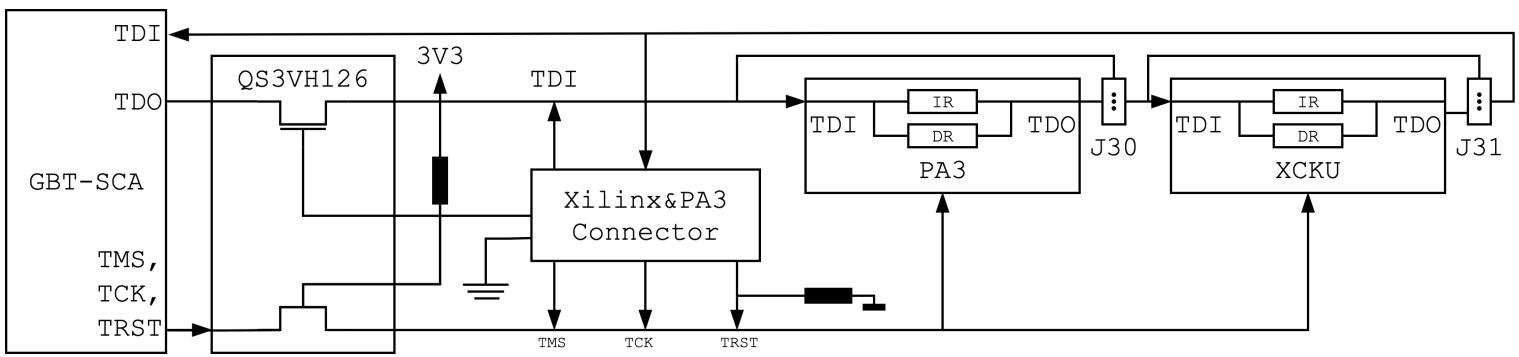
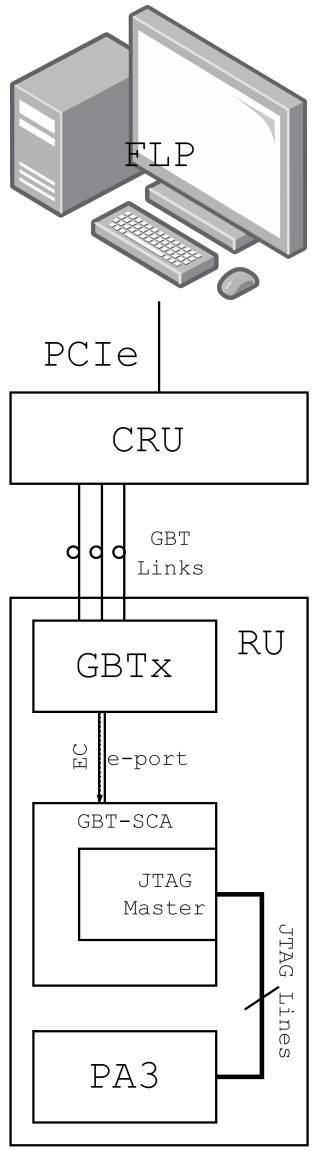


Figure: RU JTAG configuration scheme

GBT-SCA interface

- The GBTx chip is transparent in the communication between the CRU and the GBT-SCA
- An abstract layer of the JTAG master channel in the GBT-SCA for the JTAG configuration chain
- The access to the JTAG master channel registers takes on average ≈3 ms
- Minimizing the number of Slow Control transactions on GBT-SCA interface level: using 'caching' global variable for data sent to TMS, Control registers



DirectC - 1/2 Configuration file

- DirectC is a suite of C++ code for ISP
- The .dat file is generated by Microsemi Libero dev tool
- Information in a .dat file

Table: Format Description of PA3 .dat File

	Size (Byte)	Information
HB	69	Designer version number, Header size, Image size, Compression Flag, etc..
DLT	$n_1 \times 9$	Data Identifier 1, Pointer to data 1 memory location in the data block section, # of bytes of data 1; Data Identifier 2, Pointer to data 2 memory location in the data block section, etc..
DB	$n_2 + 2$	Configuration binary data, CRC of the entire image

DirectC - 2/2 Procedure

Listing 1: Pseudo code for PA3 idcode readout

```
1  void dp_read_idcode(void){
2     opcode = IDCODE ;
3     IRSCAN_in();
4     goto_jtag_state(JTAG_RUN_TEST_IDLE);
5     DRSCAN_out(IDCODE_LENGTH, global_buf);
6     device_ID = mem_shift(global_buf);
7     return;
8  }
```

- An example: read the device id code
  1. `IRSCAN_in()`: shifts the opcode to the TDO/TDI line
  2. `goto_jtag_state()`: shifts TMS data to invoke JTAG stage changes
  3. `DRSCAN_out` shifts TDO read back from the TDO/TDI line to memory space
- `IRSCAN_in()` and `DRSCAN_out` require 12+ SC transactions, receptively; `goto_jtag_state()` requires 4+ SC transactions; this function consists of ≈30 SC transactions

pa3jtag - Why DirectC

- Why not design the tool from scratch?

  - The tool relies on many parameters maintained by the vendor $\Longrightarrow$ difficult to start from scratch
- Why not use STAPL Player?

  - Tried but failed with syntax error, maybe due to incompatibility between the software and hardware
- Through discussion with a LHCb sub-detector team, they also diverted from STAPL Player to DirectC, as an algorithmic prototype for the firmware development of SOL40 SCA core*

```
>:tree
.
├── G3Algo
│   ├── dpG3alg.c
│   ├── dpG3alg.h
│   ├── dpcore.c
│   ├── dpcore.h
│   ├── dpfrom.c
│   ├── dpfrom.h
│   ├── dpnvm.c
│   ├── dpnvm.h
│   ├── dpsecurity.c
│   └── dpsecurity.h
├── G4Algo
│   ├── dpG4alg.c
│   └── dpG4alg.h
├── G5Algo
│   ├── dpG5alg.c
│   └── dpG5alg.h
├── JTAG
│   ├── dpchain.c
│   ├── dpchain.h
│   ├── dpjtag.c
│   ├── dpjtag.h
│   ├── gpio_hw_interface.c
│   └── gpio_hw_interface.h
├── RTG4Algo
│   ├── dpRTG4alg.c
│   └── dpRTG4alg.h
├── dpalg.c
├── dpalg.h
├── dpcom.c
├── dpcom.h
├── dpuser.c
├── dpuser.h
├── dputil.c
└── dputil.h

5 directories, 30 files
```

* "The new version of the LHCb SOL40-SCA core to drive front-end GBT-SCAs for the LHCb upgrade," PoS, p. 078, 2018

pa3jtag - Concatenating IRSCAN_in

Listing 2: Pseudo code for JTAG IRSCAN_in

```
1  void IRSCAN_in(void) {
2    goto_jtag_state(JTAG_SHIFT_IR);
3    dp_shift_in(OPCODE_BIT_LENGTH, &opcode);
4    goto_jtag_state(JTAG_PAUSE_IR);
5  }
```

- Each goto_jtag_state consists of: 1 write to CTL reg, 1(+) write(s) to TDO reg, 1(+) write(s) to TMS reg, 1(+) write to GO_COMMAND reg; the same for do_shift_in

  $\therefore$ IRSCAN_in requires 12+ SC transactions in DirectC algorithm
- Bottleneck: Slow Control transactions (each takes 3ms)
- To reduce the number of SC transactions $\Longrightarrow$ patterns in the bitstream: the only variable here is the opcode
- Pre-calculate the full data packet, when the full length of this packet $\leqslant$ 32 bit, pa3jtag only requires 4 GBT-SCA writes
- Eliminate the number of SC transactions $\Longrightarrow$ speed up $\times$(3+)

pa3jtag - Optimization of the CG writing

- Column Grid (CG or Row) is the memory array that stores the configuration in FPGA
- Over 99.9% of the `.dat` image is Data Block, accelerating writing on CG speeds up the whole process of configuring PA3
- PA3 has a 3444-Row configuration memory array
- Writing to one Row requires a 98-bit TDI packet $\Longrightarrow$ pre-calculate this 98-bit packet, and transmit it with 4 TDI writes, 1 JTAG GO command
- The TDO read-backs in CG writing represent the hardware information, which are constants in the context $\Longrightarrow$ make "fake TDO read-backs"
- Optimization on the number of SC transactions based on function level $\Longrightarrow$ accelerate the configuration from more than 30 hours to 31 min

Generic JTAG function	Number of SC Tr. after optimization
Wr TDI/TDO line	2 (-, depending on the GBT-SCA interface 'caching') + 1 to 4 (depending on packet length)
Rd TDI/TDO line	1 to 4 depending on packet length

Summary 1/2

- Comparison: the LHCb approach
  - A dedicated SOL40 interface board to distribute all the control signals to the FEE
  - A SOL40-SCA core in the firmware sends ECS commands to the GBT-SCA

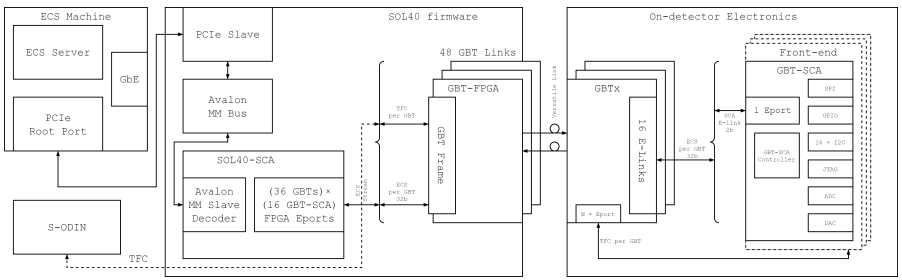


Figure: LHCb SOL40-SCA

- ALICE CRU is at the same hierarchical position as SOL40 in LHCb, however an SCA interface is not added in the CRU firmware, hence `pa3jtag` needed to be implemented as a pure software tool.

Summary 2/2

- Results
  - The effective configuration bit rate: ≈2.4 Kbps
  - Program time consumption ≈31 min
  - The rather long programming time is acceptable for the infrequent use, parallelisation is envisioned on FLPs
  - `pa3jtag` has been integrated into the ALICE ITS WP10 software tools
  - During Run 3, `pa3jtag` will only be executed in periods of no beam, e.g. technical stops $\Longrightarrow$ considered an ITS expert tool - not available to the central DCS shifter
- To be able to configure the PA3 on the RU ensures the robustness of the real-time data taking system which the ITS RU represents

Thank you.