FC7-testboard firmware definition

Stefano
Mersi, thomas.eichhorn, steven.lowette, mykyta.haranko, jelena.luetic, jarne.theo.de.clercq, andreas.muss
Alishba Kanwal

1 Introduction

In the context of Phase-2 Outer Tracker module production a variety of front-end prototype variants will not be produced and consequently the DAQ system will have to address them. In principle the test board should be able to control and readout front-ends with * 2×CBC3, 8×CBC, N×MPA * With/without CIC * With/without optical interface (GBTx, lp-GBTx)

If adequate DAQ system available, it can be used to address a large set of measurements/tests with these cards for * Component testing * System integration * Beam tests

Both the uncertainty on the availability of components and the need to test different readout chains suggest that a modular approach to the test board construction is favored.

2 Development process

The first stage of the development will be the definition of the /architecture/, in the following order: * definition of functional blocks with a coarse definition of functions * definition of the communication between blocks * precise definition of the blocks functions and interfaces

The second stage will be the actual development of VHDL code for the various blocks * creation of a github repository * creation of a vivado project importing the FC7 system firmware * creation of (empty) blocks in the project

It should be possible to switch between alternate blocks in the same structure by switch available at compilation time. Instruction on compiling the project along with the definition of compiler switches should be provided in the README.md file, so that it is accessible from the GitHub main project page.

Contributors should always follow the git workflow, merging the main branch locally and solving conflicts before creating a pull request. Tagged versions should be used in production, even if the master branch should always be compiling, functional. No “development” branch is suggested: every contributor will manage their development independently and only request pulls for code versions which where tested locally.

3 Top-level block definition

This should be configurable so that we can read-out either one or more modules (in case of optical interface). It relies on the standard FC7 system firmware for: IPbus, Memory access, etc.

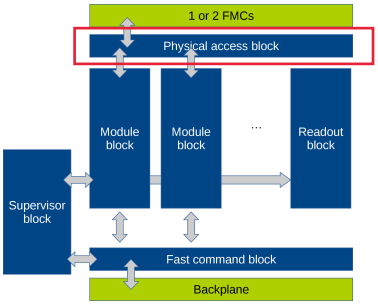

Figure 1: Replace this text with your caption

Figure 2: Zoom on the module block

For the purpose of this document we define **modes** as working modes that can be switched at run time and **implementations** as working modes that can be switched at compilation time. not all of the working modes or implementations described here are needed for the initial version of the firmware to be functional. in particular: the **standard mode** is required (but not the **hit counting**) and only the implementations dealing with electrical, non-CIC, 2×CBC2 hybrid is needed first.

Module and readout blocks should implement two modes of operation: **standard mode** and **hit counting mode** and they should be configured via the same register to avoid mismatches.

3.1 Physical access block

This block has two (main) **implementations**. **Optical implementation**: wrapping around N×GBT-FPGA blocks connecting to a COTS FMC with a number of SFP+ cages and **Electrical implementation** connecting to the custom-made interface FMC. Each custom-made FMC interface will need a different implementation of the Physical access block.

The electrical access block implements also an I^2C master and presents the same interface as the optical to the module block.

Interface to the module block: (t.b.d.)

3.2 Module block

Module block Transmits slow and fast commands Receives and unpacks data Syncs trigger and stub data Can forward stub detection to fast command block Sends data to the readout block Can pre-process hits data for calibration (hit or stub counting)

The module block is split in the following blocks: * CIC unpacker: (if needed) splits stub & hits streams, otherwise ˜ passthrough * Chip data align * Hit processing: * Hit count can be activated to count hits (or clusters or stub) per strip locally * Always sends stub to trigger with fixed latency * When requested, queues data to readout block

3.3 Readout block

Data mode: Manages memory readout and back-pressure Optionally could stream data to SFP+ connectors on the FMC (iif optical readout is present, one connector can be dedicated to fast streams) Data formatting for the DAQ, according to payload specs Performs post-scaling (to exercise fast trigger rates with limited bandwidth) Count mode: Used for commissioning: holds hit/cluster/stub counts per strip Takes care to add counters properly – as many counters as conditions (e.g. thresholds)

3.4 Fast command block

Recovers clock and trigger from backplane Can generate clock and periodic triggers Holds a trigger counter: it can be programmed to accept next N triggers and then hold the next incoming triggers Can issue a trigger upon reception of a stub from modules stefano test 123

3.5 Supervisor block

Holds a local stack of operations to be executed sequentially: I^2C transactions for F.E. (via module) Generate (or accept next) N triggers Push hit counters to appropriate memory location Significant reduction of IPBus transactions needed for a calibration operation (à la pixel DTB)