MiniDAQ Simulation Framework Environment

Guillaume Vouters
On behalf of the MiniDAQ developers

12 December 2013
AMC40 firmware meeting

Outline

1. Simulation framework environment
    - FORGE
    - GIT
2. Simulation framework
3. Simulation Tools
4. DEMO
5. Documentation

Simulation framework environment

In projects involving many people, it's important to be able to share work efficiently between all developers. But it's also important to be able to version and to track the different developments to avoid loss of work or misunderstanding

In this purpose we have decided to use two programs to help us :

A "FORGE" to gather all the documentation and information about the project
- Documentation listing
- Twiki
- Tracking bug
- News

A versioning program GIT to have a complete history and versions of the developments

FORGE

A documentation has been written to help people to get used to the FORGE
(free access)
https://lbredmine.cern.ch/projects/readout-system-upgrade/wiki/Wiki

To be able to work with the FORGE, users have to register to the CERN e-group:
- lhcb-web-lbredmine@cern.ch

To find it: https://e-groups.cern.ch

GIT

GIT is a distributed revision control program which allows the users to have a complete history of the development.

To be able to have read access to GIT, users have to register to the CERN e-group:
- lhcb-web-lbredmine@cern.ch

To find it: https://e-groups.cern.ch

If users want to have write access, they need to contact Pierre Yves Duval: duval@cppm.in2p3.fr
And ask for a write access according the sub-detector you work for. Do not forget to give him the name of the sub-detector you work for.

GIT

A documentation has been written to help people to get used to GIT (installation of GIT - first step – tuto to use GIT and the scripts …) which is available on our FORGE : https://lbredmine.cern.ch/documents/6

The idea is to provide to the collaboration a very easy to use framework. In order to do this, we have developed scripts to avoid the users wasting time configuring environment in their computer

- Scripts for ModelSim (to launch a Simulation)
- Scripts for Quartus (to launch a Compilation)

Purpose of Simulation

- Test different data format and select those which are possible to implement.

- Give the possibility to all the collaboration to be able to test and verify in close condition to reality the AMC40 firmware developments

→ what is developed is not a emulation, this is real and synthesizable code

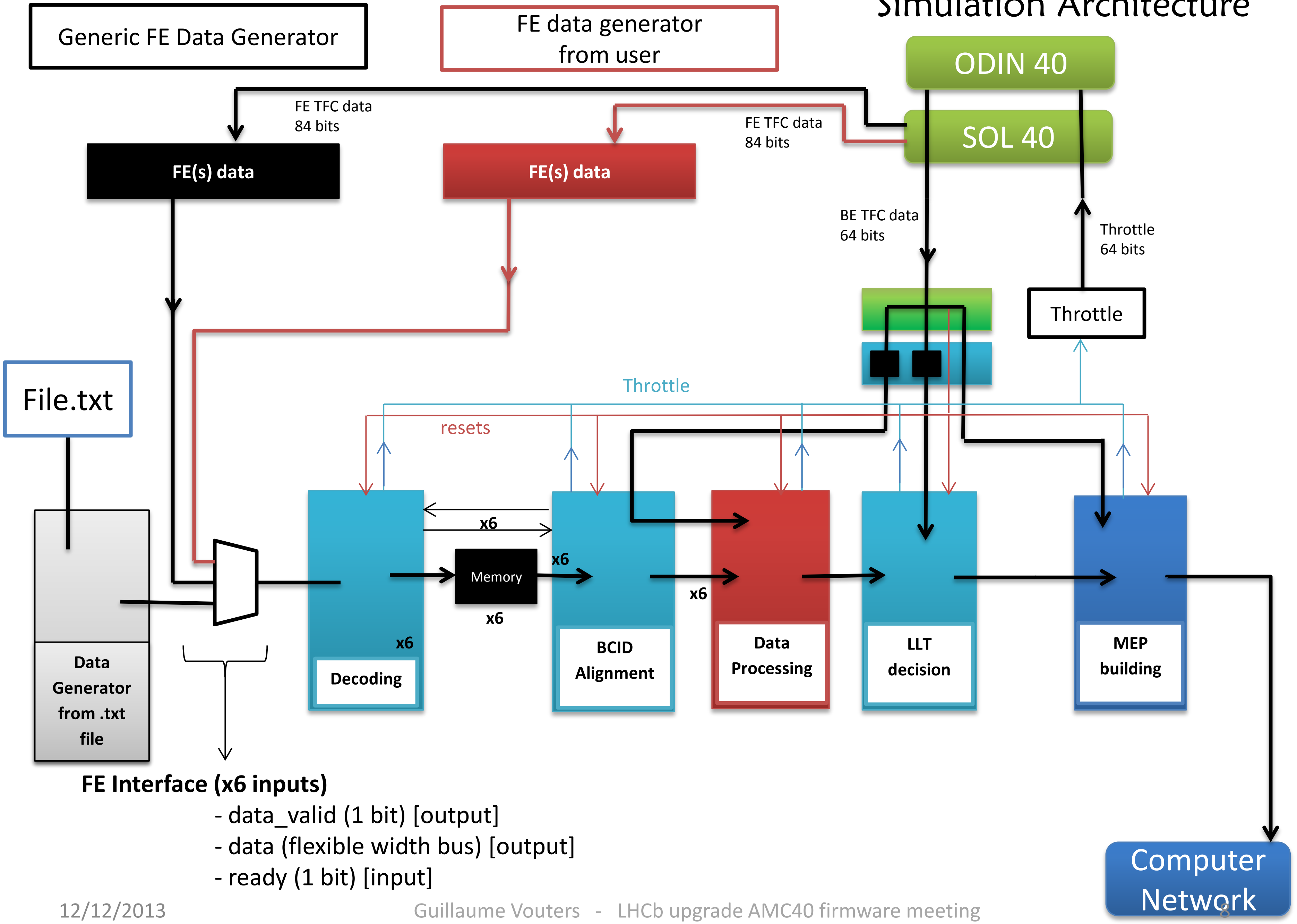
- Give the FE groups a way to verify the compatibility of their FE developments with the rest of the system (TFC and TELL40)

- Give the FE groups a way to develop and test their data processing

We have better chance to a firmware working well in the MiniDAQ if all the developments are checked before in simulation

Simulate the AMC40 and FE firmware developments in the simulation is not a possibility, it's mandatory !

Simulation Architecture



FE Interface (x6 inputs)

- data_valid (1 bit) [output]
- data (flexible width bus) [output]
- ready (1 bit) [input]

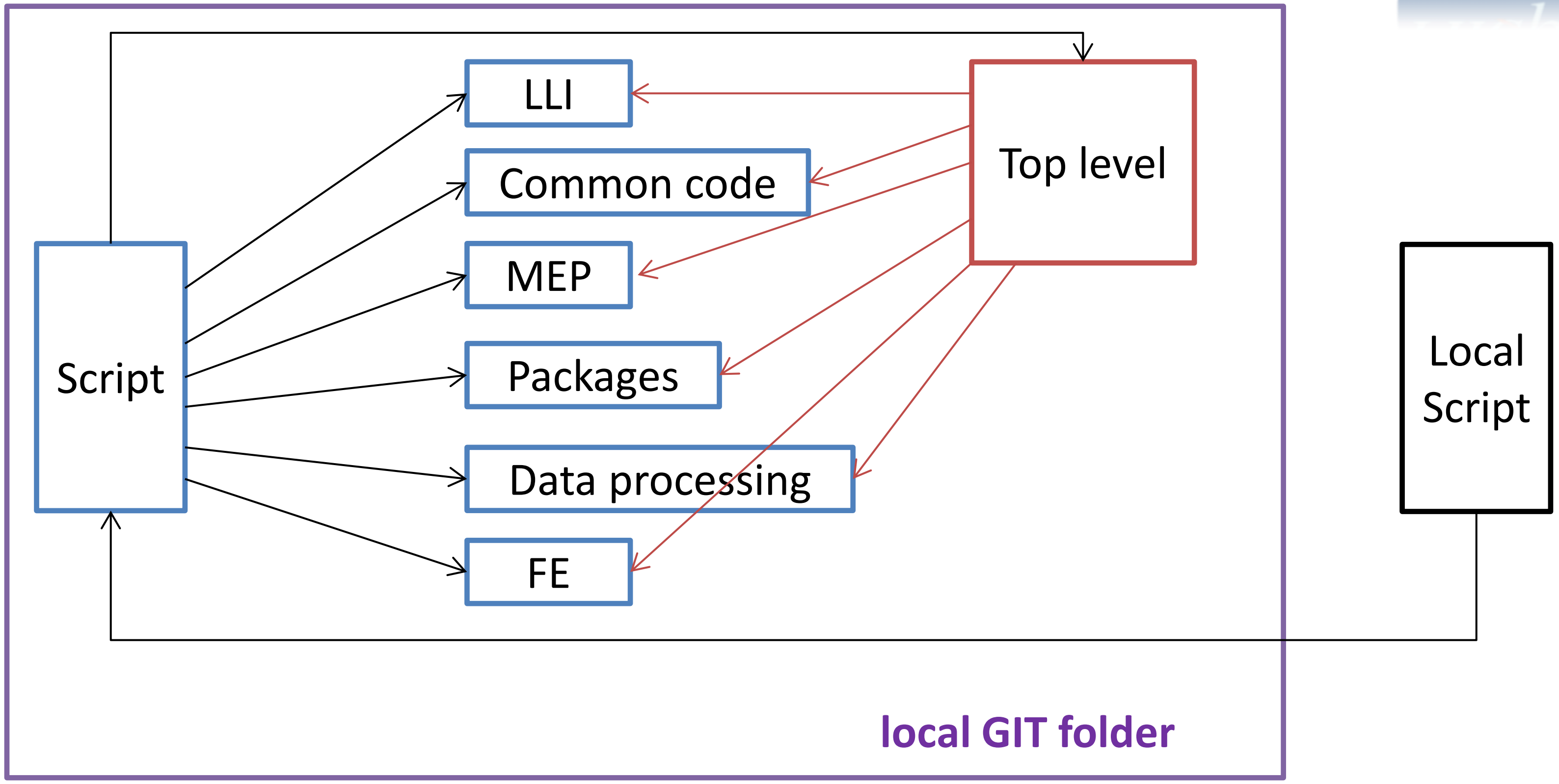
Constraints

- Common blocks and specific blocks
- Specific blocks from different sub-detectors

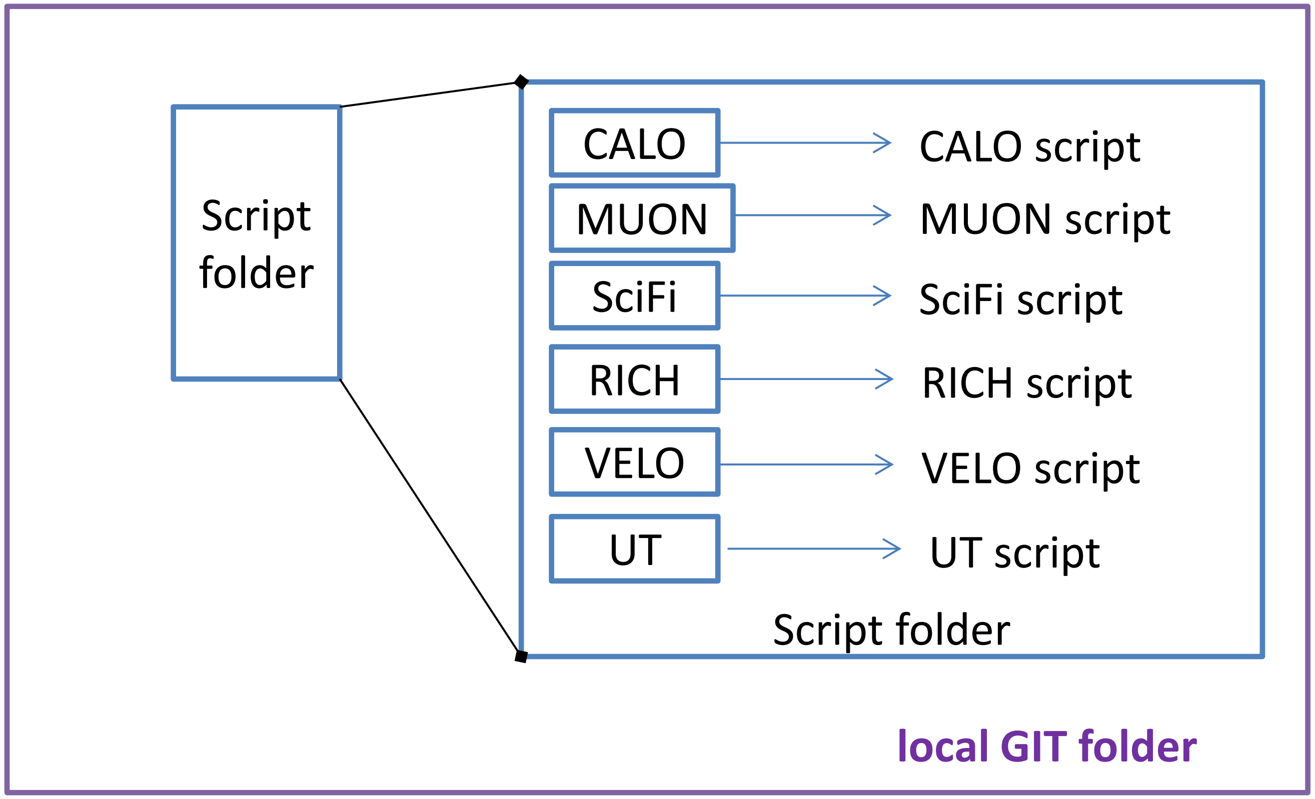
- One simulation for everybody
- Common file not duplicated

- Easy environment to use

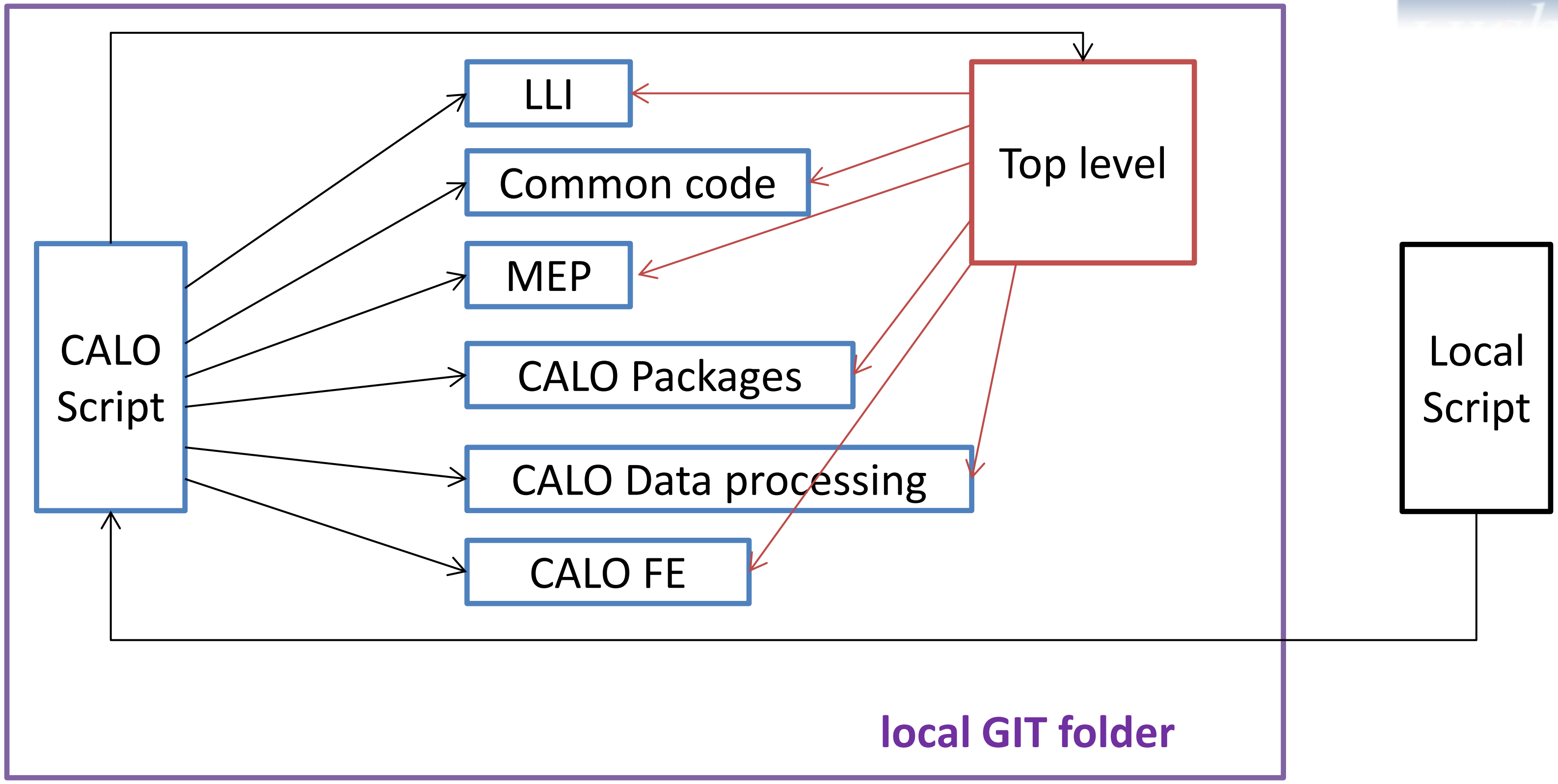
Scripts


LLI
Common code
MEP
Packages
Data processing
FE
Top level
Script
Local Script
local GIT folder

Scripts


Script folder
CALO
MUON
SciFi
RICH
VELO
UT
CALO script
MUON script
SciFi script
RICH script
VELO script
UT script
Script folder
local GIT folder

Scripts for CALO



Same architecture for each sub-detector

GIT arborescence

mini_daq/
…………..data_processing/
…………………………………..generic
…………………………………..specific
…………..generic_front_end
…………..low_level_interface
…………..mep_building
…………..packages
…………..scripts/
……………………..modelsim
……………………..quartus
…………..sub_detector_front_end
…………..tell40_common
…………..tfc
…………..top_level

In some folder like

- data_processing/speficic
- sub_detector_front_end
- packages
- script/modelsim
- script/quartus

You will find one folder by sub-detector group:

- CALO
- Central_tracker
- Inner_tracker
- Muon
- Outer_tracker
- Rich
- Upstream_tracker
- Velo_pixel

Developers have to work in the folder they belong to.

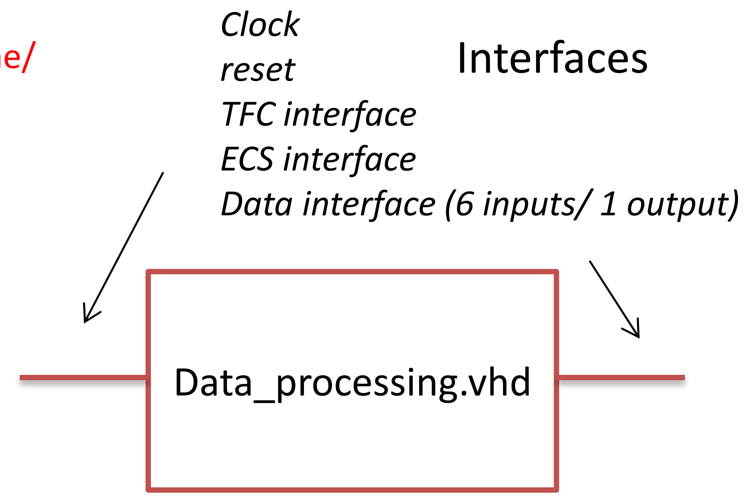
GIT arborescence

```
mini_daq/
……………data_processing/
……………………………………….generic
 ……………………………………..specific/sub detector name/
……………generic_front_end
……………low_level_interface
 …………..mep_building
……………packages/sub detector name/detector_constant_declaration.vhd
 …………..scripts/
………………………..modelsim/sub detector name/mini_daq_simulation_script.tcl
 ……………………..quartus/ sub detector name/mini_daq_compilation_script.tcl
……………sub_detector_front_end
……………tell40_common
 …………..tfc
 …………..top_level
```

See Federico's talk about detector_constant_declaration.vhd

FE developments

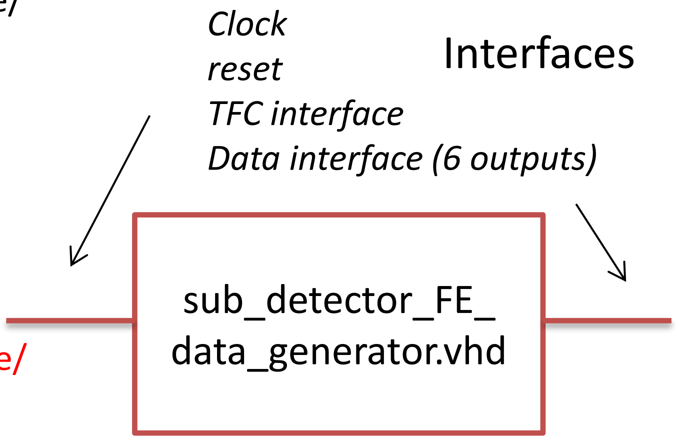
mini_daq/
……………data_processing/
………………………………………generic
………………………………………specific/sub detector name/
……………generic_front_end
……………low_level_interface
……………mep_building
……………packages
……………scripts/
…………………………modelsim
…………………………quartus
……………sub_detector_front_end
……………tell40_common
……………tfc
……………top_level



Name and interface of the hdl file has to be kept.
Use this file as a top level for your developments.

FE developments

```
mini_daq/
…………..data_processing/
……………………………………generic
 ……………………………………specific/sub detector name/
…………..generic_front_end
…………..low_level_interface
 ………….mep_building
…………..packages
 ………….scripts/
……………………….modelsim
 ……………………..quartus
…………..sub_detector_front_end/sub detector name/
…………..tell40_common
 ………….tfc
 ………….top_level
```

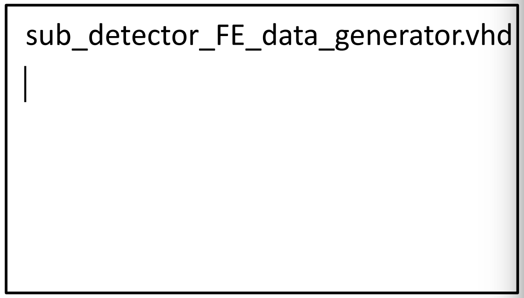


Name and interface of the hdl file has to be kept.
Use this file as a top level for your developments.

hdl_filelist.txt files

mini_daq/
……………data_processing/
………………………………………generic
………………………………………specific/sub detector name/
……………generic_front_end
……………low_level_interface
……………mep_building
……………packages
……………scripts/
…………………………modelsim
…………………………quartus
……………sub_detector_front_end/sub detector name/
……………tell40_common
……………tfc
……………top_level

sub_detector_front_end_hdl_filelist_simulation.txt



sub_detector_FE_data_generator.vhd
|

Always keep an empty line and just one at the end of the .txt file

In the folder :
sub_detector_front_end/sub detector name/ sub_detector_FE_data_generator.vhd

hdl_filelist.txt files

mini_daq/
……………data_processing/
…………………………………………generic
…………………………………………specific/sub detector name/
……………generic_front_end
……………low_level_interface
……………mep_building
……………packages
……………scripts/
…………………………modelsim
…………………………quartus
……………sub_detector_front_end/sub detector name/
……………tell40_common
……………tfc
……………top_level

sub_detector_front_end_hdl_filelist_simulation.txt

```
sub_detector_FE_data_generator.vhd
sub_detector_feature_X.vhd
|
```

Always keep an empty line and just one at the end of the .txt file

In the folder :
sub_detector_front_end/sub detector name/ sub_detector_FE_data_generator.vhd
sub_detector_front_end/sub detector name/ sub_detector_feature_X.vhd

What you need !

The simulation has to be done with the program Modelsim with a version which can deal with both vhdl and verilog source files like:

- Modelsim SE 10.1x (tested with Modelsim 10.1c and 10.1d)
- Modelsim Questa (tested with Questa)

Modelsim Altera version or Modelsim student edition version are not working for this simulation

We saw the theory, now let’s do practice !

DEMO

Documentation

→ LHCb upgrade MiniDAQ handbook (on the FORGE, soon on CDS)
https://lbredmine.cern.ch/documents/8

What is the MiniDAQ?
Framework environment : GIT and FORGE
Simulation of the MiniDAQ firmware

- Simulation Architecture
- Injection of FE data in the simulation
- Type of FE data encoding algorithms from the Generic FE data generator
- Tutorial to launch the simulation
- Simulation parameters
- In which files FE developments must be performed
- How to deal with bugs or new improvement?

Documentation

---------------------------- Framework --------------------------------

→ FORGE documentation (on the FORGE)

→ GIT documentation (on the FORGE)

---------------------------- Data format --------------------------------

→ Data format documentation (on the FORGE)

---------------------------- LHCb Upgrade --------------------------------

→ System-level Specifications of the Timing and Fast Control system for the LHCb Upgrade (on CDS)

→ Electronics Architecture of the LHCb Upgrade (on CDS)

Conclusion

You have everything to start ☺

The simulation framework is available now for sub-detector developers.

A lot of documentation have been developed to help you to use it.

If you have some trouble or if you find bugs in the simulation, DO NOT contact us directly, use the tracking bug program called “New issue” in our FORGE
→ https://lbredmine.cern.ch/projects/amc40/issues/new

To know who to contact when you create a new issue, have a look at section 3.8 in the LHCb upgrade MiniDAQ handbook.