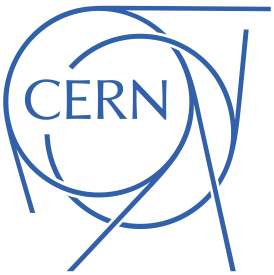
Summer Student Project Report

Extension of the ATLAS Level-1 Trigger Menu Testing Framework



Janek Both

Supervised by Ondrej Penc and Lorenzo Sanfilippo

20th September 2024

1 Introduction

At the ATLAS experiment [1] at the Large Hadron Collider, proton bunches are collided at a rate of 40 MHz. Recording data at this rate would not be manageable. In order to lower the data flow while keeping the interesting events, a two stage trigger system [2] is used. It reduces the event rate from 40 MHz to 100 kHz and then further to around 3 kHz. The first stage of the trigger system is the hardware-based Level-1 (L1) Trigger. It receives inputs from the calorimeter and the muon spectrometer and decides in 2.5 µs whether to keep an event or discard it. The decision is formed based on the Level-1 Trigger Menu, which refers to the physics configuration of the trigger.

After any changes of the physics selection requirements, the new Level-1 Trigger Menu needs to be tested by the Level-1 Central Trigger (L1CT) group using simulations and hardware replica of the Central Trigger at ATLAS. The test is performed using a Python based framework, whose improvement and extension was the goal of my project. This report is structured as follows: In section 2, a short overview of the Level-1 Central Trigger system is given, while section 3 introduces the testing methods of the Trigger Menu. In section 4, the incorporated extensions of the `TriggerMenuTest` package are presented. A short summary is given in section 5.

2 The ATLAS Level-1 Central Trigger

Figure 1 shows a simplified overview of the ATLAS Level-1 Trigger system [3]. The Level-1 Accept (L1A) decision is formed by the Central Trigger Processor (CTP) based on inputs from the Level-1 Muon Trigger (L1Muon), the Level-1 Calorimeter Trigger (L1Calo), the Topological Trigger Processor (L1Topo) and various forward detectors. The different subsystems provide information about the multiplicities of electron, photon, tau, jet and muon candidates along with flags for total scalar transverse energy, missing transverse energy and total jet scalar transverse energy.

The MUCTPI (Muon to CTP interface) is used to combine the information from the different muon trigger sectors while removing double counting of candidate tracks that traverse overlapping trigger sectors. The output of the MUCTPI is passed to the CTP, where the L1A decision is formed based on the trigger inputs and the Level-1 Trigger Menu. The menu is composed of up to 512 trigger items, each representing an event type that is accepted. The trigger items can be defined by individual trigger inputs or their combination. For example, the item L1_EM7[x1] requires at least one electron or

photon with a transverse energy above 7 GeV. When the conditions of any trigger item are fulfilled, the event is passed to the second trigger stage (the high level trigger), i.e. the L1A signal represents the logical OR of all trigger items. The Trigger, Timing and Control (TTC) system is used to fan out the trigger signals along with other CTP outputs to the read-out and detector front-end systems [3].

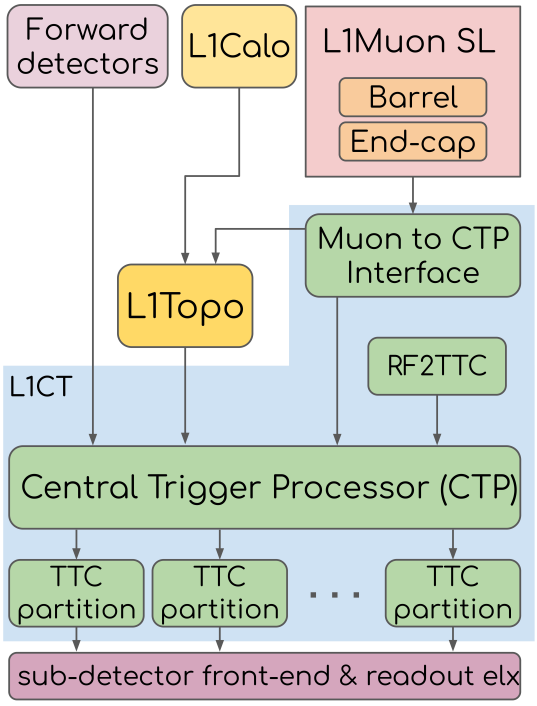


Figure 1: Overview of the ATLAS Level-1 Trigger system [4].

3 The Level-1 Trigger Menu Testing

Given the central role of the trigger system for the operations of the ATLAS experiment, every new Trigger Menu is tested before it is used at ATLAS Point 1. The trigger menu test is conducted using hardware replica of the Level-1 Central Trigger and simulation software. Figure 2 shows an overview of the components involved in the test and how they are connected. The testing procedure consists of several steps: Initially, the menu must be compiled to produce binary configuration files that are readable by the L1CT electronics. The resulting files must then be stored in the trigger database located at Point 1 and be accessible by the run control applications. Next, the input patterns for the test are generated. This involves processing the menu again and creating of binary files that represent individual tests of all trigger items. Finally, the test is performed and compared against the simulation. The framework captures the results and provides a

detailed printout of the individual tests as well as a summary of the outcomes. The trigger menu testing consists of four different logical parts which inspect different connections (see figure 2) and usually just run after each other, but can run independently:

- PIT: test of the electrical inputs to the CTPIN
- DIR: test of the direct electrical inputs to the CTPCORE
- OPT: test of the direct optical inputs to the CTPCORE
- MUO: test of the MUCTPI configuration.

The test can be done either as a full hardware test, where the output of the simulation and hardware parts are compared or as a simulation-only test. The latter does not compare the simulation outcome to the hardware, but still checks if the menu items are triggered in the expected trigger lines of the CTP in the expected time frame.

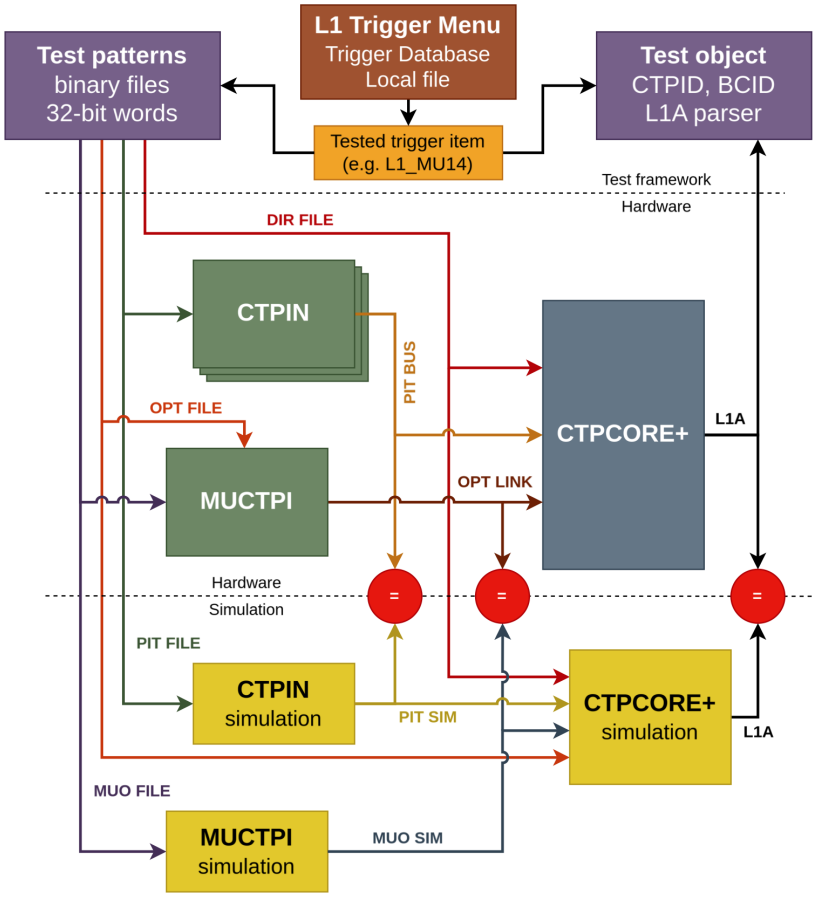


Figure 2: Overview of the different components that are used in the Level-1 Trigger Menu test. The colored connections indicate the flow of information of the different tests (PIT, DIR, OPT, MUO). [5]

4 Extensions of the TriggerMenuTest Framework

4.1 The new pattern generation

The generation of the hardware patterns for the test represents a central component of the menu test procedure. The pattern generation involves parsing the trigger menu and a subsequent derivation of binary files. The original pattern generation was written in C++. As the remainder of the testing package is written in Python, which is easier to read and to debug, the C++ code was first rewritten in Python and furthermore extended to allow for tests of higher multiplicity trigger items.

The new code is based on a single class named `CtpPlusTest`, which implements the pattern generation for all four types of the menu test as well as methods to save the CTP files after their download from the database. In order to generate the testing patterns, an object of the class must be instantiated. The instance accepts the L1 Trigger Menu as an attribute as well as a binary variable called `optimise`, which determines whether the pattern generation should be run in an optimised version or not. In the latter case, all higher multiplicity combinations of the patterns are tested, while in the optimized case only the lowest and highest multiplicity patterns are generated.
Unlike the old code, the new code permits the pattern generation for higher multiplicity tests of items which use the optical inputs. The higher multiplicity test is done in addition to the items' regular multiplicity test. In the optimized case, an additional test, corresponding to the maximum possible multiplicity for the available bits, is added for the subsequent bunch crossing. In the non-optimized case, patterns for all possible multiplicities are generated and tested in consecutive bunch crossings.

The methods of the `CtpPlusTest` class that implement the generation of the testing patterns all have a similar structure: They take an instance of the `IOManager` class, one or several basenames for the output files and they allow to only generate patterns for a specific item from the menu. For the latter, the name of the item must be passed via the `input_item` option. The following lines of code show an example of how to generate testing patterns for the OPT test:

```
1 CtpPlusTester = CtpPlusTest(L1Menu, optimise=True)
2 CtpPlusTester.dumpOptPattern(io,
3                     outfile_basename="pattern_opt",
4                     input_item=None)
```

The new code for the pattern generation is integrated into the testing package and is employed in parallel to the C++ code. To ensure that the new code is functioning as intended, the generated files are automatically compared against the C++ output after the pattern generation.

4.2 Integration of the Phase-2 Menu compiler

The Trigger Menu Compiler (`TMC`) is used to translate the L1Menu into CTP hardware files and is employed during the compilation step of the testing procedure. In view of the requirements during Phase 2, a Python-based Trigger Menu Compiler 2 (`TMC2`) was developed by the L1CT group, which we integrated into the testing procedure.

The new compiler is operating subsequently to the existing one during the compilation step. This does not only include running the compilation, but also combining and converting several files as well as getting some files from the old compiler. This is required because there will be no CTPIN in Phase-2. During the pattern generation part, an additional folder with the patterns that include the `TMC2` files is created and can be used to test the trigger menu.

To use the files generated by the new compiler in the test, one has to set `tmc2` variable to true in the config file or to provide a flag to the main executable of the package, the `TriggerMenuTestScript`, when running the test:

```
TriggerMenuTestScript dhre -c test_config.json --tmc2
```

After running the test with `TMC` and `TMC2`, one can compare the overall results of the two tests by using the following command:

```
TriggerMenuTestScript compare-compilers
```

This compares the test results item by item and generates a summary printout as well as a detailed printout for the tests in which the compilers yield different results.

4.3 Automation of the menu testing using Flask

As testing a new trigger menu can take up to 20 min a couple of times a week, it is beneficial to further automate the workflow. Given that the test always follows a similar procedure, the first idea was to fully automate it using a GitLab CI/CD pipeline. However,

difficulties were encountered when attempting to authenticate in the trigger database from the GitLab runner, which could not been resolved despite discussions with the IT department. Therefore, the Python-based web framework `Flask` was employed to automate the testing procedure.

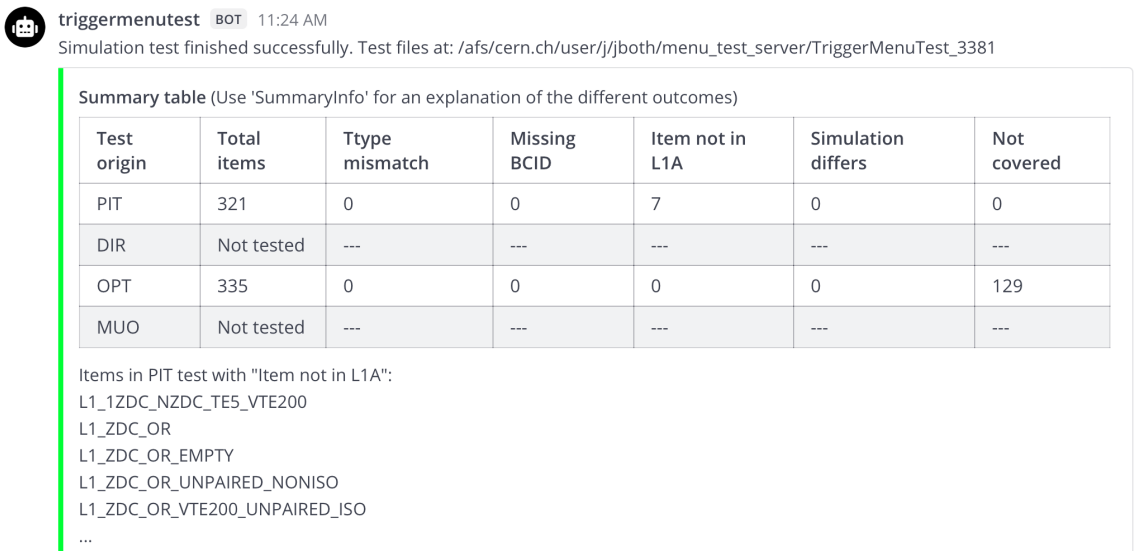
The web server can be launched from a Python script and run permanently by means of a `tmux` session on one of the computers in the lab of the L1CT group. The `tmux` session is refreshed regularly in order to guarantee the availability of a valid Kerberos and AFS access tokens. This is achieved by a cronjob, which regularly executes a bash script.
The web server can be addressed from Mattermost via an outgoing webhook and post in the channel using an incoming webhook. There are two Mattermost channels set up to communicate with the webserver: one for full scale testing by members of the L1CT team and another one for the menu team. The distinction between the two is that the latter is limited to conducting solely simulation-only tests, while the former also allows to do hardware tests.

In order to trigger a menu test, one has to provide the type of the test, i.e. hardware or simulation-only by using either the `HardwareTest` or `MenuTest` (`SimulationTest`) keyword. In addition, the super master key of the menu and the name of the CTPIn menu need to be provided. Optionally, one can specify the tests that are carried out (dir, pit, opt, muo), whereby all tests run by default:

In case the inputs (for example the SMK or the CTPIn menu) are not specified properly or if the test fails, an error message is send to the Mattermost channel. If the test was started successfully, a short summary of the test setup in the Mattermost channel is returned:

A new folder named `TriggerMenuTest_[smkey]` is created, in which all files of the test are stored. The usual testing procedure is then carried out, whereby the L1CT hardware

files are not uploaded to the trigger database at Point-1. Furthermore, the database is not accessed during the pattern generation. Instead, the locally compiled files are used in the test. As soon as the test is completed, a summary of the test results is send to the Mattermost chat:



triggermenutest BOT 11:24 AM

Simulation test finished successfully. Test files at: /afs/cern.ch/user/j/jboth/menu_test_server/TriggerMenuTest_3381

Summary table (Use 'SummaryInfo' for an explanation of the different outcomes)

Test origin	Total items	Ttype mismatch	Missing BCID	Item not in L1A	Simulation differs	Not covered
PIT	321	0	0	7	0	0
DIR	Not tested	---	---	---	---	---
OPT	335	0	0	0	0	129
MUO	Not tested	---	---	---	---	---

Items in PIT test with "Item not in L1A":
L1_1ZDC_NZDC_TE5_VTE200
L1_ZDC_OR
L1_ZDC_OR_EMPTY
L1_ZDC_OR_UNPAIRED_NONISO
L1_ZDC_OR_VTE200_UNPAIRED_ISO
...

If the test ran successfully, it is checked automatically if the L1CT hardware files are already available in the trigger database. If not, an email is sent to `atlas-l1ct-operations@cern.ch`, stating that the menu was compiled and that the files are ready for upload.

In addition to the keywords that trigger a menu test, there are other special keywords that can be used in Mattermost:

- `SummaryInfo`: Provides a detailed explanation of the summary table.
- `CheckUpload smkey=#`: Checks whether the CTP files are already uploaded to the database or not. In case they are not yet uploaded but available locally, an email is send to the l1ct operations email, informing them that the files are ready for upload.
- `CheckComputers`: Shows who is connected to the computers that are needed for the hardware test. It should be executed every time before running the hardware test to make sure to not interfere with anyone.
- `Help`: Displays an overview of the keywords available in the respective channel.

5 Summary

The ATLAS Level-1 Trigger Menu is an important part of the trigger configuration of the experiment and its testing is essential to ensure flawless data taking at Point 1. The test is done using the `TriggerMenuTest` Python package whose extension and enhancement was the goal of this project. During the summer, the package was updated significantly. This includes writing new code that generates the hardware patterns, incorporating the new, Phase-2 Menu compiler into the code, as well as fully automating the testing procedure using the web framework `Flask`. The automation allows that the entire test can be carried out easily and quickly from the Mattermost, which makes the test accessible to a broader group of people.

References

[1] G. Aad et al. (ATLAS), "The ATLAS experiment at the CERN Large Hadron Collider: a description of the detector configuration for Run 3", JINST **19**, P05063 (2024).

[2] G. Aad et al., "The ATLAS trigger system for LHC Run 3 and trigger performance in 2022", Journal of Instrumentation **19**, P06029 (2024).

[3] S. Ask et al., "The ATLAS central level-1 trigger logic and TTC system", JINST **3**, P08002 (2008).

[4] *Upgrading the Level-1 Central Trigger*, (2022) `https://ep-news.web.cern.ch/content/upgrading-atlas-level-1-central-trigger`.

[5] *ATLAS Level-1 Trigger Menu Testing*, (2024) `https://indico.cern.ch/event/1291157/contributions/5876929/`.

Acknowledgments

First and foremost, I want to thank my supervisors Ondrej and Lorenzo for their guidance and support over the summer. I also would like to thank the other members of the L1CT group and especially my office mate Emil for his helpful ideas. Finally, thanks go to the other summer students I got to know and with which I had a very good time outside of work, be it during coffee breaks or when hiking in the mountains.