The Trigger Menu Handler of the ATLAS Level-1 Central Trigger Processor

N. Ellis, P. Farthouat, G. Schuler, R. Spiwoks

CERN, 1211 Geneva 23, Switzerland

Abstract

The role of the Central Trigger Processor (CTP) in the ATLAS Level-1 trigger is to combine information from the calorimeter and muon trigger processors, as well as from other sources, e.g. calibration, and to make the final Level-1 decision. The information sent to the CTP consists of multiplicity values for a variety of (electron, gamma, tau/hadron and jet) transverse-momentum thresholds, and of flags for transverse-energy sum thresholds. The algorithm used by the CTP to combine the different trigger inputs allows events to be selected on the basis of menus. Different trigger menus have to be considered for different run conditions. In order to provide sufficient flexibility and to fulfil the required low latency, the CTP will be implemented with look-up tables and programmable logic devices. The trigger menu handler is the tool which translates the human-readable trigger menu into the configuration files necessary for the hardware. It stores several prepared configurations and down-loads them into the hardware on request. An automatic compiler of the trigger menu and a prototype of the trigger menu handler have been implemented.

I. Central Trigger Processor

The CTP receives trigger inputs from the calorimeter and muon trigger processors, and other sources, e.g. calibration. This trigger input consists of encoded multiplicity values for a variety of electron, gamma, tau/hadron and jet transverse-momentum thresholds, and of flags for transverse-energy sum thresholds. The CTP combines these values based on a trigger menu containing individually prescalable trigger items. The CTP [1,2,3] makes a Level-1 decision by taking the OR of all trigger items, adding deadtime in order to prevent the detector front-end electronics from reading overlapping events and buffers from overflowing. The Level-1 accept signal is sent to the detector front-end electronics using the timing, trigger and control (TTC) system.

The trigger menu consists of a set of trigger items. The current design allows a maximum of 96 trigger items. Each trigger item is defined by a combination of trigger objects, a mask, a priority to select between the two complex deadtime algorithms and a prescaling factor. The combination of trigger objects can be composed of a single trigger object or a combination of trigger objects using the logic operations AND, OR and NOT, or a combination thereof. A trigger object is defined by a condition on a trigger input, requiring that the trigger input be equal to or greater than a certain value. In most cases this simply corresponds to a required multiplicity value. The current design allows a maximum of 128 bits of trigger inputs. Other conditions on the bunch crossing identifier or the LHC turn counter, and gate criteria such as a general veto and priority for the deadtime, can be included. An example of an excerpt of a trigger menu is given in figure 1. The trigger objects are written as triplets of inclusive multiplicity value, trigger type and thresholds, e.g. “1MU6” means an inclusive single muon trigger with a threshold of 6 GeV. “EM” stands for electromagnetic clusters and “XE” for missing transverse energy..

```
1MU6              mask = ON, prio = LO, scal = 1000
2MU6              mask = ON, prio = HI , scal = 1
1EM20 AND XE20    mask = ON, prio = LO, scal = 1
...
```

Figure 1: Example of a Trigger Menu (Excerpt)

Different trigger menus for different run conditions have to be considered: menus for phsyics, cosmic ray and calibration runs; menus for high-luminosity and low-luminosity; and menus for the initial commissioning phase and for stable running. The trigger menu used for the CTP can and will often change from one run to the next..

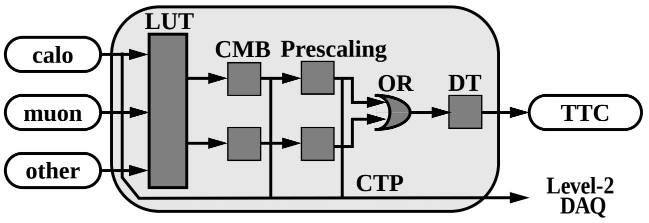


Figure 2: Central Trigger Processor

In order to provide flexibility and to fulfil the timing requirement for the Level-1 trigger, the CTP design is based on look-up tables and programmable logic devices, see figure 2. The look-up tables (LUT) are implemented with static random-access memories (SRAM). The programmable logic devices are implemented with both field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs). The FPGAs are based on SRAM technology and can be programmed easily and almost as many times as desired. The CPLDs are based on electrically erasable, programmable read-only memory (EEPROM) technology which allows them to be programmed in the system (in-system or in-situ programming, ISP) a few thousand up to a few ten thousand times,; depending on their type

II. Trigger Menu Handler

The use of programmable logic devices has moved the problem of modifying the functionality of a given hardware system into the area of software. Different configuration[1] files will be generated for the different trigger menus. They will be loaded into the hardware of the CTP together with other configuration parameters needed for the functioning of the CTP. The flexibility to handle many different trigger menus requires fast and reliable generation and loading of the configuration files belonging to a trigger menu.

The trigger menu handler deals with information for the hardware configuration, the trigger menu, and the CTP running parameters. The hardware configuration describes the internal hardware of the CTP and will not change after the design is finished, except in cases when the wiring of the trigger inputs to the CTP are modified. The trigger menu defines a set of different trigger items which are logical combinations of the trigger inputs to the CTP, some internal signals and some gate criteria. The trigger menu handler generates the configuration files necessary for the functioning of the look-up tables and the programmable logic devices. The CTP running parameters comprise all other parameters necessary for the functioning of the CTP.

The trigger menu handler has to perform the following different functions, which are schematically shown in figure 3:

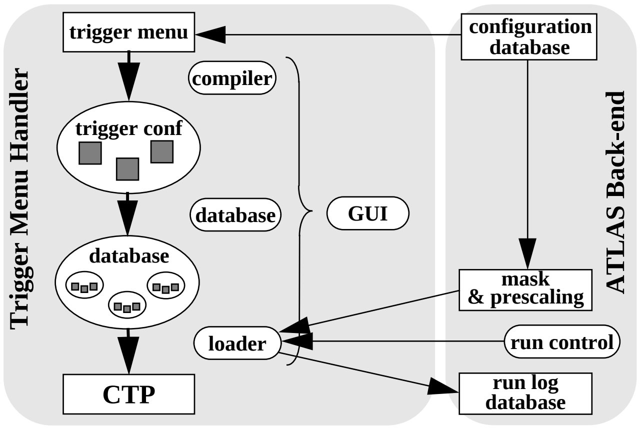


Figure 3: Trigger Menu Handler

- The compiler translates the trigger menu taken from the run configuration database into the configuration files required by the hardware[2]. This compilation will most likely have to be done in several steps, including proprietary compilation tools which are dependent on the hardware chosen for the CTP.
- The database manages a pool of different trigger menus and their related configuration and other files. Operations to be performed include the creating, copying, and deleting of trigger menus as well as selecting a menu for subsequent operations, e.g. compile and load.
- The loader takes the configuration files for the selected trigger menu, adds other configuration parameters which do not require compilation of the trigger menu, such as masks and prescaling factors, and loads the files and parameters into the hardware. This will most likely require the use of proprietary tools and might be based on standard buses like JTAG and/or VME. The loader is also responsible for updating the run-parameter database.
- The graphical user interface (GUI) combines the different functions mentioned above and allows the user in stand-alone operation easy access to the trigger menus and the different operations on them.

1. In this paper we will use the word “configuration” for the programming or configuration of SRAMs, FPGAs and CPLDs.
2. The compiler can also provide files for functional and/or timing simulation of the hardware.

III. Prototype Work

A prototype implementation of the trigger menu handler was carried out in order to test the principle and to investigate the issues involved. Another purpose of the trigger menu handler at this stage of the design was to provide the possibility to easily generate and test different hardware configurations. The main issue of the exercise was to automatize the compilation of a trigger menu and to investigate the flexibility that can be achieved. The main part of the prototype implementation is the actual compiler. The database and GUI are implemented in a rudimentary way to provide a frame for the compiler. The loader is not implemented as there is no final hardware yet.

The prototype trigger menu handler is based on the trigger menu implementation presented in reference [4] which uses the example trigger menu presented in the Level-1 TDR [1]. The menu is an example of a rather comprehensive menu using 86 out of 96 possible items, including many redundant items for monitoring and trigger efficiency calculation. The hardware configuration used to develop the compiler is based on 11 look-up tables and 2 programmable logic devices. Other configurations can be foreseen, in particular one in which the trigger inputs are connected directly to the combinatorial logic devices without using look-up tables. For the programmable logic devices, CPLDs from Lattice Semiconductor Corp. [6] were chosen. Other devices could be chosen in which case a part of the compiler would have to be modified in order to use different proprietary tools.

A. Compiler

The compiler of the prototype implementation is targeted at the CPLDs of Lattice Semiconductor Corp. Compilation is actually achieved in three different steps:

1) The compiler program translates the trigger menu from a text format into register-transfer descriptions of the combinatorial logic devices using VHDL.

2) The synthesizer, Leonardo Spectrum from Exemplar Logic, Inc. [7], synthesizes the VHDL code to a gate-level descrip-

tion using gates from a technology-dependent library. The output is a netlist file in EDIF format.

3) The place-and-route tool, ispEXPERT Compiler from Lattice Semiconductor Corp. [6], takes the gate-level netlist and places and routes it onto the chosen device (ispLSI5384V-125LB388). On output, it generates a configuration file ("fuse map") for the in-system programming of a device in JEDEC format. It further generates a VHDL and SDF file which can be used for functional and timing simulation.

A file for use with the make utility was written which combines the three successive steps and automatically produces the configuration files from the trigger-menu text file. For the synthesizer and the place-and-route tool, "batch" versions of the tools are used which do not require interactive input, but contain the necessary information in project files and command line options. While for the second and third steps of the compilation proprietary tools could be used, and in case of the place-and-route tool had to be used, for the first step a compiler program had to be developed specially for this purpose.

The compiler program first parses the trigger menu and creates a list of all trigger items. The combinations of trigger objects (AND, OR, and NOT) for each trigger item are optimized using a heuristic approach in order to minimize the number of trigger objects used in each of the combination. Then, the hardware description is parsed and lists of all input signals, look-up tables, combinatorial logic devices and their connections are created. When all the necessary information is available the program loops over all the trigger objects and finds the corresponding input signals or look-up table signals and decides on the coding scheme to be used for each of them. After that, the trigger items are partitioned over the available combinatorial logic devices. Finally, for each combinatorial logic device, VHDL code is produced by looping over all the trigger items and their corresponding trigger objects. The source code is mostly written in C with some part written in C++. It uses the GNU flex utility for lexical scanning, the GNU bison utility for syntactical parsing, and the C++ standard template library for container objects.

B. Database

For the database of the prototype implementation, a very simple approach was chosen. The file system of the CERN IT/CE server is used to house all the necessary files for the different trigger menus. Each trigger menu resides in its own dedicated sub-tree of directories. On the highest level of this subtree there are the trigger menu, the hardware configuration, a description file, and the log files from the different steps of compilation, as well as all the other files necessary to perform the actual compilation (e.g. the file for the make utility). The individual configuration files for each of the CMBs are in separate sub-directories of this tree. These sub-directories contain all the necessary project files for the different compilation tools, as well as the VHDL, the EDIF and the JEDEC files for each of the CMBs.

Access to a trigger menu is simply achieved by navigating in the directory structure. The different trigger menus are accessed as directories at the top-level. No information on the relationship between the different files is stored other than the fact that the files reside in the same trigger menu directory. Status and description of a trigger menu are written into a description file, but no consistency checks are performed. No authentication mechanism is implemented. Similarly, a logging mechanism for the operations carried out on the trigger menus is not implemented. For the final database, a history of operation will be important.

This implementation is only a very simple example of how the trigger menus can be organized. It was not the focus of the prototype study. Integration with the run configuration database will have to be defined. Issues of data consistency and access authentication will become important for the final system when the trigger menus will be accessed by different users.

C. Loader

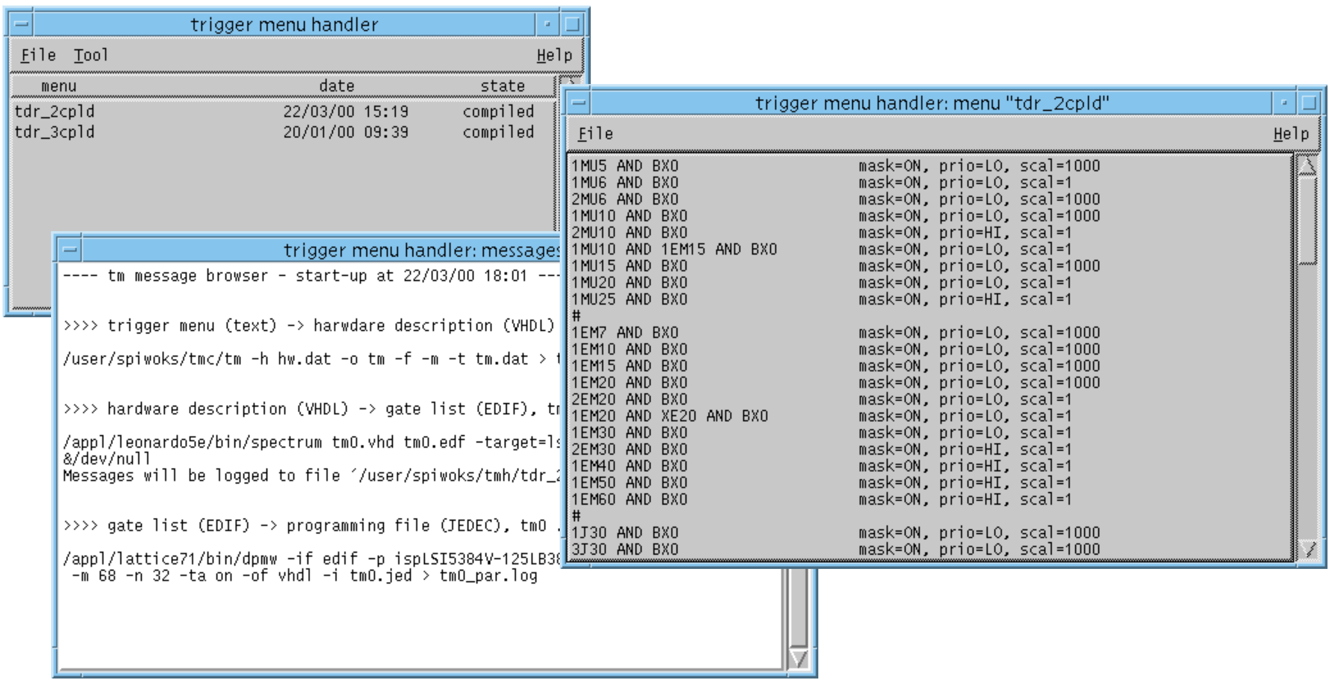
The loader is not implemented in the prototype because there is no final hardware to load the configuration files to. Independent of this prototype study, tests with a CTP demonstrator [8] have been carried out. The demonstrator uses CPLDs from ALTERA [9]. The JAM player is a program provided freely by ALTERA which can be used to load configuration files into ALTERA devices using the Bitblaster cable or JTAG port. A script can be written to specify all the necessary configuration files and options of the JAM player program. These tests show that the loading of configuration files can be carried out from a processor physically close to the CTP and that it can be automatized and carried out in a programmable way.

Like ALTERA and the JAM player, Lattice also provides code to load configuration files into their devices. The code is available in C and C++ and takes a JEDEC file which it loads into the CPLDs using the JTAG serial bus. This code can be used to write a program similar to the JAM player or can be integrated directly into any other control program for the CTP. For the final system the code can be used to communicate with the JTAG controller on the CTP using the JTAG port or via the VME bus. A script or program can be written to choose the necessary configuration files, the options and control the actual loading.

D. Graphical User Interface

For the prototype implementation a very simple GUI has been developed, which is shown in figure 4. It consists of a main window (upper left) and a message window (lower left). Also shown in the figure is a trigger menu (right).

The main window shows a list of available trigger menus. This window allows creating, copying and deleting of trigger menus. This should also be the place where tools like comparing trigger menus can be installed. The message window receives messages of all operations being called. Pop-up windows show if an operation was a success, or if an error occurred.


trigger menu handler
File Tool
Help
trigger menu handler: messages
trigger menu handler: menu "tdr_2cpld"
File
Help

```
menu                              date             state
tdr_2cpld                         22/03/00 15:19   compiled
tdr_3cpld                         20/01/00 09:39   compiled
```

```
---- tm message browser - start-up at 22/03/00 18:01 ---

>>>> trigger menu (text) -> harwdare description (VHDL)

/user/spiwoks/tmc/tm -h hw.dat -o tm -f -m -t tm.dat > t


>>>> hardware description (VHDL) -> gate list (EDIF), tr

/appl/leonardo5e/bin/spectrum tm0.vhd tm0.edf -target=ls
&/dev/null
Messages will be logged to file ´/user/spiwoks/tmh/tdr_2


>>>> gate list (EDIF) -> programming file (JEDEC), tm0 .

/appl/lattice71/bin/dpmw -if edif -p ispLSI5384V-125LB38
 -m 68 -n 32 -ta on -of vhdl -i tm0.jed > tm0_par.log
```

```
1MU5 AND BXO                    mask=ON, prio=LO, scal=1000
1MU6 AND BXO                    mask=ON, prio=LO, scal=1
2MU6 AND BXO                    mask=ON, prio=LO, scal=1000
1MU10 AND BXO                   mask=ON, prio=LO, scal=1000
2MU10 AND BXO                   mask=ON, prio=HI, scal=1
1MU10 AND 1EM15 AND BXO         mask=ON, prio=LO, scal=1
1MU15 AND BXO                   mask=ON, prio=LO, scal=1000
1MU20 AND BXO                   mask=ON, prio=LO, scal=1
1MU25 AND BXO                   mask=ON, prio=HI, scal=1
#
1EM7 AND BXO                    mask=ON, prio=LO, scal=1000
1EM10 AND BXO                   mask=ON, prio=LO, scal=1000
1EM15 AND BXO                   mask=ON, prio=LO, scal=1000
1EM20 AND BXO                   mask=ON, prio=LO, scal=1000
2EM20 AND BXO                   mask=ON, prio=LO, scal=1
1EM20 AND XE20 AND BXO          mask=ON, prio=LO, scal=1
1EM30 AND BXO                   mask=ON, prio=LO, scal=1
2EM30 AND BXO                   mask=ON, prio=HI, scal=1
1EM40 AND BXO                   mask=ON, prio=HI, scal=1
1EM50 AND BXO                   mask=ON, prio=HI, scal=1
1EM60 AND BXO                   mask=ON, prio=HI, scal=1
#
1J30 AND BXO                    mask=ON, prio=LO, scal=1000
3J30 AND BXO                    mask=ON, prio=LO, scal=1000
```

Figure 4: Prototype Trigger Menu Handler GUI

From the main window a trigger menu can be selected. The status information, including the description file and the log files from previous compilations can be read. The selected trigger menu can be edited, compiled and loaded (not implemented).

The GUI prototype is implemented using Tcl/Tk and scripts for the compilation. For the final system scripts for the loading will have to be added

IV. CONCLUSIONS

Automatic handling of trigger menus can be achieved. The prototype implementation of the trigger menu handler shows that the required flexibility can be obtained. There is, however, a dependence on proprietary tools which are usually used in the form of scripts. Most of the electronic design tools provided by vendors of combinatorial logic devices or by other vendors have the possibility to run them in “batch” (as opposed to interactive) mode. A pool of trigger menus will be prepared and compiled independent of the running of the experiment. The configuration files will be stored in the database of the trigger menu handler and will be part of the run configuration database together with the trigger menus. When the user wants to start a run he will have to select a trigger menu, add the parameters which do not require compilation, and load the prepared configuration files and the parameters into the CTP.

Open questions are the details of the CTP hardware architecture and the modifications of the trigger menu which will have to be expected. The existing trigger menu compiler will be used to investigate the best architecture and to choose components. Different trigger menus can be defined and tested with different options of the hardware configuration: with or without look-up tables, with or without masks and other gate criteria. The tests will be carried out by using simulation including timing information for the combinatorial logic devices. The simulation can be repeated for different combinatorial logic devices. In that case minor modification of the compilation scripts are required.

The prototype implementation concentrated on the actual trigger menu compiler. Loader, database and GUI are implemented only in a minimal way in order to provide a working environment. These will have to be improved for the final version of the trigger menu handler. The actual trigger menu compiler will be improved to include writing of the memory tables for the LUTs, to include optimization of the partitioning of items over the combinatorial devices, to clean up the dependency of the data structures and to improve the error handling of the parsers.

The trigger menu loader will be developed once the final hardware is available. The connection to the run database will be defined and developed in collaboration with the ATLAS data acquisition system developers. The database for the compiled configurations will be part of the ATLAS run database and will include authentication mechanisms. The loader will be under control of the ATLAS run control system.

V. REFERENCES

[1] ATLAS Collaboration, First-level Trigger Technical Design Report, CERN/LHC/98-14, June 1998, in particular chapter 15.

[2] Level-1 Central Trigger Processor: (User) Requirements Document (Version 3.0), ATLAS working document ATL-DA-ES-0003, November 1999.

[3] Central Trigger Processor Specification (Version 2.0), ATLAS working document ATL-DA-ES-0006, November 1999.

[4] Trigger Menu Implementation in the Level-1 Central Trigger Processor, ATLAS working document ATL-DA-ER-0001, November 1999.

[5] Trigger Menu Handler for the Level-1 Central Trigger Processor, ATLAS working document ATL-DA-ER-0002, February 2000.

[6] Lattice Semiconductor Corp., http://www.latticesemi.com, in particular the ispLSI 5000V family.

[7] Exemplar Logic, Inc., http://www.exemplar.com.

[8] I. Brawn et al., A Demonstrator for the ATLAS Level-1 Central Trigger Processor, 3rd Workshop on Electronics for LHC Experiments, London/UK, CERN/LHCC/97-60, September 1997.

[9] Altera Corp., http://www.altera.com.