Global Trigger firmware Specification for MP7 platform for Upgrade Phase I

Herbert Bergauer, Babak Rahbaran, Johannes Wittmann
Email: herbert.bergauer@oeaw.ac.at

Institute of High Energy Physics (HEPHY)

http://www.hephy.at
http://globaltrigger.hephy.at

September 3, 2021

Revision History

Doc Rev	Description of Change	Revision Date
6.0	New structure of document for firmware versions 1.12.x.	2021/02/10
5.7	Fixed typo in section "Invariant mass calculation for three objects" 4.4.12.1.6.	2020/12/03
5.6	Updated text in section "VHDL-Templates for VHDL-Producer" **??**.	2020/09/31
5.5	Inserted links to VHDL modules.	2020/09/18
5.4	Updated text in section "Correlation conditions" 4.4.12. Description is for v1.10.0 of Global Trigger Logic.	2020/09/17
5.3	Inserted description of "Invariant mass divided by delta R calculation" (see 4.4.12.1.5).	2020/09/10
5.2	Fixed typo (unconstrained pt).	2020/09/09
5.1	Inserted text for new muon structure in sections 4.2.2, 4.4.4 and 4.4.12.2. Added subsections in section **??**	2020/08/04
5.0	Additional text in section 4.4.12.2.2.	2020/05/25
4.9	Inserted text in section 4.4.6.1.2 for Calorimeter Overlap Remover conditions and 4.4.12.2.2 for Calo Calo Overlap Remover Correlation conditions.	2020/04/16
4.8	Updated text in sections **??**, **??** and 4.4.12 for changes which have been done for GTL VHDL version 1.8.0 (module names without version number, "five eta cuts").	2019/08/13
4.7	Inserted "Asymmetry" and "Centrality" of "Energy sums" (GTL VHDL version 1.6.0). Therefore updated sections 4.1, 4.2.1, 4.4.7 added section "Centrality condition" 4.4.10 and updated Table 2	2018/08/13
4.6	Updated text in section "Global Trigger Logic" (4) according to firmware version v1.5.0 of gtl_module.vhd	2018/02/21
4.5	Updated text in section "Framework" (3) according to firmware version v1.2.3 of frame.vhd	2018/01/19
4.4	New "icons" ET^{HF}_{miss} and HT^{HF}_{miss} in Table 2 and Section 4. Updated glossary (see Section 7)	2016/11/11
4.3	Updated table "μFDL register map" (27) and section "Register map" (5.4.1.1). Moved "List of Tables" and "List of Figures" to the end of document. Inserted link to "Scales for inputs to μGT" (4.3). Moved section "Software reset" to section "Framework" as subsection (3.7). Removed empty sections "IPBus", "Firmware Configuration" and "Bibliography"	2016/11/03

Doc Rev	Description of Change	Revision Date
4.2	Updated sections "Calo-Layer2 optical interfaces" (4.2.1) and "Energy sum quantities conditions" (4.4.7) for towercount trigger bits. Inserted section "Towercount condition" (4.4.9)	2016/10/25
4.1	Updated section "Calo-Layer2 optical interfaces" (4.2.1) for new energy sum quantities and minimum bias trigger bits. Updated sections "Firmware" (2), "Framework" (3) and "Final Desicion Logic" (5).	2016/06/09
4.0	Updated Text in section "Muon Muon Correlation condition module" (4.4.12.2.7).	2016/01/15
3.9	Removed "Double objects requirements condition with spatial correlation", because not used anymore in the future, replaced by Correlation conditions (see sections ?? and ??.	2016/01/08
3.8	Minor changes in text and updated Figure 11.	2016/01/08
3.7	Changed colour in Figure 12 and updated text for correlation conditions (see section 4.4.12.	2016/01/07
3.6	Updated Figures 11 and 10 and text 4.4.12.2.1.	2015/12/21
3.5	Inserted drawing of VHDL structure of cuts for correlation conditions (see Figure 13).	2015/11/18
3.4	Updated muon η ranges (Table 9) and inserted correlation conditions (see section 4.4.12). Created scheme for conversion of calorimeter η and φ to muon scale for calo-muon-correlation conditions (see Figure 14).	2015/11/17
3.3	Added Text in sections (4.4.6.1.3) and (4.4.6.1.4).	2015/10/08
3.2	Updated Text in section "Final Desicion Logic" (5).	2015/10/06
3.1	Updated Figure 15 and Tables 27, ?? and ??. Remaned section "Calorimeter conditions module - version 2" to "Calorimeter conditions module - version 3" (see ??), section "Muon conditions module" to "Muon conditions module - version 2" and section "Muon comparators module" to "Muon comparators module - version 2" (see ??)	2015/10/02
3.0	Updated text and tables of η ranges for Calorimeter objects (see 4.4.2).	2015/09/22
2.9	Renewed Figures in GTL and FDL (see Figure 9, 10 and 11) and FDL(see Figure 15 and 16). Added register bits description of FDL Register map (see section 5.4.1.1).	2015/09/16
2.8	Updated text, tables and listings of section "VHDL-Templates for VHDL-Producer" (see ??).	2015/09/15
2.7	Corrected calculation of muon η step width (see 4.4.4).	2015/09/10
2.6	Edited text in Tables ??, 17 and ??.	2015/08/28
2.5	Updated definition of η ranges for Calorimeter objects and Muon objects (4.4.2 and 4.4.4).	2015/08/20

Doc Rev	Description of Change	Revision Date
2.4	Added section Calo Muon Correlation condition (4.4.12.2.6).	2015/08/19
2.3	Added section "Register map" (5.4.1.1) for μFDL.	2015/06/26
2.2	Updated figures (9, 10 and 11) for GTL and edited section "Correlation conditions" (see 4.4.12).	2015/05/08
2.1	Added tables for calorimeter isolation-bits and for muon quality- and isolation-bits definition (8, 11 and 13). Edited section glossary (7) and acronyms.	2015/05/07
2.0	Added text for "Energy sum conditions" (4.4.7) and updated chapters for "Calorimeter conditions" for version 2. Inserted isolation bits for electron/γ and tau objects (4.4.2).	2015/05/06
1.9	Minor changes "Demux Lane Data" (see 3.2) and "Muon data" (see 4.4.4).	2014/11/06
1.8	Edited Section "Energy sum quantities conditions" (see 4.4.7).	2014/10/08
1.7	Added sections "Configuration of optical connections" (3.1), "Demux Lane Data" (3.2) and "Lane Mapping Process" (3.3) to framework. Removed tables of optical interfaces from gtl and referenced to tables in framework.	2014/10/07
1.6	Minor changes in "Calorimeter conditions" (**??**) and "Muon conditions" (**??**)	2014/07/01
1.5	Updated with minor changes in "Muon conditions" (**??**)	2014/06/17
1.4	Fixed bug in Figure 12	2014/04/30
1.3	Updated section "Muon conditions" (**??**)	2014/04/22
1.2	Removed section "Muon charge module" and added new section "Muon charge correlation module" (see 4.4.11). Edited text in section and subsections "Muon conditions definition" (see **??**)	2014/04/15
1.1	Changed Figure 12 and minor changes in text for anti-clockwise behaviour in φ	2014/04/04
1.0	Added definition for "calorimeter conditions over bx", see section **??**.	2014/03/12
0.9	Changed text of condition description in subsections **??** and **??**	2014/02/12
0.8	Updated calorimeter data structure in 4.2.1	2013/12/03
0.7	Updated muon data structure in 4.2.2	2013/12/02
0.6	Moved decription of VHDL templates for TME to **??**	2013/11/18
0.5	Subsection 4.2 added to section 4	2013/11/11
0.4	GTL and FDL firmware implemented for new data structure (GTL firmware version v1.0.0 [fix part of GTL], FDL firmware version v1.0.0)	2013/11/06

Doc Rev	Description of Change	Revision Date
0.3	New framework implementation based on new object types definition. Additionally, the ROP is implemented based on production requirements	2013/10/13
0.2	First framework implementation + ROP	2012/07/01
0.1	Document created	2012/02/22

Contents

List of Figures

List of Tables

1 Global Trigger System overview

Entire document is "under construction"!

The Global Trigger System is based on uTCA technology and 10Gbps optical links. A set of 6 MP7 boards with FPGAs of the powerful Xilinx Virtex-7 family is available. The Global Trigger firmware is implemented on these FPGAs. Every FPGA contains a part of the VHDL representation of a L1 Menu, the partitioning is done by VHDL Producer tool. The trigger decision of every MP7 board is collected on an AMC502 board to generate the "final OR" signal which triggers the readout of the detector.

2 Firmware overview

The figure 1 shows the architecture of μGT payload. It consists of framework and the algorithm logic which it consists of the following modules:

1. Global Trigger Logic Data Mapping
2. μGTL
3. μFDL

The output mux (part of framework) collects data for read-out record which are send via MP7 read-out to AMC13.

The IPBus system allows the control of hardware via a ‘virtual bus’, using a standard IP-over-gigabit-Ethernet network connection.

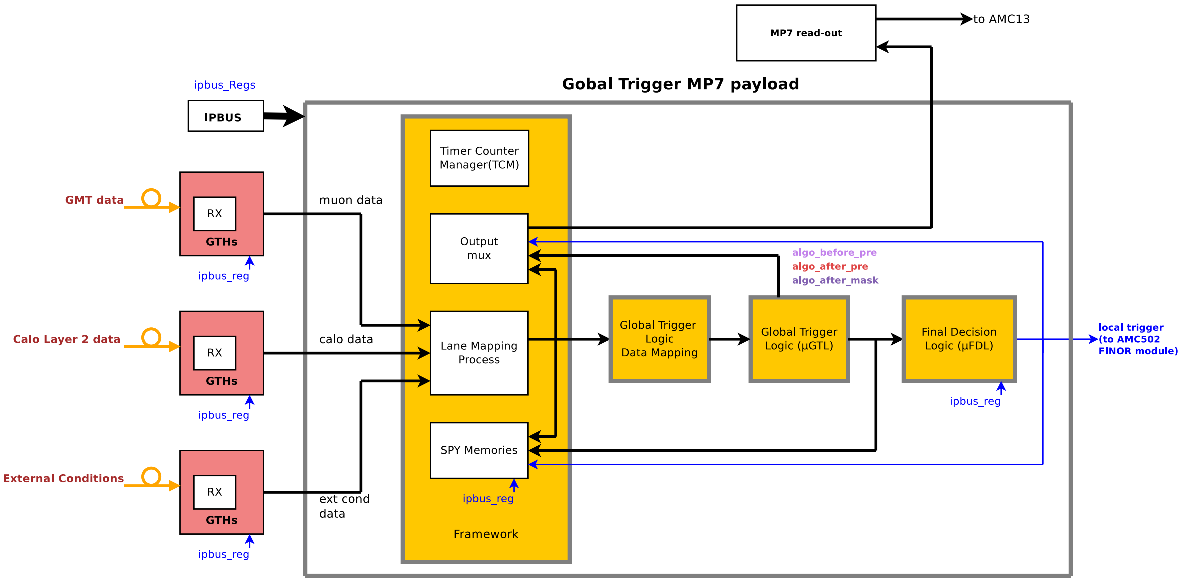


Figure 1: μGT payload

2.1 Firmware version

This firmware description is valid for version 1.15.3 of Global Trigger firmware, containing the following module versions:

- Framework: 1.2.3
- Global Trigger Logic: 1.15.1
- Final Decision Logic: 1.3.6

2.2 Directory structure of Global Trigger firmware

INSERT TEXT !!!

2.2.1 Package: lhc_data_pkg

The VHDL record `lhc_data_t` (shown in Listing 1) is used as a container for all object streams processed by the system. It is declared in the VHDL package `lhc_data_pkg`. For debugging and simulation purposes a second package (`lhc_data_debug_util_pkg`) is created which contains functions to convert the `lhc_data_t` to a hexadecimal string representation and vice versa. The testbench of the design uses this functions to load the contents of the SIM memory from a file.

Listing 1: lhc_data_t record specification

```
type lhc_data_t is record
        muon : muon_array_t;
        eg : eg_array_t;
        tau : tau_array_t;
        jet : jet_array_t;
        ett : std_logic_vector(ETT_DATA_WIDTH-1 downto 0);
        ht : std_logic_vector(HT_DATA_WIDTH-1 downto 0);
        etm : std_logic_vector(ETM_DATA_WIDTH-1 downto 0);
        htm : std_logic_vector(HTM_DATA_WIDTH-1 downto 0);
        etmhf : std_logic_vector(ETMHF_DATA_WIDTH-1 downto 0);
        htmhf : std_logic_vector(HTMHF_DATA_WIDTH-1 downto 0);
        link_11_fr_0_data : std_logic_vector(LINK_11_FR_0_WIDTH-1 downto
            0);
        link_11_fr_1_data : std_logic_vector(LINK_11_FR_1_WIDTH-1 downto
            0);
        link_11_fr_2_data : std_logic_vector(LINK_11_FR_2_WIDTH-1 downto
            0);
        link_11_fr_3_data : std_logic_vector(LINK_11_FR_3_WIDTH-1 downto
            0);
        link_11_fr_4_data : std_logic_vector(LINK_11_FR_4_WIDTH-1 downto
            0);
        link_11_fr_5_data : std_logic_vector(LINK_11_FR_5_WIDTH-1 downto
            0);
        external_conditions : std_logic_vector(
            EXTERNAL_CONDITIONS_DATA_WIDTH-1 downto 0);
end record;
```

3 Framework

Remark:
with frame v1.2.3 "Delay Manager" (dm.vhd) and "Data Source Multiplexer" (dsmux.vhd) are removed because these features were never used in production system, only for tests. Simmem data not useable anymore, because of removed dsmux. The reason of removing is to get more available resources.

Figure 2 shows the basic components the framework together with Readout-Process.

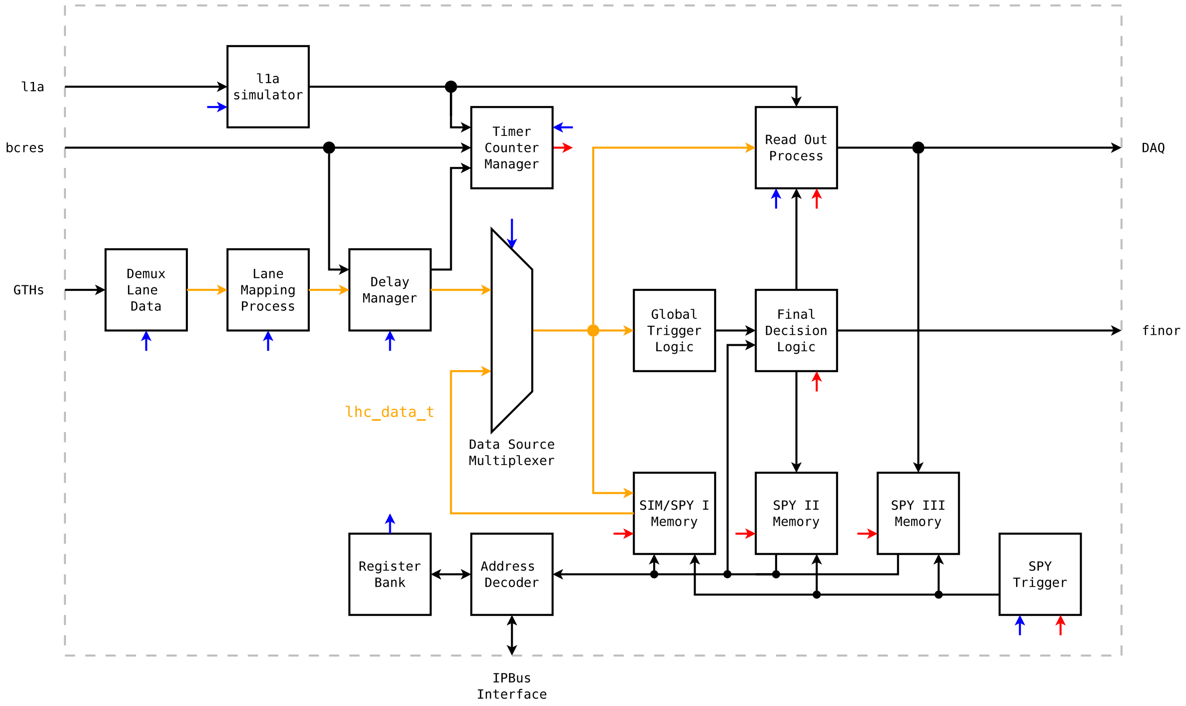


Figure 2: System architecture overview

The central data type of the framework is shown in Listing 1 (see Section 2.2.1 for details). In the current configuration it comprises 2304 bits (288 Bytes). Data from the GTH interfaces is demultiplexed (from 240 MHz clock domain to LHC clock domain, see Demux Lane Data 3.2) and mapped to this data type in the LMP (Lane Mapping Process). It is also used as input and output type for the SIM/SPY I memory. The DM (Delay Manager) takes the output of the LMP and applies software configurable delays to the the different object streams (e.g. muon data, jet, tau etc.) in the `lhc_data_t` to produce a consistent output (also regarding to the bcres signal). The software configurable multiplexer DSMUX (Data Source Multiplexer) is used to select which data stream is used as input for the processing elements (trigger logic). The output of the DSMUX is routed to the GTL (Global Trigger Logic) and ROP (Read Out Process) and can optionally be stored in the SPY I memory.

3.1 Configuration of optical connections

The configuration of the optical connections to Calo-Layer2 is (currently) done as described in Table 2, where frame means the 32 bits data (240 MHz) within a LHC clock period.

3.2 Demux Lane Data

Data from GTH interfaces is in the 240 MHz clock domain. The demultiplexing to the LHC clock domain (about 40 MHz) is done in demux_lane_data.vhd, which is instantiated in frame.vhd as often as lanes are used (currently 16 lanes are used).

3.3 Lane Mapping Process

In the Lane Mapping Process module data from the lanes are mapped to objects structure defined in lhc_data_pkg.vhd.

3.3.1 Implementation

Currently lane mapping is "fixed" in lmp.vhd module, see Table 3

3.4 Delay Manager

Remark:
with frame v1.2.3 "Delay Manager" (dm.vhd) and "Data Source Multiplexer" (dsmux.vhd) is removed because these features were never used in production system, only for tests. The reason of removing is to get more available resources.

The Delay Manager is responsible for creating a delayed version of the `lhc_data` and the bcres signal on its input. For this purpose it uses an internal memory to record the history of the input signals.

3.4.1 Implementation

The DM is basically a reimplementation of the concept of the last design. The reimplementation was necessary because the new framework version uses the register bank for software registers and the old DM was not flexible enough to handle the `lhc_data_t` introduced in the new framework.

The DM instantiates one `delay_element` for every object type defined in the `lhc_data_t` (e.g. muon, eg, etc.). The `delay_element` uses RAM blocks to implement the delay line. However, for the delays 0 and 1 this memory can not be used (write latency) and must be bypassed (Figure 3).

Table 2: Configuration of optical connections

	frame					
link	**0**	**1**	**2**	**3**	**4**	**5**
0	reserved	reserved	muon obj. 0 [0..31]	muon obj. 0 [32..63]	muon obj. 1 [0..31]	muon obj. 1 [32..63]
1	reserved	reserved	muon obj. 2 [0..31]	muon obj. 2 [32..63]	muon obj. 3 [0..31]	muon obj. 3 [32..63]
2	reserved	reserved	muon obj. 4 [0..31]	muon obj. 4 [32..63]	muon obj. 5 [0..31]	muon obj. 5 [32..63]
3	reserved	reserved	muon obj. 6 [0..31]	muon obj. 6 [32..63]	muon obj. 7 [0..31]	muon obj. 7 [32..63]
4	electron/γ obj. 0	electron/γ obj. 1	electron/γ obj. 2	electron/γ obj. 3	electron/γ obj. 4	electron/γ obj. 5
5	electron/γ obj. 6	electron/γ obj. 7	electron/γ obj. 8	electron/γ obj. 9	electron/γ obj. 10	electron/γ obj. 11
6	jet obj. 0	jet obj. 1	jet obj. 2	jet obj. 3	jet obj. 4	jet obj. 5
7	jet obj. 6	jet obj. 7	jet obj. 8	jet obj. 9	jet obj. 10	jet obj. 11
8	tau obj. 0	tau obj. 1	tau obj. 2	tau obj. 3	tau obj. 4	tau obj. 5
9	tau obj. 6	tau obj. 7	tau obj. 8	tau obj. 9	tau obj. 10	tau obj. 11
10	ET ETTEM MBT0HFP	HT TOWER-COUNT MBT0HFM	ET_{miss} ASYMET MBT1HFP	HT_{miss} ASYMHT MBT1HFM	ET_{miss}^{HF} ASYM-ETHF CENT[3:0]	HT_{miss}^{HF} ASYM-HTHF CENT[7:4]
11	free	free	free	free	free	free
12	external-conditions [0..31]	external-conditions [32..63]	reserved	reserved	reserved	reserved
13	external-conditions [64..95]	external-conditions [96..127]	reserved	reserved	reserved	reserved
14	external-conditions [128..159]	external-conditions [160..191]	reserved	reserved	reserved	reserved
15	external-conditions [192..223]	external-conditions [224..255]	reserved	reserved	reserved	reserved

Table 3: Current lane mapping

lane	**objects**
0	muon objects 0..1
1	muon objects 2..3
2	muon objects 4..5
3	muon objects 6..7
4	electron/γ objects 0..5
5	electron/γ objects 6..11
6	jet object 0..5
7	jet object 6..11
8	tau object 0..5
9	tau object 6..11
10	energy sum quantities (incl. minimum bias trigger bits and towercounts)
11	n/a (currently not used)
12	external-conditions [0..63]
13	external-conditions [64..127]
14	external-conditions [128..191]
15	external-conditions [192..255]

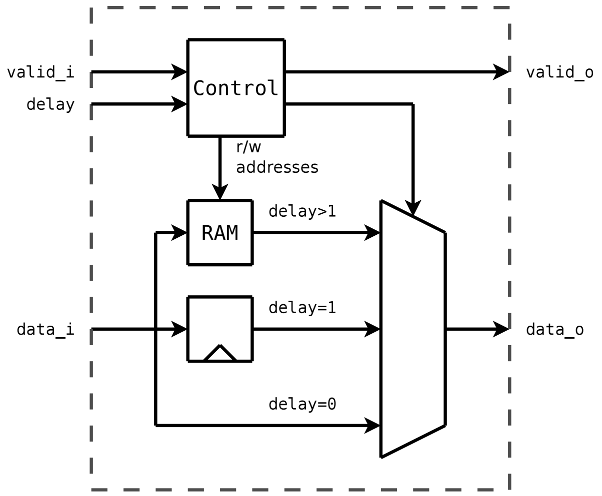


Figure 3: Delay Manager: delay_element

The Implementation of the DM is very generic because it makes extensive use of the constants provided by the `lhc_data_pkg`. The *lhc_data_i* input signal is converted into a `std_logic_vector`. The constants `LHC_DATA_SLV_START_INDICES` and `LHC_DATA_SLV_OBJECT_WIDTH` provide the start index of each object in this vector and their width respectively. The number of objects is given by the `LHC_DATA_OBJECT_COUNT` constant. This information is used by a for-generate statement to instantiate the required `delay_element` components.

For the *bcres_o* and the *bcres_FDL_o* signals two additional `delay_element` components are instantiated.

If only the data width or the array size of an object in the `lhc_data_t` is changed the DM does not need any modification. If, however, a new object is added a new delay register must be added, as described in the register bank section.

The registers for all delays have the same layout (see Register 3.1). The names for the individual (per object) delay registers are given in Table 4. For the software addresses of these registers refer to the `xml/gt_amc514_dm.xml` file.

Register 3.1: Delay Manager Registers

reserved	delay
31 12	11 0
0	0

delay The delay in lhc clock cycles (40 MHz) used for the specific object data.

Table 4: delay manager registers

object description	register name
bcres for TCM	bcres_tcm_delay
bcres for FDL	bcres_fdl_delay
muon data	muons_delay
e/g	eg_delay
tau	tau_delay
jet	jet_delay
ett	ett_delay
ht	ht_delay
etm	etm_delay
htm	htm_delay
external conditions	ex_con_delay

3.5 SIM and SPY Memory

Remark:
with frame v1.2.3 Simmem data not useable anymore, because of removed "Data Source Multiplexer". The reason of removing "Data Source Multiplexer" is to get more available resources.

Under construction!!!

Figure 4 shows the SIM/SPY memory subsystem of the framework. It is used to calibrate the system, i.e. to set the correct delays in the Delay Manager, to record results of the GTL/FDL and output packages of the ROP and to provide simulation data for the system. All source files for the memory subsystem are located in `src/mem` directory.

3.5.1 Implementation

The memory subsystem consists of four main parts, which will be discussed in more detail in the following sections

- SPY Trigger
- SIM/SPY Memory
- SPY Memory II
- SPY Memory III

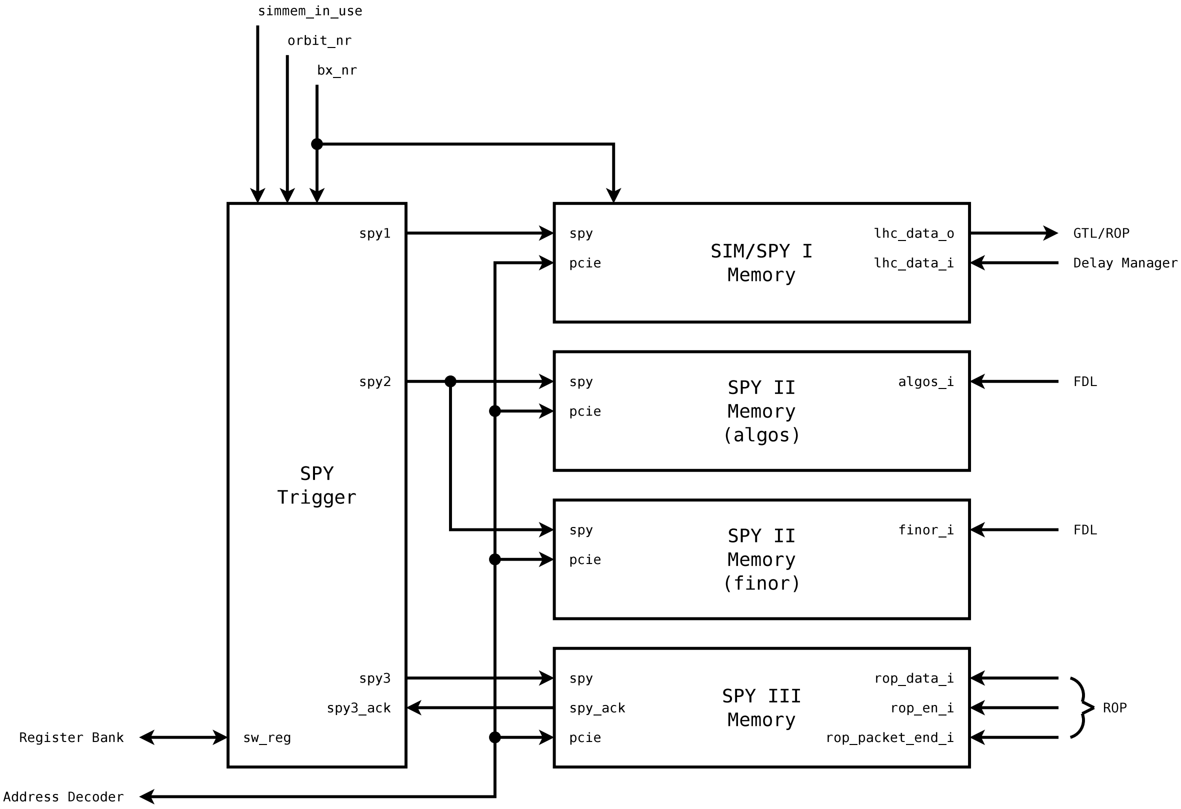


Figure 4: Memory subsystem

3.5.1.1 SPY Trigger

The SPY trigger controls the SPY memories and decides when data is recorded. It can be configured and controlled using software registers 3.2 and 3.3 provided by the register bank.

When the SPY trigger receives a spy12 command (next or once) over the software register interface it asserts the *spy*1 and *spy*2 signals for the appropriate orbit. This means that the *spy* signals go high with the bunch crossing counter reaching the value zero and stay high until it reaches zero again (overflow). Note that when the SIM memory is being used (indicated by the *simem_in_use_i* input provided by the DSMUX component) the *spy*1 output will not be asserted.

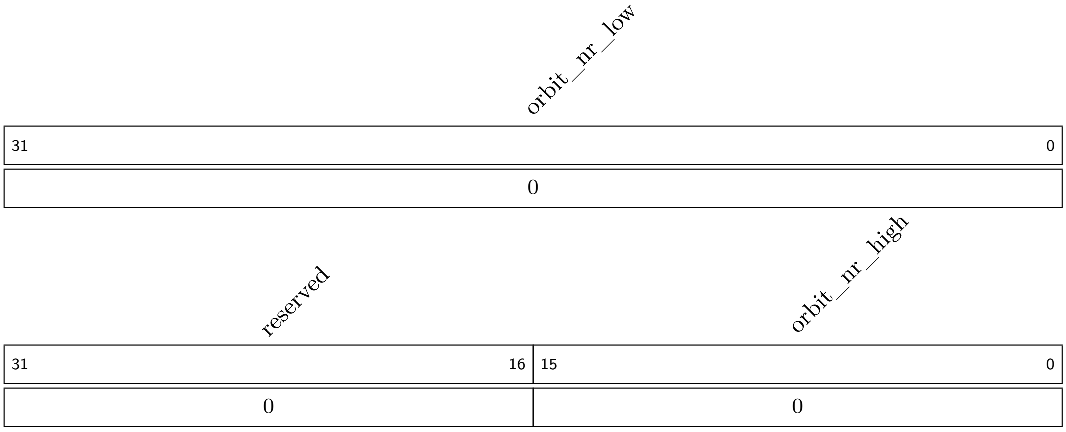
When a spy3 command is received the SPY trigger asserts the *spy*3 signal and waits until the *spy*3*_ack* signal is asserted.

3.5.1.2 SIM/SPY memory

This component combines the SIM memory and the SPY memory I. This optimization is possible because these two memories are never used at the same time. There are basically two use cases for this memory.

- SIM memory: Data is read form the memory and provided to GTL and ROP to test these components.

Register 3.2: SPY Trigger Orbit Number Registers



orbit_nr_low	The 32 low bits of the 48 bit orbit number, used for the spy once trigger.
orbit_nr_high	The 16 high bits of the 48 bit orbit number, used for the spy once trigger.

- SPY memory: External data is received by the GTX and stored in the memory to check the alignment of the data.

It is very important to guarantee that the spy input signal is not asserted, as long as the memory is used as SIM memory. Note that this functionality is implemented in the SPY trigger component.

The SIM/SPY memory converts the *lhc_data_i* input signal to a `std_logic_vector` using the converter function provided by the `lhc_data_pkg`. This vector is then divided into chunks of 32 bits (the PCIe data width). For each of these chunks a 32 bit true-dual-port memory (2 read ports, 2 write ports, 2 clock domains) is instanciated. Thus, every memory has a read/write port in both clock domain, the 125MHz PCIe clock domain and the 40MHz LHC clock domain, which can be used simultaneously. The PCIe data-in signal (*sw_i.data*) is connected to PCIe-clock domain write port of the memories. A memory select signal is generated form the LSBs of the software address (*sw_i.addr*). The memory select signal also controls the multiplexer on the output of the memories to generate the *sw_o.data* signal.

Depending on whether the SIM/SPY memory is used to provide simulation data or to store/spy data the address on the LHC-clock domain port of the internal memories is adjusted. If data is recorded (SPY) the bunch crossing counter is used as memory address directly. When the memory is read the read latency (two clock cycles) must be taken into account. This is achieved by subtracting 2 form the bunch crossing number before using it as address. To generate the *lhc_data_o* signal the LHC-clock domain data out ports of the internal memories are concatenated and converted back to the `lhc_data_t`.

If the `lhc_data_t` is changed (e.g. new objects added) no modifications in the SIM/SPY memory are required. The SIM/SPY memory only depends on the (auto-generated) functions

Register 3.3: SPY Trigger Configuration Register

spy12__bsy	spy3__bsy	spy12__rdy	spy3__rdy	spy12__err	reserved	clr__spy12__err	clr__spy3__rdy	clr__spy12__rdy	spy3	spy12__next	spy12__once	
31	30	29	28	27	26 6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	Reset

spy12__once	Triggers the recording of the selected orbit to SPY memories I and II, when written with 1.
spy12__next	Triggers the recording of the next whole orbit to SPY memories I and II, when written with 1.
spy3	Triggers the recording of the next package that will be sent by the ROP to SPY memory III, when written with 1.
clr__spy12__rdy	Clears the ready flag of the SPY trigger for SPY memories I and II, when written with 1.
clr__spy3__rdy	Clears the ready flag of the SPY trigger for SPY memory III, when written with 1.
clr__spy12__err	Clears the error flag, when written with 1.
spy12__bsy	Indicates that the SPY trigger for SPY memories I and II is busy.
spy3__bsy	Indicates that the SPY trigger for SPY memory III is busy.
spy12__rdy	Indicates that one orbit has been recorded in SPY memories I and II and that the SPY trigger is ready for new commands.
spy3__rdy	Indicates that packet has been recorded in SPY memory III and that the SPY trigger is ready for new commands.
spy12__err	Indicates an error condition (Set only when the selected orbit number for the spy once trigger lies in the past and can therefore not be recorded).

used to convert a `lhc_data_t` signal to `std_logic_vector` and vice versa (see Section 2.2.1 for details).

In the current implementation the size every object in the `lhc_data_t` is a multiple of 32 bit. This is also expected by the SIM/SPY memory. If objects with 16 bit sizes are added the SIM/SPY memory must be modified to support this situation (e.g. zero pad the `lhc_data_t`). Furthermore take into account that the PCIe memory bus is 32 bits wide. So 16 bit objects should be added to the end of the `lhc_data_t` (as last entry) to keep software memory access simple.

3.5.1.3 SPY memory II

The SPY memory II is divided into two subcomponents, to store the *algos* and $finor$ outputs of the FDL. Both memory can only be read over the SW interface. A write access has no effect. The algos memory uses the same architecture as the SIM memory. The finor memory uses a true-dual-port memory with asymmetric ports. This memory can be written with a data width of one bit and read with a data width of 32 bit.

3.5.1.4 SPY memory III

The SPY memory III stores the output of the ROP, which is sent to the DAQ. The input data width is configurable to bus widths of 16, 32 or 64 bits. Depending on the input data width the memory uses different architectures.

- 16 Bit
  A true-dual-port memory with asymmetric ports (16 and 32 bits) is used.
- 32 Bit
  A true-dual-port memory with 32 Bit data width is used.
- 64 Bit
  Two true-dual-port memories with 32 Bit data width are used.

3.5.2 Interface Specification

Listing 2: SPY trigger interface specification

```
entity spytrig is
        port
        (
                lhc_clk    : in  std_logic;
                lhc_rst    : in  std_logic;
                orbit_nr   : in  orbit_nr_t;
                bx_nr      : in bx_nr_t;
                sw_reg_i   : in  sw_reg_spytrigger_in_t;
                sw_reg_o   : out sw_reg_spytrigger_out_t;

                spy1_o     : out std_logic;
                spy2_o     : out std_logic;
                spy3_o     : out std_logic;
                spy3_ack_i : in  std_logic;

                simmem_in_use_i : in std_logic
        );
end;
```

3.6 TCM

Under construction!!!

The Timer Counter Manager (TCM) provides different counters, listed in table 5.

3.6.1 Counter Overview

Table 5: counters of the timer counter manager

Counter	range	increase condition	reset condition	Comments
bx_nr	$0to3563$	rising_edge(lhc_clk)	overflow	
event_nr	$0to2^{32}-1$	l1a=1 and rising_edge(lhc_clk)	BGOS: event counter reset	
trigger_nr	$0to2^{48}-1$	l1a=1 and rising_edge(lhc_clk)	BGOS: start run	
orbit_nr	$0to2^{48}-1$	overflow of bx_nr	BGOS: orbit counter reset	
luminosity_seg_nr	$0to2^{32}-1$	rising_edge(orbit_nr(18))	BGOS: orbit counter reset	
start_lumisection	$0to1$	luminosity_seg_nr increases	after 25ns	’1’ for 25ns
bx_nr_d_fdl	$0to3563$	rising_edge(lhc_clk)	overflow	

3.6.2 Bunch Crossing Number and counters derived from it

All counters except for event_nr and the trigger_nr (which are trivial because they are increased with l1a) are dependent on the bunch crossing counter bx_nr as stated in table 5. The bx_nr is zero at startup, then waits for the the first bcres_d (bunch crossing reset delayed) and starts counting as depicted in figure 5. It’s maximal value is 3563 (0xdeb), then it automatically overflows and starts at zero again (see figure 6). Exactly when bx_nr = 0, bcres_d has to be asserted. Otherwise the counter is out of synchronization. If this happens, the software register err_det is set and the counter waits for the next bcres_d to synchronize again. Note that the value of the counter is invalid until it has synchronized again.

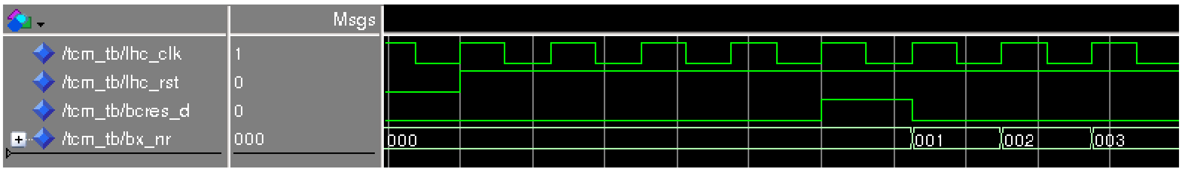


Figure 5: start of the bunch crossing number with the first bcres_d

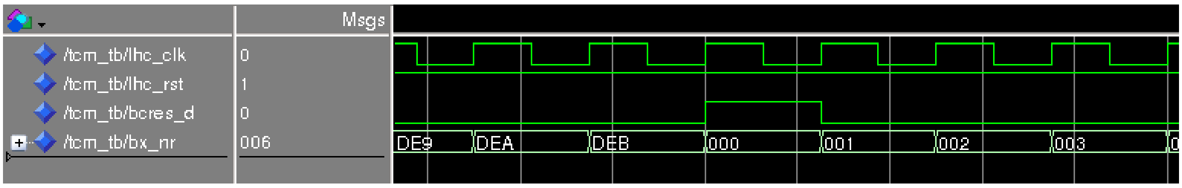


Figure 6: normal operation of the bunch crossing number

3.6.3 Special counter: bx_nr_d_fdl

The bx_nr_d_fdl is derived from bcres_d_fdl in the same manner as bx_nr is derived from bcres_d. bx_nr_d_fdl will automatically resync if the logic described in subsection 3.6.5 detects a synchronization error for bx_nr.

3.6.4 Counters derived from l1a

The counters event_nr and trigger_nr are increased with l1a, i.e. they are increased twice if l1a is high for 2 clock cycles, etc. They differ only in their value range and the condition that resets the counters, see table 5.

3.6.5 Errors

As stated above, bcres_d has to be asserted exactly when bx_nr = 0, otherwise the counter is out of sync. Then the software register err_det is set as depicted in figure 7. err_det can be reset via the software event register err_det_reset_event as depicted in figure 8. Furthermore err_det is set if bgos = Resync-0x1 and the counter value is not 3563.

The TCM implements two additional counters (bx_nr_chk and bx_nr_max) for debugging purposes. These counters are not visible by any other module but readable via software. bx_nr_chk is a 32bit Counter that increases with every LHC clock cycle and resets with bcres_d. bx_r_max holds the highest value bx_nr_chk ever reached (should be 3563 if the link is aligned).

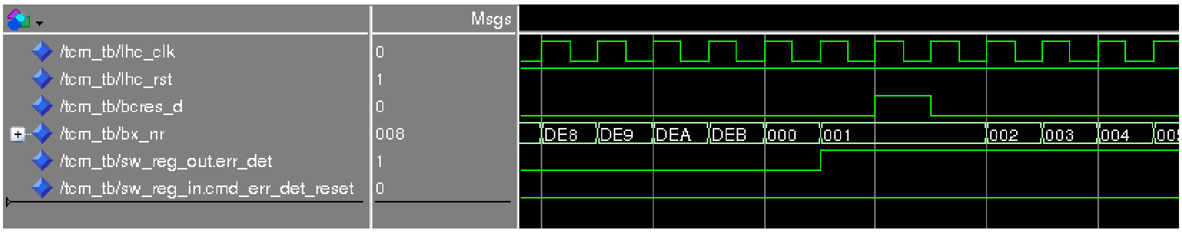


Figure 7: set of the software register err_det when bc_res_d is not asserted correctly

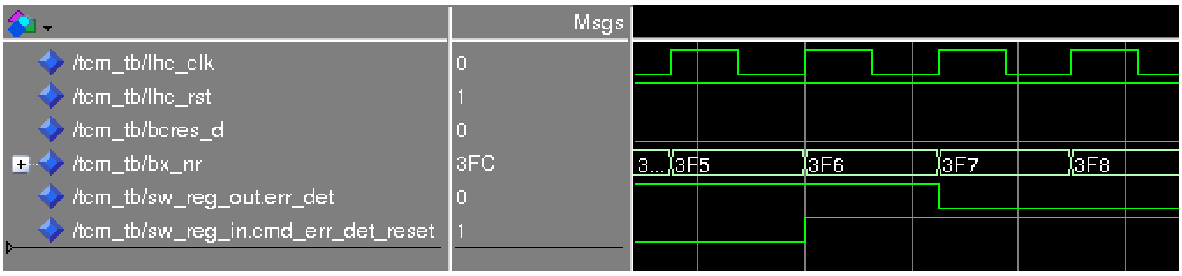


Figure 8: reset of the software register err_det when err_det_reset_event toggles

3.6.6 SW-Registers

All counters except for the start_lumisection described in table 5 can be read by software via the following sw registers:

Register 3.4: TCM Bunch Crossing Number Register

reserved	bx_nr
31 … 12	11 … 0
0	0

Register 3.5: TCM Event Number Register

event_nr
31 … 0
0

Register 3.6: TCM Trigger Number Registers

trigger__nr__l
31 0
0

reserved	trigger__nr__h
31 16	15 0
0	0

trigger__nr__l	The 32 low bits of the 48 bit trigger number.
trigger__nr__h	The 16 high bits of the 48 bit trigger number.

Register 3.7: TCM Orbit Number Registers

orbit__nr__l
31 0
0

reserved	orbit__nr__h
31 16	15 0
0	0

orbit__nr__l	The 32 low bits of the 48 bit orbit number.
orbit__nr__h	The 16 high bits of the 48 bit orbit number.

Register 3.8: TCM Luminosity Segment Number Register

luminosity__seg__nr
31 0
0

Register 3.9: TCM Bunch Crossing Number FDL Register

reserved	bx_nr_d_fdl
31 12	11 0
0	0

Register 3.10: TCM Bunch Crossing Number Check Register

bx_nr_chk
31 0
0

Register 3.11: TCM Bunch Crossing Number Max Register

bx_nr_max
31 0
0

Some additional control register can be used to check and reset err_det, disable the check of bcres_d and bcres_d_fdl (bx_nr and bx_nr_d_fdl automatically reset when they overflow if cmd_ign_bcres is set, bcres_d is ignored) and simulate the bgos signal. To do this, a value of the orbit signal has to be written to sw-register bgos. The value of the input signal bgos is replaced by the value of the sw-register for exactly one clock cycle, when "1" is written to the event register bgos_event.

Register 3.12: TCM cmd_ign_bcres

reserved	cmd_ign_bcres	
31 1	0	
0	0	Reset

cmd_ign_bcres bcres is ignored (not checked) when this is set.

Register 3.13: TCM ERR_DET

reserved	err_det	
31 1	0	
0	0	Reset

err_det Set when out of synchronization.

Register 3.14: TCM ERR_DET_RESET_EVENT

reserved	err_det_reset_event	
31 1	0	
0	0	Reset

err_det_reset_event Event register: resets err_det.

Register 3.15: TCM BGOS

reserved	bgos
34 4	3 0
0	0

Reset

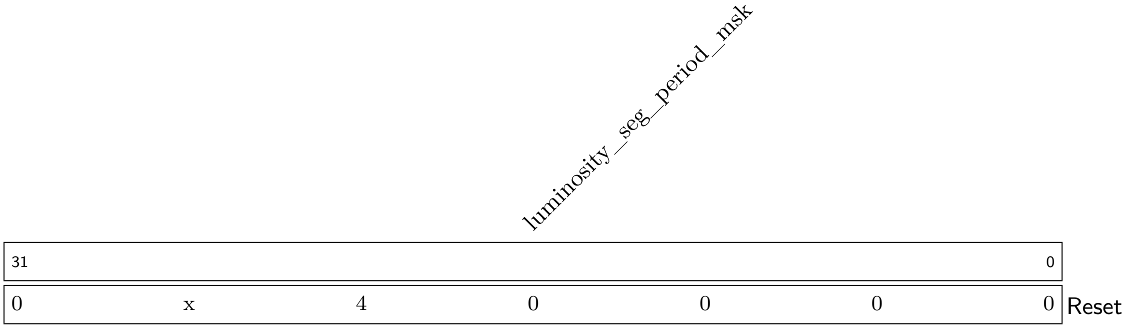
bgos For simulation of the bgos signal.

Register 3.16: TCM BGOS_EVENT

reserved	bgos_event	
31 1	0	
0	0	Reset

bgos_event Event register: replaces the input signal bgos by the sw-register bgos for exactly one clock cycle.

Register 3.17: TCM LUMINOSITY__SEG__PERIOD__MSK



luminosity__seg__period__msk luminosity__seg__nr is increased when the orbit__nr mod lum__seg__period__mask = 0.

3.6.7 Hardware Test

There are various python scripts located in the software/GtControl/branches/fpga-design-2013/python/GtControl directory for testing the tcm module. Please refer to the output of the scripts for information how the tests are performed in detail. See table 6.

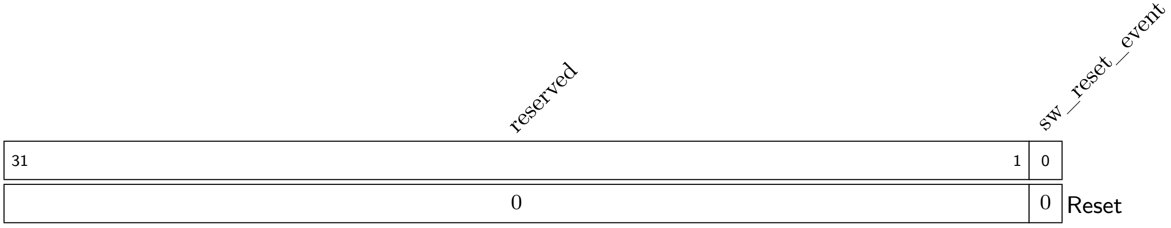
Table 6: scripts for testing the tcm

script	purpose
`tcm_counter_values.py`	outputs the values of all counters defined above
`tcm_produce_err_det`	produces an err__det by manipulating bgos
`tcm_err_det_reset`	resets the err__det software register
`tcm_trigger_test`	tests trigger__nr and event__nr by generating l1a signals using l1asim
`tcm_lum_seg_nr_test`	checks the period of two successive increases of the luminositiy__seg__nr

3.7 Software Reset

The software reset module (sw__reset) provides the possiblity for a software reset via the software event register sw__reset__event.

Register 3.18: SOFTWARE RESET REGISTER



sw__reset__event Event register: Generates a reset signal for exactly one clock cycle.

4 Global Trigger Logic

Remark:
This description is for version 1.16.0 of Global Trigger Logic.

The Global Trigger Logic (μGTL) firmware contains conditions and algorithms for trigger decision.

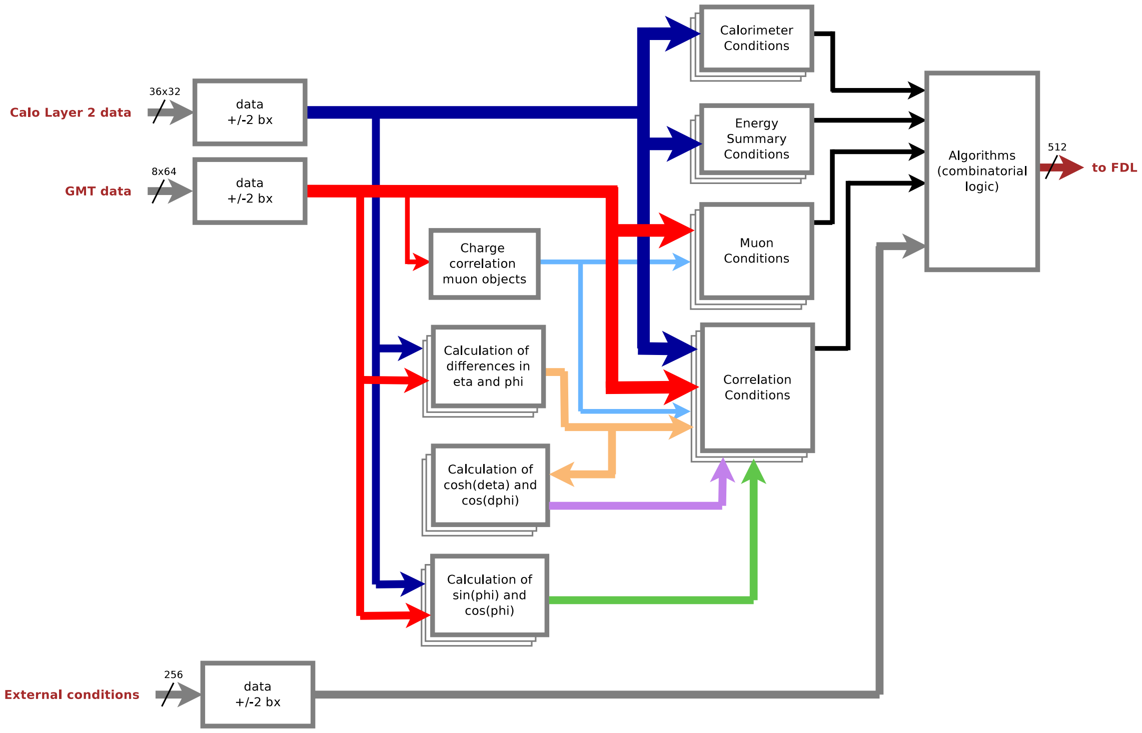


Figure 9: μGTL firmware

4.1 μGTL Interface

Inputs:

- Calo-Layer2 data
  - Electron/γ objects
  - Jet objects
  - Tau objects
  - Energy summary information:
    - Total Et (ET)
    - total Et from ECAL only (ETTEM)
    - total calibrated Et in jets (HT)

        - missing Et (ET_{miss})
        - missing Et including HF (ET_{miss}^{HF})
        - missing Ht objects (HT_{miss})
        - missing Ht including HF (HT_{miss}^{HF})
        - "Asymmetry" information (ASYMET, ASYMHT, ASYMETHF, ASYMHTHF)
    - Minimum bias HF bits (included in energy summary information data structure)
    - Towercount bits (number of firing HCAL towers, included in energy summary information data structure)
    - "Centrality" bits
- Global Muon Trigger data
- External conditions

Outputs:

- Algorithms

4.2 Definition of optical interfaces

Remark:
All definitions for scales in the following chapters are from a CMS Detector Note: "Scales for inputs to μGT" (see actual version in https://raw.githubusercontent.com/cms-l1-globaltrigger/mp7_ugt_legacy/master/doc/scales_inputs_2_ugt/pdf/scales_inputs_2_ugt.pdf).

4.2.1 Calo-Layer2 optical interfaces

The configuration of optical connections from Calo-Layer2 to μGT is shown in Table 2.

The data structure of an electron/γ object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 25	24 17	16 9	8 0
`qual/spare`	`iso`	φ	η	E_{T}

The data structure of a jet object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 19	18 11	10 0
`iso/qu/sp`	φ	η	E_{T}

The data structure of a tau object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 25	24 17	16 9	8 0
qual/spare	*iso*	φ	η	E_{T}

The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits]:

31 28	27 24	23 12	11 0
MBT0HFP	*spare*	E_{T} *[ETTEM]*	E_{T} *[ET]*

The data structure of "total calibrated Et in jets" (HT) quantity [including "towercount" and "minimum bias HF- threshold 0" bits]:

31 28	27 25	24 12	11 0
MBT0HFM	*spare*	*TOWERCOUNT*	E_{T}

The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits]:

31 28	27 20	19 12	11 0
MBT1HFP	*ASYMET*	φ	E_{T}

The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits]:

31 28	27 20	19 12	11 0
MBT1HFM	*ASYMHT*	φ	E_{T}

The data structure of "missing Et including HF" (ET_{miss}^{HF}) quantity [including "Asymmetry" ASYMETHF and "Centrality" bits (3:0)]:

31 28	27 20	19 12	11 0
[CENT3:0]	*ASYMETHF*	φ	E_{T}

The data structure of "missing Ht including HF" (HT_{miss}^{HF}) quantity [including "Asymmetry" ASYMHTHF and "Centrality" bits (7:4)]:

31 28	27 20	19 12	11 0
CENT[7:4]	*ASYMHTHF*	φ	E_{T}

4.2.2 Global Muon Trigger optical interfaces

The data structure of a muon object (64 bits - bit 34 = charge sign, bit 35 = charge valid, bit 61 is a spare bit, bit 63..62 = impact parameter):

63 62	61	60 53	52 43	42 36	35 34	33 32
imp para	r	unconst.p_T	φ (out)	index bits	ch	iso

31 23	22 19	18 10	9 0
η (extrapol.)	qual	p_T	φ (extrapol.)

4.3 Implementation in firmware

The firmware of μGTL consists of two main parts:

- A top-of-hierarchy file (`gtl_module.vhd`), which contains the pipeline for ±2bx data, the instantiations of calculators for differences in η and φ, the instantiations of conditions, the instantiations of charge correlation logic of muons and the Algorithms logic for 512 Algorithms, as well as a package file (`gtl_pkg.vhd`) for declarations. Actually 6 AMC board are used to contain 512 Algorithms. Therefore the 512 Algorithms are **partitioned by VHDL Producer**. The VHDL Producer for every Trigger Menu creates VHDL snippets files (`algo_index.vhd`, `gtl_module_instances.vhd`, `gtl_module_signals.vhd`, `ugt_constants.vhd`), these snippets are inserted into templates for `gtl_module.vhd` (`gtl_module_tpl.vhd`) and `gtl_pkg.vhd` (`gtl_pkg_tpl.vhd`) during simulation and synthesis.
- A set of VHDL-files exists for all the modules instantiated in top-of-hierarchy and the modules in the hierarchy. These files, called the "fixed part", are not influenced by VHDL Producer.

The latency of μGTL is fixed to 5 bunch-crossings, 2 bunch-crossings for the pipeline of ±2bx data (for data with +2bx and +1bx), 2 bunch-crossings for conditions (fixed), also for the conditions requested in the future, 1 bunch-crossing for the logic of Algorithms (See Figure 11).

4.3.1 Top-of-hierarchy module

The top-of-hierarchy module (`gtl_module.vhd`) contains

- the pipeline for ±2bx data
- the instantiations of charge correlation logic of muons (generated by VHDL Producer)
- the instantiations of calculators for differences in η and φ (generated by VHDL Producer)
- the instantiations of conditions (generated by VHDL Producer)
- a boolean logic for Algorithms (generated by VHDL Producer)

Listing 3 contains the entity-declaration of the top-of-hierarchy file (`gtl_module.vhd`).

Listing 3: Entity declaration of `gtl_module.vhd`

```
entity gtl_module is
    port(
        lhc_clk : in std_logic;
        eg_data : in calo_objects_array(0 to NR_EG_OBJECTS-1);
        jet_data : in calo_objects_array(0 to NR_JET_OBJECTS-1);
        tau_data : in calo_objects_array(0 to NR_TAU_OBJECTS-1);
```

```
        ett_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        ht_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        etm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        htm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
--
    *************************************************************************************

-- HB 2016-04-18: updates for "min bias trigger" objects (quantities) for Low-
    pileup-run May 2016
        mbt1hfp_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt1hfm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt0hfp_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt0hfm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
-- HB 2016-06-07: inserted new esums quantities (ETTEM and ETMHF).
        ettem_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        etmhf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
-- HB 2016-09-16: inserted HTMHF and TOWERCNT
        htmhf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        towercount_data : in std_logic_vector(MAX_TOWERCOUNT_BITS-1 downto 0);
-- HB 2018-08-06: inserted signals for "Asymmetry" and "Centrality" (included in
    esums data structure).
        asymet_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymht_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymethf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymhthf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        centrality_data : in std_logic_vector(NR_CENTRALITY_BITS-1 downto 0);
--
    *************************************************************************************

        muon_data : in muon_objects_array(0 to NR_MUON_OBJECTS-1);
        external_conditions : in std_logic_vector(NR_EXTERNAL_CONDITIONS-1 downto
             0);
        algo_o : out std_logic_vector(NR_ALGOS-1 downto 0));
end gtl_module;
```

4.3.2 Package module

All the declarations for arrays ('type'), parameters ('constant') and look-up-tables ('constant') used in modules are available in `gtl_pkg.vhd` package-file.

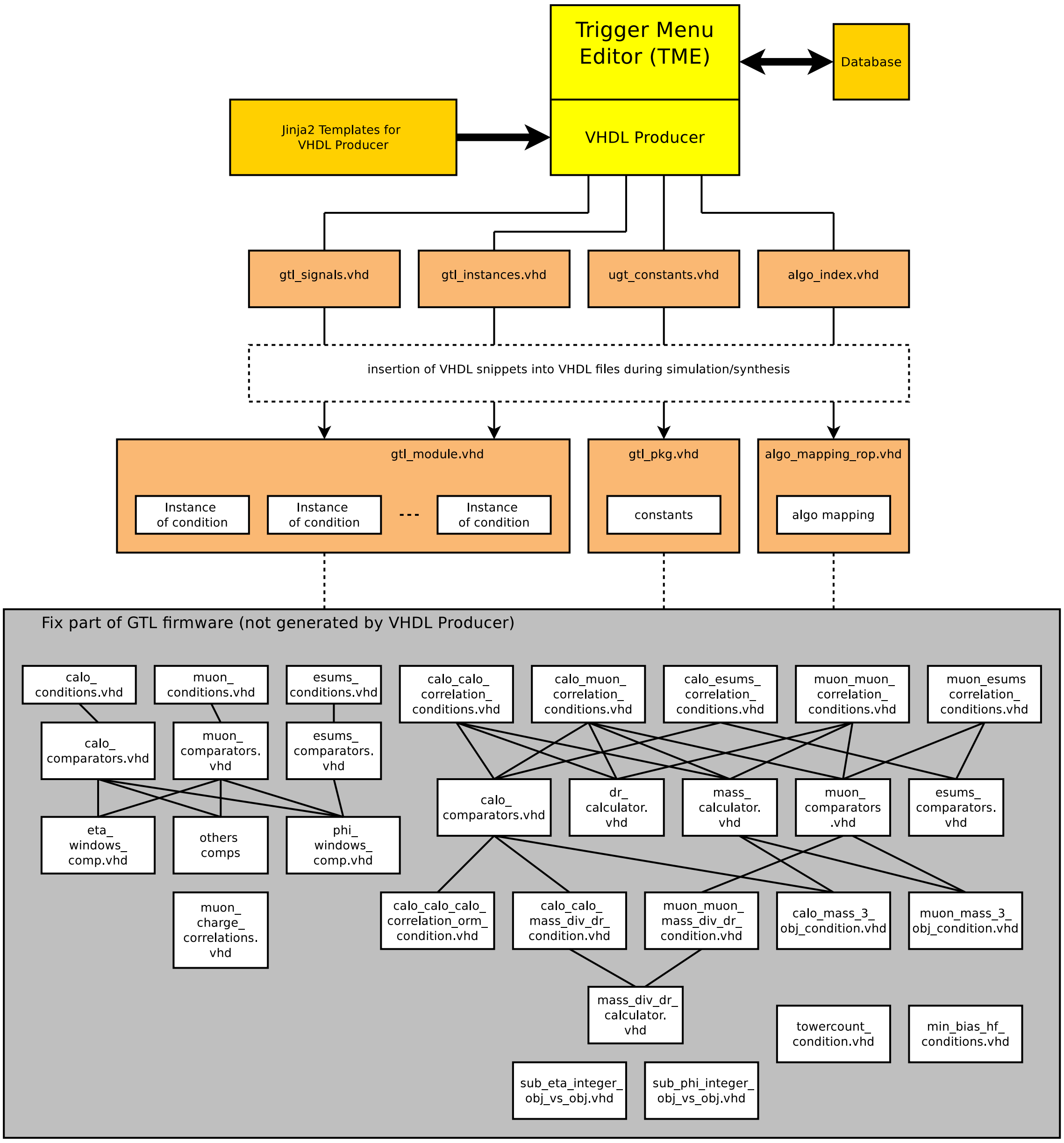


Figure 10: VHDL file generation by VHDL Producer

4.4 μGTL structure

4.4.1 Data ±2bx

The μGTL input data flow through a register pipeline of four stages. With those data it is possible to have conditions with objects from different bunch-crossings (within ±2 bunch-crossings), e.g. for Correlation conditions.
See Figure 11 for a scheme of μGTL pipeline structure. The data "data_p_1bx" and "data_p_2bx" occur 1 respectively 2 bunch-crossings after data for a certain bunch-crossing, therefore we got 2 bunch-crossings of latency from those data. The data "data_m_1bx" and "data_m_2bx" have no influence on latency, because coming before data for a certain bunch-crossing.

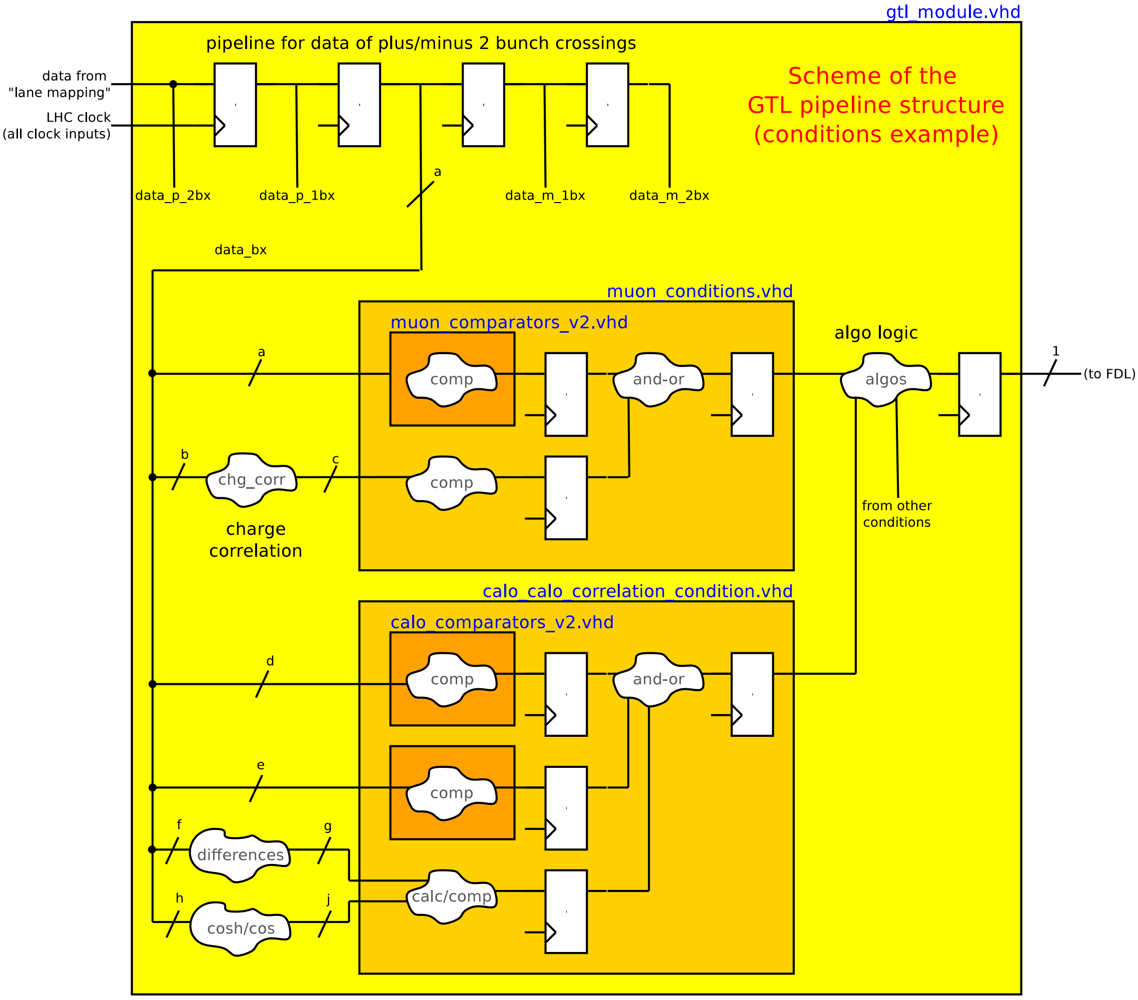


Figure 11: Scheme of μGTL pipeline structure

4.4.2 Definitions of Calorimeter data

The calorimeter trigger processing identifies **electron/γ, jet and tau** objects and **energy sum quantities**.

electron/γ:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters for E_T and for position and quality information - encoded in 32 bits:

- 9 bits p_T, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 138 bins (HW index = 0xBC..0x44)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\hat{=}$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 2 bits isolation
- 5 bits spare

The data structure of an electron/γ object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 25	24 17	16 9	8 0
`spare`	`iso`	φ	η	p_T

jet:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters: E_T, for position and quality information - encoded in 32 bits:

- 11 bits p_T, range = 0..1023 GeV (HW index = 0..0x7FF), step = 0.5, the highest bin will mark an overflow (HW index 0x7FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 230 bins (HW index = 0x8E..0x72)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\hat{=}$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 5 bitsspare

The data structure of a jet object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 19	18 11	10 0
`spare`	φ	η	p_{T}

tau:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters for E_{T} and for position and quality information - encoded in 32 bits:

- 9 bits p_{T}, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 138 bins (HW index = 0xBC..0x44)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\widehat{=}2.5°$), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 2 bits isolation
- 5 bits spare

The data structure of a tau object (bits 27..31 are not defined yet, reserved for quality, ...):

31 27	26 25	24 17	16 9	8 0
`spare`	`iso`	φ	η	E_{T}

The representation of the 8 bits (called "hardware index [HW index]") in η is expected as Two's Complement notation as shown below.

HW index	η **range**	η **bin**
0x72	114∗0.087/2 to 115∗0.087/2	114
...	...	...
0x01	0.087/2 to 2∗0.087/2	1
0x00	0 to 0.087/2	0
0xFF	0 to -0.087/2	-1
0xFE	-0.087/2 to -2∗0.087/2	-2
...	...	...
0x8E	-114∗0.087/2 to -115∗0.087/2	-115

The representation of the 8 bits in φ is expected as shown in Table 7.

The representation of the 2 bits for isolation (e/γ and tau) is expected as shown in Table 8.

Table 7: φ scale of calorimeter objects

HW index	φ range	φ range [degrees]	φ bin
0x00	0 to $2\pi/144$	0 to 2.5	0
0x01	$2\pi/144$ to $2{*}2\pi/144$	2.5 to 5.0	1
...	...	...	...
0x8F	$143{*}2\pi/144$ to 2π	357.5 to 360	143

Table 8: Definition of e/γ and tau isolation bits

bits [26..25]	definition
00	not isolated
01	isolated
10	TBD
11	TBD

4.4.3 Definitions of Energy sum quantities data

energy sum quantities:
Consists of following quantities (naming convention see 7):

- **ET**
- **HT**
- ET_{miss}
- HT_{miss}
- **ETTEM**
- $\mathbf{ET}_{miss}^{HF}$
- $\mathbf{HT}_{miss}^{HF}$
- **ASYMET**
- **ASYMHT**
- **ASYMETHF**
- **ASYMHTHF**
- **CENT0**
- ..
- **CENT7**

Calo-Layer2 sends 6 frames (each 32 bits) with Energy sum quantities containing the following information:

- E_{T}, 12 bits, range = 0..2047 GeV (HW index = 0..0xFFF), step = 0.5, the highest bin will mark an overflow (HW index 0xFFF): meaning has to be defined
- azimuth angle (φ) position, 8 bits, range = 2π, step $\approx 2\pi/144$ ($\hat{=}$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- "Towercount", 13 bits, range = 0..8191
- "Minimum bias", 4 bits, range = 0..15
- "Asymmetry", 8 bits, range = 0..255 (used 0..100)
- "Centrality", 8 bits, used as signals

Frame0: The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits]:

31 28	27 24	23 12	11 0
MBT0HFP	*spare*	E_{T} *[ETTEM]*	E_{T} *[ET]*

Frame1: The data structure of "total calibrated Et in jets" (HT) quantity [including "towercount" and "minimum bias HF- threshold 0" bits]:

31 28	27 25	24 12	11 0
MBT0HFM	*spare*	*TOWERCOUNT*	E_{T}

Frame2: The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits]:

31 28	27 20	19 12	11 0
MBT1HFP	*ASYMET*	φ	E_{T}

Frame3: The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits]:

31 28	27 20	19 12	11 0
MBT1HFM	*ASYMHT*	φ	E_{T}

Frame4: The data structure of "missing Et including HF" (ET^{HF}_{miss}) quantity [including "Asymmetry" ASYMETHF and "Centrality" bits (3:0)]:

31 28	27 20	19 12	11 0
CENT[3:0]	*ASYMETHF*	φ	E_{T}

Frame5: The data structure of "missing Ht including HF" (HT^{HF}_{miss}) quantity [including "Asymmetry" ASYMHTHF and "Centrality" bits (7:4)]:

31 28	27 20	19 12	11 0
CENT[7:4]	*ASYMHTHF*	φ	E_{T}

4.4.4 Definitions of Muon data

Eight Muon objects are provided by Global Muon Trigger. One Muon object has a 64 bits data structure with parameters for p_{T}, for unconstrained p_{T}, for impact parameter, for position, charge, quality and isolation information:

- 10 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/576$ ($\widehat{=}$0.625°), 576 bins (HW index = 0..0x23F), HW index starting at 0° (anti-clockwise)
- 9 bits p_{T}, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 4 bits quality, 16 types for quality (meaning not defined yet!)
- 9 (8+1 sign) bits pseudo-rapidity (η) position, range = -2.45 to 2.45, step = 0.087/8, linear scale, 452 bins (-225..225, HW index = 0x11F..0x0E1)
- 2 bits isolation, 4 types for isolation (meaning not defined yet!)
- 1 bit charge sign, charge sign = '0' means "positive" charge, charge sign = '1' means "negative" charge
- 1 bit charge valid (='1' means "valid")
- 7 index bits
- 10 bits azimuth angle (φ) position, raw data
- 8 bits unconstrained p_{T}, range = 0..255 GeV (HW index = 0..0xFF), step = 1.0, the highest bin will mark an overflow (HW index 0xFF)
- 1 spare bit
- 2 bits impact parameter

The data structure of a muon object (64 bits - bit 34 = charge sign, bit 35 = charge valid, bit 61 is a spare bit, bit 63..62 = impact parameter):

63 62	61	60 53	52 43	42 36	35 34	33 32
`imp para`	*r*	`unconst.`p_T	φ *(out)*	`index bits`	`ch`	`iso`

31 23	22 19	18 10	9 0
η *(extrapol.)*	`qual`	p_T	φ *(extrapol.)*

The representation of the 9 bits (called "hardware index [HW index]") in η is expected as Two's Complement notation as shown in Table 9.
The central value of the bin 0 (-0.010875/2 to +0.010875/2) = 0.0, the left edge of the bins will range from $-255 \times 0.010875 - 0.010875/2 = -2.7785625$ to $+255 \times 0.010875 - 0.010875/2 = 2.7676875$. The central value of the bins will range between ± 2.773125 . The physical η range of the muon detectors is about ± 2.45, so that not all possible η bins will be used.

Table 9: η scale of muon objects

HW index	η range	η bin
0x0E1	224.5∗0.087/8 to 225.5∗0.087/8	225
0x0E0	223.5∗0.087/8 to 224.5∗0.087/8	224
...	...	...
0x001	0.5∗0.087/8 to 1.5∗0.087/8	1
0x000	0.5∗-0.087/8 to 0.5∗0.087/8	0
0x1FF	0.5∗-0.087/8 to 1.5∗-0.087/8	-1
0x1FE	1.5∗-0.087/8 to -2.5∗0.087/8	-2
...	...	...
0x11F	-224.5∗0.087/8 to -225.5∗0.087/8	-225

The representation of the 10 bits in φ is expected as shown in Table 10.

Table 10: φ scale of muon objects

HW index	φ range	φ range [degrees]	φ bin
0x000	0 to $2\pi/576$	0 to 0.625	0
0x001	$2\pi/576$ to $2*2\pi/576$	0.625 to 1.250	1
...	...	...	...
0x23F	$575*2\pi/576$ to 2π	359.375 to 360	575

The representation of the 4 bits for quality is expected as shown in Table 11.

The representation of the 2 bits for isolation is expected as shown in Table 13.

Table 11: Definition of muon quality bits

bits [22..19]	definition
0000	quality "level 0"
0001	quality "level 1"
0010	quality "level 2"
0011	quality "level 3"
0100	quality "level 4"
0101	quality "level 5"
0110	quality "level 6"
0111	quality "level 7"
1000	quality "level 8"
1001	quality "level 9"
1010	quality "level 10"
1011	quality "level 11"
1100	quality "level 12"
1101	quality "level 13"
1110	quality "level 14"
1111	quality "level 15"

Table 12: Definition of muon isolation bits

bits [33..32]	definition
00	not isolated
01	isolated
10	TBD
11	TBD

The representation of the 2 bits for impact parameter is expected as shown in Table 13.

4.4.5 Calculation of differences in η and φ

Some condition types namely correlation conditions uses differences in η and φ to make the decision. Therefore these differences are calculated out of these conditions, because the differences can be used several times in different condition types. The differences in η and φ are calculated in bins. These differences in bins are converted to numbers (by LUTs), which represents values of differences (multiples of units in η and φ). Differences in φ are provided by module `sub_phi_integer_obj_vs_obj.vhd`, which instantiates the module `sub_unsigned_phi.vhd` as many times as the numbers of both objects determine.
In the module `sub_unsigned_phi.vhd` a calculation of a difference of two objects is done, both objects must have the same resolution, namely the higher one. The result is the absolute

Table 13: Definition of muon impact parameter bits

bits [63..62]	definition
00	TBD
01	TBD
10	TBD
11	TBD

value of the difference. There are two differences in φ, one "clockwise" and one "anti-clockwise". For the final result the smaller difference is taken.
Differences in η are provided by module `sub_eta_integer_obj_vs_obj.vhd`, which instantiates the module `sub_unsigned_eta.vhd` as many times as the numbers of both objects determine.
In the module `sub_unsigned_eta.vhd` a calculation of a difference of two objects is done with a signed subtraction, because of the Two's Complement notation of η values. The result is the absolute value of the difference.

4.4.6 Combination conditions

4.4.6.1 Combination conditions definition

A condition consists of input data and a set of requirements, which contain the requirements to be complied. The requirements are called "object cuts".

The requirement list contains:
thresholds for p_{T}, ranges for η and φ, LUTs for isolation, LUTs for quality, requsted charges, thresholds for unconstrained p_{T}, a LUTs for impact parameter. The condition is complied, if every comparison between object parameters and requirements is valid for the following object cuts (only for requested cuts):

For Calorimeter input data:

- p_{T} greater-equal (or equal) threshold
- η in range
- φ in range
- iso LUT

For Muon input data:

- p_{T} greater-equal (or equal) threshold
- η in range
- φ in range

- iso LUT
- requested charge
- quality LUT
- unconstrained p_{T} greater-equal (or equal) threshold
- impact parameter LUT

There are different types of conditions implemented, depending of how many objects have to comply the requirements.

- "Quad objects requirements condition": this condition type consists of requirements for 4 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 4 different objects, each of which fulfills at least one of the requirements.
- "Triple objects requirements condition": this condition type consists of requirements for 3 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 3 different objects, each of which fulfills at least one of the requirements.
- "Double objects requirements condition": this condition type consists of requirements for 2 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 2 different objects, each of which fulfills at least one of the requirements.[1]
- "Single object requirement condition": this condition type consists of one requirement for one trigger object of a given object type. To fulfill this condition, there must exist at least one object which fulfills the requirement.

The values of the requirements are given by VHDL Producer for every Trigger Menu.
The input data objects have to be of same type and same bunch-crossing.

With "Double objects requirements condition" a correlation cut of "two-body pt" can be required (calorimeter and muon objects).
Additionally charge correlation cuts with "Double objects requirements condition", "Triple objects requirements condition" and "Quad objects requirements condition" of muon objects can be required.

4.4.6.1.1 Combination conditions module

TBD

---

[1] "Double objects requirements condition with spatial correlation" not used anymore, replaced by Correlation conditions

Listing 4: Entity declaration of `comb_conditions.vhd`

```
entity comb_conditions is
     generic(

        slice_1_low_obj1: natural := 0;
        slice_1_high_obj1: natural := NR_MU_OBJECTS-1;
        slice_2_low_obj1: natural := 0;
        slice_2_high_obj1: natural := NR_MU_OBJECTS-1;
        slice_3_low_obj1: natural := 0;
        slice_3_high_obj1: natural := NR_MU_OBJECTS-1;
        slice_4_low_obj1: natural := 0;
        slice_4_high_obj1: natural := NR_MU_OBJECTS-1;
        pt_ge_mode_obj1: boolean := true;
        pt_thresholds_obj1: common_templates_array := (others => (others => '0'))
            ;
        nr_eta_windows_obj1: common_templates_natural_array := (others => 0);
        eta_w1_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w1_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w2_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w2_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w3_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w3_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w4_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w4_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w5_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        eta_w5_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        nr_phi_windows_obj1: common_templates_natural_array := (others => 0);
        phi_w1_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        phi_w1_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        phi_w2_upper_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        phi_w2_lower_limits_obj1: common_templates_array := (others => (others =>
             '0'));
        iso_luts_obj1: common_templates_iso_array := (others => (others => '1'));
        requested_charges_obj1: common_templates_string_array := (others => "ign"
            );
        qual_luts_obj1: common_templates_quality_array := (others => (others =>
            '1'));
        upt_cuts_obj1: common_templates_boolean_array := (others => false);
        upt_upper_limits_obj1: common_templates_array := (others => (others =>
            '0'));
        upt_lower_limits_obj1: common_templates_array := (others => (others =>
            '0'));
        ip_luts_obj1: common_templates_ip_array := (others => (others => '1'));
```

```
        requested_charge_correlation: string(1 to 2) := "ig";

        slice_low_obj2: natural := 0;
        slice_high_obj2: natural := NR_TAU_OBJECTS-1;
        pt_ge_mode_obj2: boolean := true;
        pt_threshold_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0) := (
            others => '0');
        nr_eta_windows_obj2: natural := 0;
        eta_w1_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w1_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w2_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w2_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w3_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w3_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w4_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w4_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w5_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        eta_w5_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        nr_phi_windows_obj2: natural := 0;
        phi_w1_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        phi_w1_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        phi_w2_upper_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        phi_w2_lower_limit_obj2: std_logic_vector(MAX_TEMPLATES_BITS-1 downto 0)
            := (others => '0');
        iso_lut_obj2: std_logic_vector(2**MAX_ISO_BITS-1 downto 0) := (others =>
            '1');

        twobody_pt_cut: boolean := false;
        pt_width: positive := EG_PT_VECTOR_WIDTH;
        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0) := (others => '0');
        twobody_upt_cut: boolean := false;
        upt_width: positive := MU_UPT_VECTOR_WIDTH;
        upt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0) := (others => '0');
        sin_cos_width: positive := CALO_SIN_COS_VECTOR_WIDTH;
        pt_sq_sin_cos_precision : positive := MU_MU_SIN_COS_PRECISION;

        deta_orm_cut: boolean := false;
        deta_orm_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0) := (others => '0');
        deta_orm_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0) := (others => '0');
```

```
        dphi_orm_cut: boolean := false;
        dphi_orm_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0) := (others => '0');
        dphi_orm_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0) := (others => '0');

        dr_orm_cut: boolean := false;
        dr_orm_upper_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0) := (others => '0');
        dr_orm_lower_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0) := (others => '0');

        nr_obj1: natural := NR_EG_OBJECTS;
        type_obj1 : natural := EG_TYPE;
        nr_obj2: natural := NR_TAU_OBJECTS;
        type_obj2 : natural := TAU_TYPE;
        nr_templates: positive := COMMON_NR_TEMPLATES

    );
    port(
        lhc_clk: in std_logic;
        obj1_calo: in calo_objects_array(0 to nr_obj1-1) := (others => (others =>
             '0'));
        obj1_muon: in muon_objects_array(0 to NR_MU_OBJECTS-1) := (others => (
            others => '0'));
        obj2: in calo_objects_array(0 to nr_obj2-1) := (others => (others => '0')
            );
        ls_charcorr_double: in muon_charcorr_double_array := (others => (others
            => '0'));
        os_charcorr_double: in muon_charcorr_double_array := (others => (others
            => '0'));
        ls_charcorr_triple: in muon_charcorr_triple_array := (others => (others
            => (others => '0')));
        os_charcorr_triple: in muon_charcorr_triple_array := (others => (others
            => (others => '0')));
        ls_charcorr_quad: in muon_charcorr_quad_array := (others => (others => (
            others => (others => '0'))));
        os_charcorr_quad: in muon_charcorr_quad_array := (others => (others => (
            others => (others => '0'))));
        deta_orm: in deta_dphi_vector_array(0 to nr_obj1-1, 0 to nr_obj2-1) := (
            others => (others => (others => '0')));
        dphi_orm: in deta_dphi_vector_array(0 to nr_obj1-1, 0 to nr_obj2-1) := (
            others => (others => (others => '0')));
        pt : in diff_inputs_array(0 to nr_obj1-1) := (others => (others => '0'));
        cos_phi_integer : in sin_cos_integer_array(0 to nr_obj1-1) := (others =>
            0);
        sin_phi_integer : in sin_cos_integer_array(0 to nr_obj1-1) := (others =>
            0);
        condition_o: out std_logic
    );
end comb_conditions;
```

Table 14: Explanation of Listing 4

Item	Explanation
`slice_1_low_obj1`	low value of slice for first object.
`slice_1_high_obj1`	high value of slice for first object.
`slice_2_low_obj1`	low value of slice for second object.
`slice_2_high_obj1`	high value of slice for second object.
`slice_3_low_obj1`	low value of slice for third object.
`slice_3_high_obj1`	high value of slice for third object.
`slice_4_low_obj1`	low value of slice for fourth object.
`slice_4_high_obj1`	high value of slice for fourth object.
`pt_ge_mode_obj1`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_thresholds_obj1`	array of four threshold values for comparison in pt (four threshold, because of max. 4 requirements).
`nr_eta_windows_obj1`	array of four integer values for number of η cuts.
`eta_w1_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 1 for η.
`eta_w1_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 1 for η.
`eta_w2_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 2 for η.
`eta_w2_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 2 for η.
`eta_w3_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 3 for η.
`eta_w3_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 3 for η.
`eta_w4_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 4 for η.
`eta_w4_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 4 for η.
`eta_w5_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 5 for η.
`eta_w5_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 5 for η.
`nr_phi_windows_obj1`	array of four integer values for number of φ cuts.
`phi_w1_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 1 for φ.
`phi_w1_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 1 for φ.
`phi_w2_upper_limits_obj1`	array of four "upper limits" of "window"-comparator 2 for φ.
`phi_w2_lower_limits_obj1`	array of four "lower limits" of "window"-comparator 2 for φ.
`iso_luts_obj1`	array of four LUTs (4 bits) for isolation.
`requested_charges_obj1`	array of four strings for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
`qual_luts_obj1`	array of four LUTs (16 bits) for quality.
`upt_cuts_obj1`	array of four boolean for using unconstrained p_{T} cuts.
`upt_upper_limits_obj1`	array of four "upper limits" of unconstrained p_{T}.
`upt_lower_limits_obj1`	array of four "lower limits" of unconstrained p_{T}.
`ip_luts_obj1`	array of four LUTs (4 bits) for impact parameter.

Table 14: Explanation of Listing 4

Item	Explanation
`requested_charge_correlation`	string (2 characters) for requested charge correlation ("ls" means "like sign", "os" means "opposite sign" or "ig" means "ignore").
`pt_ge_mode_obj2`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_thresholds_obj2`	array of four threshold values for comparison in pt (four threshold, because of max. 4 requirements).
`nr_eta_windows_obj2`	array of four integer values for number of η cuts.
`eta_w1_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 1 for η.
`eta_w1_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 1 for η.
`eta_w2_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 2 for η.
`eta_w2_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 2 for η.
`eta_w3_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 3 for η.
`eta_w3_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 3 for η.
`eta_w4_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 4 for η.
`eta_w4_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 4 for η.
`eta_w5_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 5 for η.
`eta_w5_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 5 for η.
`nr_phi_windows_obj2`	array of four integer values for number of φ cuts.
`phi_w1_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 1 for φ.
`phi_w1_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 1 for φ.
`phi_w2_upper_limits_obj2`	array of four "upper limits" of "window"-comparator 2 for φ.
`phi_w2_lower_limits_obj2`	array of four "lower limits" of "window"-comparator 2 for φ.
`iso_luts_obj2`	array of four LUTs (4 bits) for isolation.
`twobody_pt_cut`	valid strings are 'true' and 'false' (type is boolean).
`pt_width`	vector length of pt value for two-body pt.
`pt_sq_threshold_vector`	hex value for threshold of two-body pt comparison (value for pt square).
`twobody_upt_cut`	valid strings are 'true' and 'false' (type is boolean).
`upt_width`	vector length of unconstrained pt value for two-body pt.
`upt_sq_threshold_vector`	hex value for threshold of two-body unconstrained pt comparison (value for pt square).
`sin_cos_width`	vector length of sine and cosine.
`pt_sq_sin_cos_precision`	precision of sine and cosine calculation in LUTs.
`deta_orm_cut`	selection of $\Delta\eta$ cut for overlap removal.
`deta_orm_upper_limit_vector`	"upper limit" of "window"-comparator for comparison of $\Delta\eta$ for overlap removal (hex value).
`deta_orm_lower_limit_vector`	"lower limit" of "window"-comparator for comparison of $\Delta\eta$ for overlap removal (hex value).

Table 14: Explanation of Listing 4

Item	Explanation
`dphi_orm_cut`	selection of $\Delta\varphi$ cut for overlap removal.
`dphi_orm_upper_limit_vector`	"upper limit" of "window"-comparator for comparison of $\Delta\varphi$ for overlap removal (hex value).
`dphi_orm_lower_limit_vector`	"lower limit" of "window"-comparator for comparison of $\Delta\varphi$ for overlap removal (hex value).
`dr_orm_cut`	selection of ΔR cut for overlap removal.
`dr_orm_upper_limit_vector`	"upper limit" of "window"-comparator for comparison of ΔR^2 for overlap removal (hex value).
`dr_orm_lower_limit_vector`	"lower limit" of "window"-comparator for comparison of ΔR^2 for overlap removal (hex value).
`nr_obj1`	number of objects of input obj1.
`type_obj1`	type of input obj1.
`nr_obj2`	number of objects of input obj2 (for overlap removal).
`type_obj2`	type of input obj2 (for overlap removal).
`nr_templates`	number of requirements, selector of condition-type. Valid values are 1, 2, 3 and 4.
`lhc_clk`	clock input (LHC clock).
`obj1_calo`	calo input data.
`obj1_muon`	muon input data.
`obj2`	calo input data for overlap removal.
`ls_charcorr_double`	array of "like sign" charge correlation for double condition.
`os_charcorr_double`	array of "opposite sign" charge correlation for double condition.
`ls_charcorr_triple`	array of "like sign" charge correlation for triple condition.
`os_charcorr_triple`	array of "opposite sign" charge correlation for triple condition.
`ls_charcorr_quad`	array of "like sign" charge correlation for quad condition.
`os_charcorr_quad`	array of "opposite sign" charge correlation for quad condition.
`deta_orm`	array of $\Delta\eta$ for overlap removal.
`dphi_orm`	array of $\Delta\varphi$ for overlap removal.
`pt`	pt value for two-body pt.
`cos_phi_integer`	interger value of cosine for two-body pt.
`sin_phi_integer`	interger value of sine for two-body pt.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

4.4.6.1.2 Calorimeter Overlap Remover conditions module

The Calorimeter Overlap Remover conditions consits of a Calorimeter condition (**??**) and a single condition for a different calo object type. One or more correlation cut(s) ($\Delta\eta$, $\Delta\varphi$ and ΔR - 4.4.12) for overlap removal is required between different calo object types. Overlap Remover conditions `calo_conditions_orm.vhd` are implemented only for calo object types.

4.4.6.1.3 Calorimeter comparators module

A comparator between the energy (p_{T}) and a threshold (pt_threshold) and a comparison in η with five "window"-comparators and φ with two "window"-comparators is done in this basic module. The values for p_{T} threshold, the 'mode-selection' for the p_{T} comparator and the "limits" of the "window"- comparators is given in the generic interface list of the module. Additionally the data-structure of input data (data_i in port interface list) is provided as a record in this list. The output signal of the module is in high state, if all comparisons are true.

The comparison in η is done with five "window"-comparators, so one gets max. five ranges for η. The η value (HW index) has a Two's Complement notation, the comparisons is done signed. Number of windows is given for η.

The comparison in φ is done with two "window"-comparators, so one gets two ranges for φ. The comparisons is done unsigned. Number of windows is given for φ.

There are two cases how the limits of one "window"-comparator could be set (see also Figure 12 and Listing 5):

- Upper limit is less than lower limit => φ range between the limits, including the φ bin with value = 0 (HW index).
- Upper limit is greater/equal than lower limit => φ range between the limits, not including the φ bin with value = 0 (HW index).

Listing 5: VHDL code of "window"-comparator in φ

```
phi_comp_w1 <= '1' when phi_w1_upper_limit < phi_w1_lower_limit and
                 (phi <= phi_w1_upper_limit or phi >= phi_w1_lower_limit) else
               '1' when phi_w1_upper_limit >= phi_w1_lower_limit and
                 (phi <= phi_w1_upper_limit and phi >= phi_w1_lower_limit)
                else '0';
```

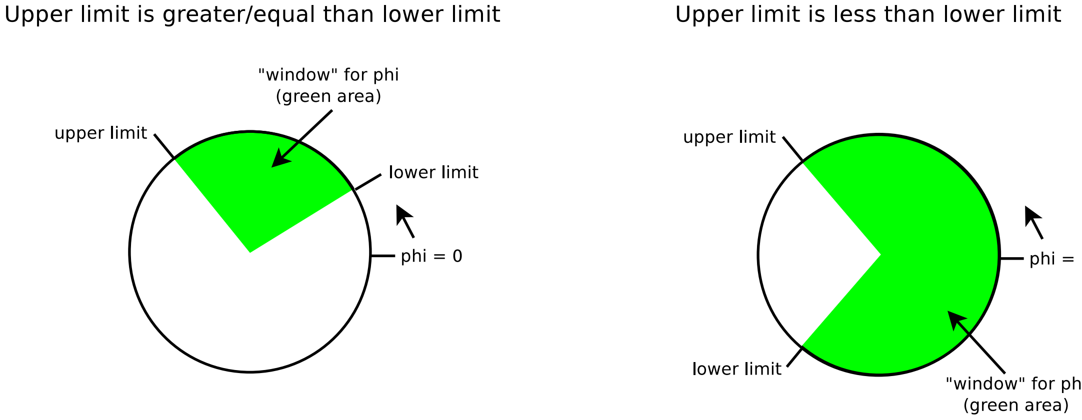


Figure 12: Setting the limits for "window"-comparators for φ

The values of η and φ have to be inside of only one of the required ranges ("or").

The comparison of isolation (for electron/γ, tau and muon) is done with a LUT (Table 15). [To ignore quality comparison, all bits in the LUT have to be '1'.]

Table 15: LUT contents for isolation comparison

LUT content (4 bits)	isolation (2 bits)	trigger
X"0"	xx	no trigger
X"1"	00	trigger on isolation bits = 00
X"2"	01	trigger on isolation bits = 01
X"3"	00 or 01	trigger on isolation bits = 00 or 01
X"4"	10	trigger on isolation bits = 10
X"5"	00 or 10	trigger on isolation bits = 00 or 10
X"6"	01 or 10	trigger on isolation bits = 01 or 10
X"7"	00 or 01 or 10	trigger on isolation bits = 00 or 01 or 10
X"8"	11	trigger on isolation bits = 11
X"9"	00 or 11	trigger on isolation bits = 00 or 11
X"A"	01 or 11	trigger on isolation bits = 01 or 11
X"B"	00 or 01 or 11	trigger on isolation bits = 00 or 01 or 11
X"C"	10 or 11	trigger on isolation bits = 10 or 11
X"D"	00 or 10 or 11	trigger on isolation bits = 00 or 10 or 11
X"E"	01 or 10 or 11	trigger on isolation bits = 01 or 10 or 11
X"F"	00 or 01 or 10 or 11	trigger on isolation bits = 00 or 01 or 10 or 11 (= "ignore" isolation)

4.4.6.1.4 Muon comparators module

This module contains the same comparators as in Calorimeter comparators module (4.4.6.1.3).

Additionally a comparator between unconstrained p_{T} and a threshold (upt_threshold), a comparison of impact parameter with LUT, a comparison of quality with LUT and a comparison of requested charge is done in this basic module.

The comparison of impact parameter is done with LUT (Table 16). [To ignore quality comparison, all bits in the LUT have to be '1'.]

The comparison of quality is done with LUT (Table 17). [To ignore quality comparison, all bits in the LUT have to be '1'.]

Charge valid and charge sign bits must be equal to the requested charge.

Table 16: LUT contents for impact parameter comparison

LUT content (4 bits)	impact parameter (2 bits)	trigger
X"0"	xx	no trigger
X"1"	00	trigger on impact parameter bits = 00
X"2"	01	trigger on impact parameter bits = 01
X"3"	00 or 01	trigger on impact parameter bits = 00 or 01
X"4"	10	trigger on impact parameter bits = 10
X"5"	00 or 10	trigger on impact parameter bits = 00 or 10
X"6"	01 or 10	trigger on impact parameter bits = 01 or 10
X"7"	00 or 01 or 10	trigger on impact parameter bits = 00 or 01 or 10
X"8"	11	trigger on impact parameter bits = 11
X"9"	00 or 11	trigger on impact parameter bits = 00 or 11
X"A"	01 or 11	trigger on impact parameter bits = 01 or 11
X"B"	00 or 01 or 11	trigger on impact parameter bits = 00 or 01 or 11
X"C"	10 or 11	trigger on impact parameter bits = 10 or 11
X"D"	00 or 10 or 11	trigger on impact parameter bits = 00 or 10 or 11
X"E"	01 or 10 or 11	trigger on impact parameter bits = 01 or 10 or 11
X"F"	00 or 01 or 10 or 11	trigger on impact parameter bits = 00 or 01 or 10 or 11 (= "ignore" impact parameter)

Table 17: LUT contents for quality comparison of muon objects

LUT content (16 bits)	quality bits (4 bits)	trigger
X"0000"	xxxx	no trigger
X"0001"	0000	trigger on quality "level 0"
X"0002"	0001	trigger on quality "level 1"
X"0003"	0001 or 0000	trigger on quality "level 1" or "level 0"
X"0004"	0010	trigger on quality "level 2"
...	...	...
X"8000"	1111	trigger on quality "level 15"
X"C000"	1111 or 1110	trigger on quality "level 15" or "level 14"
...	...	...
X"FFFF"	xx	trigger on all quality "levels" (= "ignore")

4.4.7 Energy sum quantities conditions

4.4.7.1 Energy sum quantities conditions module (including Asymmetry conditions)

For the entity-declaration of `esums_conditions.vhd`, see Listing 6.

Listing 6: Entity declaration of `esums_conditions.vhd`

```
entity esums_conditions is
    generic     (
        et_ge_mode : boolean;
        obj_type : natural := ETT_TYPE; -- ett=0, ht=1, etm=2, htm=3
        et_threshold: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0);
        phi_full_range : boolean;
        phi_w1_upper_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w1_lower_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w2_ignore : boolean;
        phi_w2_upper_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w2_lower_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
   );
    port(
        clk : in std_logic;
        data_i : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        condition_o : out std_logic
    );
end esums_conditions;
```

Table 18: Explanation of Listing 6

Item	Explanation
`et_ge_mode`	'mode-selection' for the E_T comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`obj_type`	valid strings are 'ETT_TYPE', 'HTT_TYPE', 'ETM_TYPE', 'HTM_TYPE' and 'ETMHF_TYPE'.
`et_threshold`	threshold value for comparison in E_T. The size of the std_logic_vector depends on the number of E_T bits.
`phi_full_range`	boolean to set full range of φ.
`phi_w1_upper_limits`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limits`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limits`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limits`	"lower limit" of "window"-comparator 2 for φ.
`clk`	clock input (LHC clock).
`data_i`	input data, structure defined in `obj_type`.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

A comparator between E_T and a threshold (et_threshold) and, depending on object type, a comparison in φ with two "window"-comparators is done in this module. The value for

E_{T} threshold, the 'mode-selection' for the E_{T} comparator and the limits for the "window"-comparators are given in the generic interface list of the module. The selection whether a comparison in φ is part of the condition is done with the value of the generic parameter 'obj_type' ('ETM_TYPE', 'ETMHF_TYPE', 'HTM_TYPE' and 'HTMHF_TYPE' force a comparison). The comparison in φ is done in the same way as for calorimeter conditions (see 4.4.6.1.3). Additionally the data-structure of input data (data_i in port interface list) is provided as a record in this list. The output signal of the module is in high state, if all comparisons are true.
Data for Asymmetry trigger are received on 4 frames on bits 27..20 (8 bits). For every type a comparision with an 8-bit threshold (greater-equal [or equal]) is done. Asymmetry data are interpreted as counts.

4.4.7.2 Energy sum quantities conditions module - template for VHDL-Producer

A VHDL-template for VHDL-Producer of instantiating `esums_conditions.vhd` is given below (see Listing **??**).

4.4.8 Minimum bias trigger conditions

Data for Minimum bias trigger are received on the 4 MSBs of 4 frames used for Energy sum quantities (see 4.4.7).

- MBT0HFP: "minimum bias HF+ threshold 0" bits
- MBT0HFM: "minimum bias HF- threshold 0" bits
- MBT1HFP: "minimum bias HF+ threshold 1" bits
- MBT1HFM: "minimum bias HF- threshold 1" bits

In minimum bias trigger conditions module (`min_bias_hf_conditions.vhd`) there is a comparision with a 4-bit threshold (greater-equal [or equal]).

4.4.9 Towercount condition

Data for Towercount trigger (number of firing HCAL towers) are received on frame HT (see 4.4.7) on bits 24..12 (13 bits) of HT data structure.
In towercount condition module (`towercount_condition.vhd`) there is a comparision with a 13-bit threshold (greater-equal [or equal]).

4.4.10 Centrality condition

Centrality bits used as a signals for triggers (similar to external signals).

4.4.11 Muon charge correlation module

For definition of muon charge, see 4.4.4.
In the muon charge correlation module (`muon_charge_correlations.vhd`), the charge correlations are made for different muon conditions-types. The module is instantiated in the top-of-hierarchy module (`gtl_module.vhd`) and not inside of a muon conditions module. The charges of objects (number of objects depends on muon condition type) are compared to get "like sign charge" ("LS") or "opposite sign charge" ("0S"), "LS" means that the charges (charge sign) of objects are the same, "0S" means that at least one object has different charge than the others. This information is used in all instatiated muon conditions. There is no charge correlation for single type conditions.
In all cases the "charge valid" bit of the objects must be set.
In TME one can select "LS", "0S" or ignore for charge correlation in muon conditions.

Table 19: Muon charge correlation - Double Muon

x x	I ignore (charge x = +, -, I)
+ +	LS both positive muons
− −	LS both negative muons
I I	LS both muons with the same sign, positive or negative
+ −	OS two muons of opposite sign
− +	OS idem
I I	OS idem

Table 20: Muon charge correlation - Triple Muon

x x x	I ignore (charge x = +, -, I)
+ + +	LS three muons of positive charge
− − −	LS three muons of negative charge
I I I	LS three muons of the same sign (positive or negative)
+ + −	OS a pair plus a positive muon
+ − −	OS a pair plus a negative muon
+ − I	OS a pair plus a negative or positive muon

Table 21: Muon charge correlation - Quad Muon

x x x x	I ignore (charge x = +, -, I)
+ + + +	LS four muons of positive charge
− − − −	LS four muons of negative charge
I I I I	LS four muons of the same sign (positive or negative)
+ + + −	OS a pair plus two positive muons
+ + − −	OS two pairs
+ − − −	OS a pair plus two negative muons
+ − I I	OS a pair plus two negative or positive muons

4.4.12 Correlation conditions

The correlation conditions contain a combination of two "Single object requirement conditions" of two object types or one "Double objects requirement condition" of objects of the same type. In addition with object cuts there are correlation cuts for $\Delta\eta$, $\Delta\varphi$, ΔR, mass, mass divided by ΔR and "two-body pt".
The correlation condition of "Invariant mass for three objects" contains one "Triple objects requirement condition" of objects of the same type with one object cut for mass.

The following cuts can be used:

- Cut for $\Delta\eta$ (DETA).
- Cut for $\Delta\varphi$ (DPHI).
- Cut for ΔR (DR).
- Cuts for charge correlation (only for muon).
- Cuts for mass (MASS) of following mass types:
  - Cut for Invariant mass.
  - Cut for Invariant mass with unconstrained pt (only for muons).
  - Cut for Invariant mass divided by ΔR.
  - Cut for Transverse mass.
- Cut for Two-body pt.

There is one correlation condition type for a mass cut with three objects:

- Cut for invariant mass for three objects (MASS).

4.4.12.1 Calculation of cuts

Calculation of $\Delta\eta$ and $\Delta\varphi$ see section "Calculation of differences in η and φ" (4.4.5).

4.4.12.1.1 ΔR calculation

The calculation of ΔR of two objects is done with formula:

$$\Delta R = \sqrt{(\eta_1 - \eta_2)^2 + (\varphi_1 - \varphi_2)^2}.$$

In the TME there are two thresholds for ΔR: "greater/equal lower limit" and "less/equal upper limit", given in floating point notation with one position after decimal point. The comparison in VHDL is done with ΔR^2 (no square root in VHDL), thresholds for ΔR^2 are provided by VHDL-Producer.

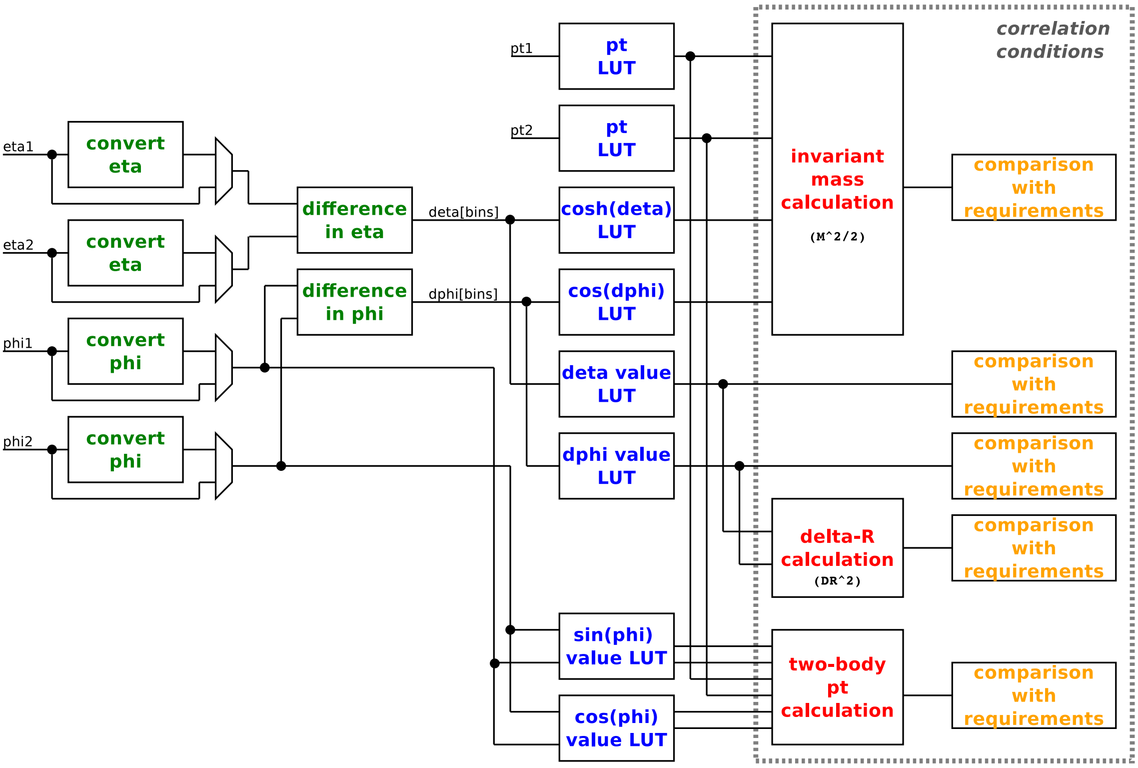


Figure 13: VHDL structure of cuts for correlation conditions

4.4.12.1.2 Invariant mass calculation

The calculation of *invariant mass of two objects* is done with formula:

$$M = \sqrt{2pt_1pt_2(\cosh(\eta_1 - \eta_2) - \cos(\varphi_1 - \varphi_2))}.$$

In the TME there are two thresholds for M: "greater/equal lower limit" and "less/equal upper limit", given in GeV (floating point notation) with one position after decimal point in even numbers.[2] The comparison in VHDL is done with $\frac{M^2}{2}$ (no square root in VHDL), thresholds for $\frac{M^2}{2}$ are provided by VHDL-Producer.

4.4.12.1.3 Transverse mass calculation

The calculation of *transverse mass of two objects* is done with formula:

$$M = \sqrt{2pt_1pt_2(1 - \cos(\varphi_1 - \varphi_2))}.$$

In the TME there are two thresholds for M: "greater/equal lower limit" and "less/equal upper limit", given in GeV (floating point notation) with one position after decimal point in even

---

[2] even numbers to get a precision of one position after decimal point after dividion by 2, because VHDL-Producer calculates thresholds for $\frac{M^2}{2}$, which includes a division by 2.

numbers.
The comparison in VHDL is done with $\frac{M^2}{2}$ (no square root in VHDL), thresholds for $\frac{M^2}{2}$ are provided by VHDL-Producer.

4.4.12.1.4 Two-body pt calculation

The calculation of *two-body pt* is done with formula:

$$\mathrm{pt}{=}\sqrt{pt_1^2+pt_2^2+2pt_1pt_2(\cos(\varphi_1)\cos(\varphi_2)+\sin(\varphi_1)\sin(\varphi_2))}$$

In the TME there is one threshold for pt, given in GeV (floating point notation) with one position after decimal point. The comparison in VHDL is done with pt^2 (no square root in VHDL), threshold for pt^2 is provided by VHDL-Producer.

4.4.12.1.5 Invariant mass over ΔR calculation

The formulas for *invariant mass over* ΔR *of two objects* are:

$$\mathrm{M}{=}\sqrt{2pt_1pt_2(\cosh(\eta_1-\eta_2)-\cos(\varphi_1-\varphi_2))}.$$

$$\Delta R=\sqrt{(\eta_1-\eta_2)^2+(\varphi_1-\varphi_2)^2}.$$

In the TME there is one threshold for M/ΔR, given in GeV (floating point notation) with one position after decimal point. The calculation of *invariant mass over* ΔR *of two objects* is done in an own module outside of the condition with $\frac{M^2}{2}\times(1/\Delta R^2)$ (no square root in VHDL).
A direct calculation of $1/\Delta R^2$ is not possible in firmware (VHDL code), therefore the implementation of the calculation is done by LUTs. In the hardware the values of these LUTs are stored in "large" ROMs, which was realized using the Block RAMs (BRAMs) of the Virtex chip.
Due the number of available BRAMs there are some restrictions for creating algorithms with *invariant mass over* ΔR:

- 1. Objects must have the same type (e.g.: muon_muon, eg_eg, ...)
- 2. Objects must be of same bx
- 3. Resolution of delta Eta and delta Phi:
  - Full resolution for calos (max. deta bins=230, max. dphi bins=72)
  - Half resolution only for muons (max. deta bins=226, max. dphi bins=144)
- 4. The precision of delta Eta and delta Phi:
  - calo_calo = 5

  - muon_muon = 6

- 5. If $1/\Delta R^2$=0 (deta=0 and dphi=0) then correlation cut *invariant mass over* ΔR is true
- 6. The values of LUTs are only valid for current definitions and restrictions. Every change might cause a recalculation of the values and a regeneration of IPs (representing LUTs in BRAMs) in Vivado (firmware generation tool)

The LUTs values (of $1/\Delta R^2$) are listed in `emulator_lut_calo_inv_dr_sq_calc.txt` and `emulator_lut_muon_inv_dr_sq_calc.txt`. These files have been created by scripts `calo_inv_dr_sq_calc.py` and `muon_inv_dr_sq_calc.py`.
In these files the following values are listed in columns (from left to right):

- 1. Difference of phi [bins]
- 2. Difference of eta [bins]
- 3. Value of difference in phi (with resolution shown in restrictions)
- 4. Value of difference in eta (with resolution shown in restrictions)
- 5. Value of $1/\Delta R^2$
- 6. Value of $1/\Delta R^2$ rounded (with value of precisions)
- 7. Integer value of $1/\Delta R^2$ rounded, multiplied with $10^{\text{precision}}$ (content of LUTs in firmware)

The values of LUTs in firmware are listed in coe files of ROMs (created by same scripts mentioned above), currently 5 ROMs for calo_calo and 9 ROMs for muon_muon (see `lut_calo_inv_dr_sq_rom1.coe`, etc. and `lut_muon_inv_dr_sq_rom1.coe`, etc.). The addresses of the BRAMs are given by $\Delta\eta$ and $\Delta\varphi$. All ROMs have 4096 addresses, exept ROM 9 for muon_muon with 8192 addresses. The data width of ROMs is different depending on the highest value in ROM. Because of these different data widths the partitioning of several ROMs was done to save BRAM resources. Only one calculation of *invariant mass over* ΔR is possible in one Virtex chip, but one can have some algorithms containing *invariant mass over* ΔR with different thresholds, but with same objects and same bx.
Following numbers of BRAMs (36kb) are needed:

- calo_calo: 660
- muon_muon: 630

[Available BRAM (36kb) resources per chip: 873]

4.4.12.1.6 Invariant mass calculation for three objects

The calculation of *invariant mass calculation for three objects* is done by calculating the invariant mass for all two-object combinations and take the sum of the three invariant masses of the two-object combinations.
In the TME there are two thresholds for M: "greater/equal lower limit" and "less/equal upper limit", given in GeV (floating point notation) with one position after decimal point in even numbers.

4.4.12.1.7 Overview of possible correlation cuts

The following list gives an overview of possible correlation cuts in conditions:

- Calo conditions:
  - two-body pt (for double condition)
- Calo conditions overlap removal:
  - $\Delta\eta$ overlap removal
  - $\Delta\varphi$ overlap removal
  - ΔR overlap removal
  - two-body pt (for double condition)
- Muon conditions:
  - charge correlation
  - two-body pt (for double condition)
- Calo calo correlation condition with calo overlap removal:
  - $\Delta\eta$ overlap removal
  - $\Delta\varphi$ overlap removal
  - ΔR overlap removal
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass
  - two-body pt
- Calo calo correlation condition:
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass

  - two-body pt
- Calo calo correlation condition for invariant mass divided by ΔR:
  - invariant mass divided by ΔR
- Calo calo correlation condition mass with three objects:
  - invariant mass with three objects
- Calo muon correlation condition:
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass
  - two-body pt
- Calo esums correlation condition:
  - $\Delta\varphi$
  - transverse mass
  - two-body pt
- Muon muon correlation condition:
  - charge correlation
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass or invariant mass unconstraint pt
  - two-body pt
- Muon muon correlation condition for invariant mass divided by ΔR:
  - charge correlation
  - invariant mass divided by ΔR
- Muon muon correlation condition mass with three objects:
  - charge correlation
  - invariant mass with three objects
- Muon esums correlation condition:
  - $\Delta\varphi$
  - transverse mass
  - two-body pt

4.4.12.2 Correlation condition modules

As described in section Correlation conditions (4.4.12), correlations of two object types are available. Therefore several modules are provided with possible correlations (objects 1-objects 2):

- Correlation condition with calorimeter objects (`calo_calo_correlation_condition.vhd`: electron/γ-electron/γ, electron/γ-jet, electron/γ-tau, jet-jet, jet-tau and tau-tau are possible.)
- Correlation condition for mass divided by ΔR with calorimeter objects (`calo_calo_mass_div_dr_condition.vhd`: electron/γ-electron/γ, electron/γ-jet, electron/γ-tau, jet-jet, jet-tau and tau-tau are possible.)
- Correlation condition with calorimeter objects and energy sum quantities (ET_{miss}, ET_{miss}^{HF} and HT_{miss} only) (`calo_esums_correlation_condition.vhd`: electron/γ-etm, jet-etm, tau-etm, electron/γ-htm, jet-htm, tau-htm, electron/γ-etmhf, jet-etmhf and tau-etmhf are possible.)
- Correlation condition with calorimeter objects and muons objects (`calo_muon_correlation_condition.vhd`: electron/γ-muon, jet-muon and tau-muon are possible.)
- Correlation condition with muon objects (`muon_muon_correlation_condition.vhd`)
- Correlation condition for mass divided by ΔR with muon objects (`muon_muon_mass_div_dr_condition.vhd`)
- Correlation condition with muon objects and energy sum quantities (ET_{miss}, ET_{miss}^{HF} and HT_{miss} only) (`muon_esums_correlation_condition.vhd`: muon-etm, muon-etmhf and muon-htm are possible.)

There are two modules for mass with three objects:

- Correlation condition for mass with three objects with calorimeter objects (same type, same bunch-crossing) (`calo_mass_3_obj_condition.vhd`
- Correlation condition for mass with three objects with muon objects (`muon_mass_3_obj_condition.vhd`)

4.4.12.2.1 Calo Calo Correlation condition module

The calo calo correlation condition module contains two "Single object requirement conditions" for different types of calo objects (electron/γ, jet or tau) or same type with data from different bunch-crossings as one possible mode and a "Double objects requirement condition"

for calo objects of same type and same bunch-crossing as a second mode (selection is done by a parameter in the generic list of `calo_calo_correlation_condition.vhd` named "same_bx").
In addition there are "Cuts" for differences in η ($\Delta\eta$) and φ ($\Delta\varphi$), a calculation of ΔR (DR), a calculation of invariant mass (MASS) and a calculation of two-body pt, see Figure 13.
The differences in η and φ are calculated in bins. These differences in bins are converted to numbers (by LUTs, e.g. EG_EG_DIFF_ETA_LUT, EG_EG_DIFF_PHI_LUT, ...), which represents values of differences (multiples of units in η and φ). These values given in the LUTs are calculated as floating-point values (based on the scales of η and φ), which are multiplied by a factor and truncated to an integer value. So, in the LUTs we have integer values, the factor is $10^{\text{precision}}$. This "precision" is a parameter given for certain LUTs.

Remark: Definitions of scales (see Tables 4.4.2, 7, 9 and 10):

- Calorimeter objects:
- η bin width $= \frac{0.087}{2}$ (bin 0 from 0.0 to $\frac{0.087}{2}$)
- ϕ bin width $= \frac{2\pi}{144}$ (bin 0 from 0.0 to $\frac{2\pi}{144}$)

The contents of the LUTs for cosh($\Delta\eta$) (EG_EG_COSH_DETA_LUT, ...) and cos($\Delta\varphi$) (EG_EG_COS_DPHI_LUT, ...) for invariant mass (formular see 4.4.12.1.2) are created by calculating hyperbolic cosine and cosine, rounding-up at the 3rd position after decimal point, and multiplying by 1000 to get integer values.[3]
The contents of the LUTs for cos(φ) (CALO_COS_PHI_LUT) and sin(φ) (CALO_SIN_PHI_LUT) for two-body pt (formular see 4.4.12.1.4) are created by calculating cosine and sine, rounding-up at the 3rd position after decimal point and multiplying by 1000 to get integer values.
The condition is complied, if at least one comparison between object parameters and requirements is valid for the both "Single object requirement condition" or the "Double objects requirement condition" and the results of selected "Cuts" are inside of a range (upper and lower limit) or greater/eual a threshold (e.g. for two-body pt). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for $\Delta\eta$ and $\Delta\varphi$ are expressed with a precision of 3rd position after decimal point, for DR, MASS and two-body pt with 1st position after decimal point.

For the VHDL entity declaration of calo calo correlation condition module in `calo_calo_correlation_condition.vhd`, see Listing 7.

---

[3] Definition of "`constant CALO_INV_MASS_COSH_COS_PRECISION...`" in file `gtl_pkg.vhd`. Value 1000 from $10^{\text{CALO\_INV\_MASS\_COSH\_COS\_PRECISION}}$.

Listing 7: Entity declaration of `calo_calo_correlation_condition.vhd`

```
entity calo_calo_correlation_condition is
    generic(
        same_bx: boolean;

        deta_cut: boolean;
        dphi_cut: boolean;
        dr_cut: boolean;
        mass_cut: boolean;
        mass_type : natural;
        twobody_pt_cut: boolean;

        calo1_object_low: natural;
        calo1_object_high: natural;
        et_ge_mode_calo1: boolean;
        obj_type_calo1: natural := EG_TYPE;
        et_threshold_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_calo1 : natural;
        eta_w1_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_calo1: boolean;
        phi_w1_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_calo1: boolean;
        phi_w2_upper_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_calo1: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        iso_lut_calo1: std_logic_vector(2**MAX_CALO_ISO_BITS-1 downto 0);

        calo2_object_low: natural;
        calo2_object_high: natural;
        et_ge_mode_calo2: boolean;
        obj_type_calo2: natural := JET_TYPE;
        et_threshold_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_calo2 : natural;
        eta_w1_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
```

```
        downto 0);
    eta_w1_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w2_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w2_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w3_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w3_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w4_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w4_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w5_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    eta_w5_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    phi_full_range_calo2: boolean;
    phi_w1_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    phi_w1_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    phi_w2_ignore_calo2: boolean;
    phi_w2_upper_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    phi_w2_lower_limit_calo2: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
        downto 0);
    iso_lut_calo2: std_logic_vector(2**MAX_CALO_ISO_BITS-1 downto 0);

    diff_eta_upper_limit_vector: std_logic_vector(
        MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
    diff_eta_lower_limit_vector: std_logic_vector(
        MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

    diff_phi_upper_limit_vector: std_logic_vector(
        MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
    diff_phi_lower_limit_vector: std_logic_vector(
        MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

    dr_upper_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
        downto 0);
    dr_lower_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
        downto 0);

    mass_upper_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
        downto 0);
    mass_lower_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
        downto 0);

    pt1_width: positive;
    pt2_width: positive;
    mass_cosh_cos_precision : positive;
    cosh_cos_width: positive;

    pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
```

```
            downto 0);
        sin_cos_width: positive;
        pt_sq_sin_cos_precision : positive

    );
    port(
        lhc_clk: in std_logic;
        calo1_data_i: in calo_objects_array;
        calo2_data_i: in calo_objects_array;
        diff_eta: in deta_dphi_vector_array;
        diff_phi: in deta_dphi_vector_array;
        pt1 : in diff_inputs_array;
        pt2 : in diff_inputs_array;
        cosh_deta : in calo_cosh_cos_vector_array;
        cos_dphi : in calo_cosh_cos_vector_array;
        cos_phi_1_integer : in sin_cos_integer_array;
        cos_phi_2_integer : in sin_cos_integer_array;
        sin_phi_1_integer : in sin_cos_integer_array;
        sin_phi_2_integer : in sin_cos_integer_array;
        condition_o: out std_logic
    );
end calo_calo_correlation_condition;
```

Table 22: Explanation of Listing 7

Item	Explanation
`same_bx`	boolean indicating whether data are from same Bx - 'true' for same Bx.
`deta_cut`	boolean for using $\Delta\eta$ cut.
`dphi_cut`	boolean for using $\Delta\varphi$ cut.
`dr_cut`	boolean for using DR cut.
`mass_cut`	boolean for using MASS cut.
`mass_type`	selection of mass type (INVARIANT_MASS_TYPE, INVARIANT_MASS_PT_TYPE, TRANSVERSE_MASS_TYPE or TRANSVERSE_MASS_PT_TYPE are allowed).
`calo1_object_low`	low index of object range (valid numbers: 0..11).
`calo1_object_high`	high index of object range (valid numbers: 0..11, but greater or equal calo1_object_low).
`et_ge_mode_calo1`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only).
`obj_type_calo1`	selection of calo1 object type (EG_TYPE, JET_TYPE or TAU_TYPE are allowed)
`et_threshold_calo1`	threshold value for comparison in E_{T}.
`nr_eta_windows_calo1`	integer value for number of η cuts.
`eta_w1_upper_limit_calo1`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_calo1`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_calo1`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_calo1`	"lower limit" of "window"-comparator 2 for η.

Table 22: Explanation of Listing 7

Item	Explanation
`eta_w3_upper_limit_calo1`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_calo1`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_calo1`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_calo1`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_calo1`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_calo1`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_calo1`	boolean to set full range of φ.
`phi_w1_upper_limit_calo1`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_calo1`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_calo1`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_calo1`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_calo1`	"lower limit" of "window"-comparator 2 for φ.
`iso_lut_calo1`	content of LUT (4 bits) for isolation comparison.
`calo2_object_low`	low index of object range (valid numbers: 0..11).
`calo2_object_high`	high index of object range (valid numbers: 0..11, but greater or equal calo2_object_low).
`et_ge_mode_calo2`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`obj_type_calo2`	selection of calo2 object type (EG_TYPE, JET_TYPE or TAU_TYPE are allowed)
`et_threshold_calo2`	threshold value for comparison in E_{T}.
`nr_eta_windows_calo2`	integer value for number of η cuts.
`eta_w1_upper_limit_calo2`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_calo2`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_calo2`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_calo2`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_calo2`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_calo2`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_calo2`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_calo2`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_calo2`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_calo2`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_calo2`	boolean to set full range of φ.
`phi_w1_upper_limit_calo2`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_calo2`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_calo2`	boolean to ignore "window"-comparator 2 for φ.

Table 22: Explanation of Listing 7

Item	Explanation
`phi_w2_upper_limit_calo2`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_calo2`	"lower limits" of "window"-comparator 2 for φ.
`iso_lut_calo2`	content of LUT (4 bits) for isolation comparison.
`diff_eta_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_eta_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_phi_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in φ (hex value).
`diff_phi_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in φ (hex value).
`dr_upper_limit`	"upper limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`dr_lower_limit`	"lower limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`DETA_DPHI_VECTOR_WIDTH`	vector width of $\Delta\eta$ and $\Delta\varphi$ for calculation of ΔR^2.
`DETA_DPHI_PRECISION`	position after decimal point for $\Delta\eta$ and $\Delta\varphi$.
`mass_upper_limit`	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`mass_lower_limit`	"lower limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`MASS_PRECISION`	position after decimal point for $\frac{M^2}{2}$.
`pt1_width`	number of bits of pt1.
`pt2_width`	number of bits of pt2.
`MASS_COSH_COS_PRECISION`	position after decimal point for cosh($\Delta\eta$) and cos($\Delta\varphi$).
`cosh_cos_width`	number of bits for the maximum value in the LUT for cosh($\Delta\eta$).
`pt_sq_threshold`	threshold value for comparison in two-body pt (pt^2).
`sin_cos_width`	number of bits for the maximum value in the LUT for cos(φ) and sin(φ).
`PT_PRECISION`	position after decimal point for pt^2.
`PT_SQ_SIN_COS_PRECISION`	position after decimal point for cos(φ) and sin(φ).
`lhc_clk`	clock input (LHC clock).
`calo1_data_i`	calorimeter input data, structure defined with `obj_type_calo1`.
`calo2_data_i`	calorimeter input data, structure defined with `obj_type_calo2`.
`diff_eta`	differences in η, calculated in an instance of module `sub_eta_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`), see 4.4.5.
`diff_phi`	differences in φ, calculated in an instance of module `sub_phi_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`).
`pt1`	calo1 E_T values [from LUT, in $GeV \times 10$].[4]

---

[4] value 10 from $10^{\mathrm{CALO\_INV\_MASS\_PT\_PRECISION}}$

Table 22: Explanation of Listing 7

Item	Explanation
`pt2`	calo2 E_T values [from LUT, in $GeV \times 10$].
`cosh_deta`	cosh($\Delta\eta$) values [from LUT, cosh($\Delta\eta$)×1000].[5]
`cos_dphi`	cos($\Delta\varphi$) values [from LUT, cosh($\Delta\varphi$)×1000].
`cos_phi_1`	cos(φ) values from LUT for calo1.
`cos_phi_2`	cos(φ) values from LUT for calo2.
`sin_phi_1`	sin(φ) values from LUT for calo1.
`sin_phi_2`	sin(φ) values from LUT for calo2.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

4.4.12.2.2 Calo Calo Overlap Remover Correlation condition module

The Calo Calo Overlap Remover Correlation conditions consits of two modes. One with a Calo Calo Correlation condition with "Double objects requirement condition" for calo objects of same type and same bunch-crossing (4.4.12.2.1) and a single condition for a different calo object type (can have different bunch-crossing too). There has to be at least one correlation cut for the objects of "Double objects requirement condition" and a correlation cut for overlap removal between objects (one or more cut(s) of $\Delta\eta$, $\Delta\varphi$ and ΔR) of different object types ("2plus1"). A second mode ("1plus1") with a Calo Calo Correlation condition with a single condition and a different calo object type (can have different bunch-crossing too) also with a single condition. There has to be at least one correlation cut for the different objects (e.g. invariant mass) and a correlation cut for overlap removal between the objects (one or more cut(s) of $\Delta\eta$, $\Delta\varphi$ and ΔR).

Overlap Remover Correlation conditions `calo_calo_calo_correlation_orm_condition.vhd` are implemented only for calo object types.

4.4.12.2.3 Calo Calo Correlation condition module for Invariant Mass Divided by ΔR

The calo calo correlation condition module for invariant mass divided by ΔR contains two "Single object requirement conditions" for different types of calo objects (electron/γ, jet or tau) or same type with data from different bunch-crossings as one possible mode and a "Double objects requirement condition" for calo objects of same type and same bunch-crossing as a second mode (selection is done by a parameter in the generic list of `calo_calo_mass_div_dr_condition.vhd` named "same_bx").
The calculation of *invariant mass divided by ΔR of two objects* is done in an own module outside of the condition (`mass_div_dr_calculator.vhd`), see 4.4.12.1.5. This module is instantiated once for every object type bunch-crossing combination.

---

[5] value 1000 from $10^{\text{CALO\_INV\_MASS\_COSH\_COS\_PRECISION}}$

The comparison of calculated values and threshold is done inside the module (`calo_calo_mass_div_dr_condition.vhd`).
In the TME there is one threshold for M/ΔR: "greater/equal threshold", given in GeV (floating point notation).
The threshold for comparison with $\frac{M^2}{2}\times(1/\Delta R^2)$ (no square root in VHDL) is provided by VHDL-Producer.
No other correlation cuts available in this condition type.

4.4.12.2.4 Calo Correlation condition module for Invariant Mass with Three Objects

The calo correlation condition module for invariant mass with three objects (`calo_mass_3_obj_condition.vhd)` contains a "Triple objects requirement condition" for calo objects of same type and same bunch-crossing.
In addition a "Cut" for calculation of *invariant mass with three objects* (see 4.4.12.1.6) is mandatory.
No other correlation cuts available in this condition type.

4.4.12.2.5 Calo Esums Correlation condition module

The calo esums correlation condition module (`calo_esums_correlation_condition.vhd`) contains two "Single object requirement conditions", one of calo objects (electron/γ, jet or tau) and one of esums (ET_{miss}, ET^{HF}_{miss} or HT_{miss}).
In addition there are "Cuts" for differences in φ ($\Delta\varphi$) or a calculation of mass (MASS) for Transverse mass or Transverse mass with two-body pt.
The differences in φ are calculated in bins. These differences in bins are converted to numbers (by LUTs, e.g. EG_ETM_DIFF_PHI_LUT, ...), which represents values of differences (multiples of units in φ). These values given in the LUTs are calculated as floating-point values (based on the scales of φ), which are multiplied by a factor and truncated to an integer value. So, in the LUTs we have integer values, the factor is $10^{\mathrm{precision}}$.

The contents of the LUTs cos($\Delta\varphi$) (EG_ETM_COS_DPHI_LUT, ...) for Transverse mass (formular see 4.4.12.1.3) are created by calculating cosine, rounding-up at the 3rd position after decimal point and multiplying by 1000 to get integer values.[6]
The contents of the LUTs for cos(φ) (CALO_COS_PHI_LUT) and sin(φ) (CALO_SIN_PHI_LUT) for two-body pt (formular see 4.4.12.1.4) are created by calculating cosine and sine, rounding-up at the 3rd position after decimal point and multiplying by 1000 to get integer values.
The condition is complied, if at least one comparison between object parameters and requirements is valid for the both "Single object requirement condition" and the results of selected "Cuts" are inside of a range (upper and lower limit). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for $\Delta\varphi$

---

[6] Definition of `"constant CALO_INV_MASS_COSH_COS_PRECISION..."` in file `gtl_pkg.vhd`. 1000 from $10^{\mathrm{CALO\_INV\_MASS\_COSH\_COS\_PRECISION}}$.

are expressed with a precision of 3rd position after decimal point, for MASS with 1st position after decimal point.

For VHDL entity declaration for calo esums correlation condition module in `calo_esums_correlation_condition.vhd`, see Listing 8.

Listing 8: Entity declaration of `calo_esums_correlation_condition.vhd`

```
entity calo_esums_correlation_condition is
     generic(

        dphi_cut: boolean;
        mass_cut: boolean;
        mass_type : natural;
        twobody_pt_cut: boolean;

        calo_object_low: natural;
        calo_object_high: natural;
        et_ge_mode_calo: boolean;
        obj_type_calo: natural := EG_TYPE;
        et_threshold_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_calo : natural;
        eta_w1_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_calo: boolean;
        phi_w1_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_calo: boolean;
        phi_w2_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        iso_lut_calo: std_logic_vector(2**MAX_CALO_ISO_BITS-1 downto 0);

        et_ge_mode_esums: boolean;
        obj_type_esums: natural := ETM_TYPE;
```

```
        et_threshold_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_full_range_esums: boolean;
        phi_w1_upper_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_esums: boolean;
        phi_w2_upper_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);

        diff_phi_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_phi_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

        mass_upper_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);
        mass_lower_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);

        pt1_width: positive;
        pt2_width: positive;
        mass_cosh_cos_precision : positive;
        cosh_cos_width: positive;

        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0);
        sin_cos_width: positive;
        pt_sq_sin_cos_precision : positive

    );
    port(
        lhc_clk: in std_logic;
        calo_data_i: in calo_objects_array;
        esums_data_i: in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        diff_phi: in deta_dphi_vector_array;
        pt1 : in diff_inputs_array;
        pt2 : in diff_inputs_array;
        cos_dphi : in calo_cosh_cos_vector_array;
        cos_phi_1_integer : in sin_cos_integer_array;
        cos_phi_2_integer : in sin_cos_integer_array;
        sin_phi_1_integer : in sin_cos_integer_array;
        sin_phi_2_integer : in sin_cos_integer_array;
        condition_o: out std_logic
    );
end calo_esums_correlation_condition;
```

Table 23: Explanation of Listing 8

Item	Explanation
`dphi_cut`	boolean for using $\Delta\varphi$ cut.
`mass_cut`	boolean for using MASS cut.
`mass_type`	selection of mass type (TRANSVERSE_MASS_TYPE or TRANSVERSE_MASS_PT_TYPE are allowed).
`calo_object_low`	low index of object range (valid numbers: 0..11).
`calo_object_high`	high index of object range (valid numbers: 0..11, but greater or equal calo_object_low).
`et_ge_mode_calo`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only).
`obj_type_calo`	selection of calo1 object type (EG_TYPE, JET_TYPE or TAU_TYPE are allowed)
`et_threshold_calo`	threshold value for comparison in E_{T}.
`nr_eta_windows_calo`	integer value for number of η cuts.
`eta_w1_upper_limit_calo`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_calo`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_calo`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_calo`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_calo`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_calo`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_calo`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_calo`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_calo`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_calo`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_calo`	boolean to set full range of φ.
`phi_w1_upper_limit_calo`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_calo`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_calo`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_calo`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_calo`	"lower limit" of "window"-comparator 2 for φ.
`iso_lut_calo`	content of LUT (4 bits) for isolation comparison.
`et_ge_mode_esums`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`obj_type_esums`	selection of esums type (ETM_TYPE, ETMHF_TYPE or HTM_TYPE are allowed)
`et_threshold_esums`	threshold value for comparison in E_{T}.
`phi_full_range_esums`	boolean to set full range of φ.
`phi_w1_upper_limit_esums`	"upper limit" of "window"-comparator 1 for φ.

Table 23: Explanation of Listing 8

Item	Explanation
`phi_w1_lower_limit_esums`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_esums`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_esums`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_esums`	"lower limits" of "window"-comparator 2 for φ.
`diff_phi_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in φ (hex value).
`diff_phi_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in φ (hex value).
`DETA_DPHI_VECTOR_WIDTH`	vector width of $\Delta\varphi$.
`DETA_DPHI_PRECISION`	position after decimal point for $\Delta\varphi$.
`mass_upper_limit`	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`mass_lower_limit`	"lower limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`MASS_PRECISION`	position after decimal point for $\frac{M^2}{2}$.
`pt1_width`	number of bits of pt1.
`pt2_width`	number of bits of pt2.
`MASS_COSH_COS_PRECISION`	position after decimal point for $\cos(\Delta\varphi)$.
`cosh_cos_width`	number of bits for the maximum value in the LUT for $\cos(\Delta\varphi)$.
`pt_sq_threshold`	threshold value for comparison in two-body pt (pt^2).
`sin_cos_width`	number of bits for the maximum value in the LUT for $\cos(\varphi)$ and $\sin(\varphi)$.
`PT_PRECISION`	position after decimal point for pt^2.
`PT_SQ_SIN_COS_PRECISION`	position after decimal point for $\cos(\varphi)$ and $\sin(\varphi)$.
`lhc_clk`	clock input (LHC clock).
`calo_data_i`	calorimeter input data, structure defined with `obj_type_calo1`.
`esums_data_i`	esums input data, structure defined with `obj_type_esums`.
`diff_phi`	differences in φ, calculated in an instance of module `sub_phi_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`).
`pt1`	calo1 E_T values [from LUT, in $GeV \times 10$].[7]
`pt2`	esums E_T values [from LUT, in $GeV \times 10$].
`cos_dphi`	$\cos(\Delta\varphi)$ values from LUT.
`cos_phi_1`	$\cos(\varphi)$ values from LUT for calo1.
`cos_phi_2`	$\cos(\varphi)$ values from LUT for esums.
`sin_phi_1`	$\sin(\varphi)$ values from LUT for calo1.
`sin_phi_2`	$\sin(\varphi)$ values from LUT for esums.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

---

[7] value 10 from $10^{\mathrm{CALO\_INV\_MASS\_PT\_PRECISION}}$

4.4.12.2.6 Calo Muon Correlation condition module

The calo muon correlation condition module (`calo_muon_correlation_condition.vhd`) contains a "Single object requirement condition" for one type of calo objects (electron/γ, jet or tau) and a "Single object requirement condition" for muon objects. In addition there are "Cuts" for differences in η ($\Delta\eta$) and φ ($\Delta\varphi$), a calculation of ΔR (DR), a calculation of invariant mass (MASS) and a calculation of two-body pt, see Figure 13.
The differences in η and φ are calculated in bins. These differences in bins are converted to numbers (by LUTs, e.g. EG_MU_DIFF_ETA_LUT, EG_MU_DIFF_PHI_LUT, ...), which represents values of differences (multiples of units in η and φ). These values given in the LUTs are calculated as floating-point values (based on the scales of η and φ), which are multiplied by a factor and truncated to an integer value. So, in the LUTs we have integer values, the factor is $10^{\text{precision}}$. This "precision" is a parameter given for certain LUTs.
Because of the different scales of calorimeter and muon objects in η and φ, there are LUTs for conversion the calorimeter bins to muon bins (in `gtl_pkg.vhd`: e.g. EG_ETA_CONV_2_MUON_ETA_LUTand EG_PHI_CONV_2_MUON_PHI_LUT).

Remark:
The center value of bins are used as reference value for conversion. The content of EG_ETA_CONV_2_MUON_ETA_LUTis calculated with formular:
"converted-calo-eta[bin] = calo-eta[bin] $\times$ 4 + 2",
of EG_PHI_CONV_2_MUON_PHI_LUTwith formular:
"converted-calo-phi[bin] = calo-phi[bin] $\times$ 4 + 2".
The conversion calculations are preliminary, others may be proposed.

Definitions of scales (see Tables 4.4.2, 7, 9 and 10):

- Calorimeter objects:
  - η bin width = $\frac{0.087}{2}$ (bin 0 from 0.0 to $\frac{0.087}{2}$)
  - ϕ bin width = $\frac{2\pi}{144}$ (bin 0 from 0.0 to $\frac{2\pi}{144}$)
- Muon objects:
  - η bin width = $\frac{0.087}{8}$ (bin 0 from $0.5\times\frac{-0.087}{8}$ to $0.5\times\frac{+0.087}{8}$)
  - ϕ bin width = $\frac{2\pi}{576}$ (bin 0 from 0.0 to $\frac{2\pi}{576}$)

The contents of the LUTs for cosh($\Delta\eta$) (EG_MUON_COSH_DETA_LUT, ...) and cos($\Delta\varphi$) (EG_MUON_COS_DPHI_LUT, ...) for invariant mass (formular see 4.4.12.1.2) are created by calculating hyperbolic cosine and cosine, rounding-up at the 4th position after decimal point, and multiplying by 10000($10^{\text{CALO\_MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$) to get integer values.[8]
The contents of the LUTs for cos(φ) (CALO_COS_PHI_LUT and MUON_COS_PHI_LUT) and sin(φ) (CALO_SIN_PHI_LUT and MUON_SIN_PHI_LUT) for two-body pt (formular see 4.4.12.1.4) are created by calculating cosine and sine, rounding-up at the 3rd position after

---

[8] Definition of "`constant CALO_MUON_INV_MASS_COSH_COS_PRECISION` ...", "`constant EG_ETA_CONV_2_MUON_ETA_LUT` ..." and "`constant EG_PHI_CONV_2_MUON_PHI_LUT` ..." in file `gtl_pkg.vhd`.

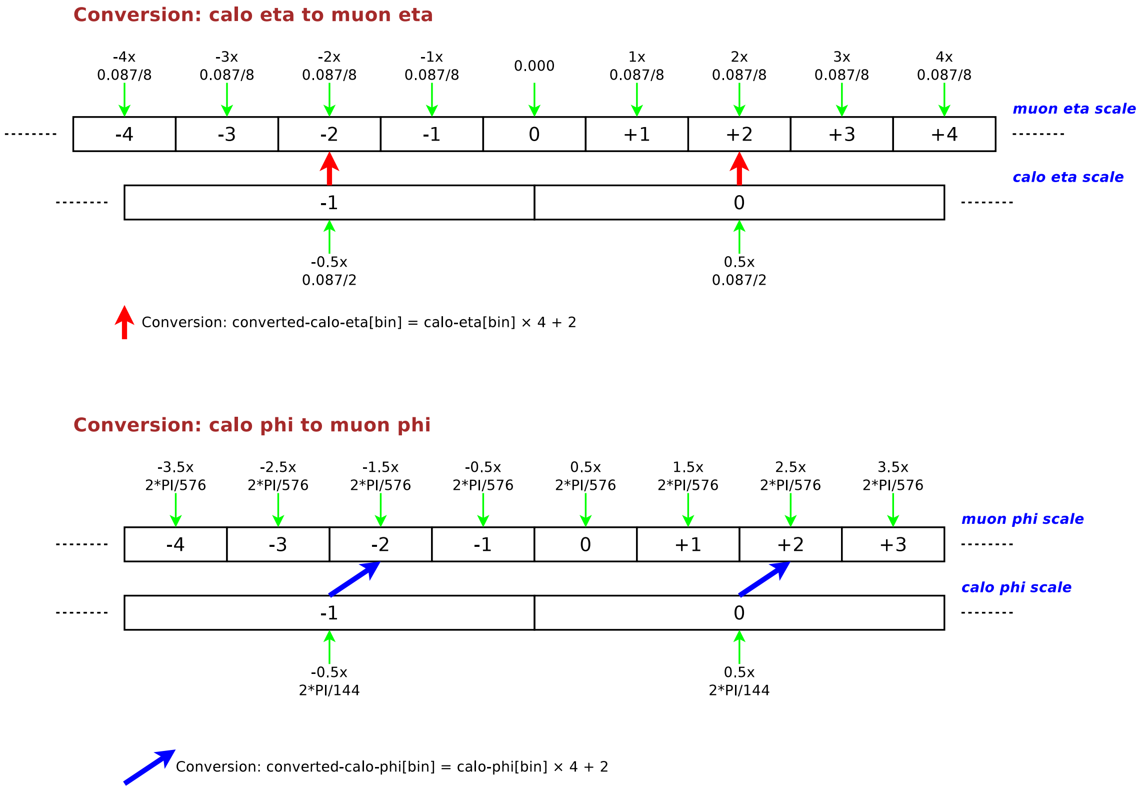


Figure 14: Conversion of calorimeter η and φ to muon scales

decimal point, and multiplying by 1000 to get integer values.
The condition is complied, if at least one comparison between object parameters and requirements is valid for the both "Single object requirement condition" and the results of selected "Cuts" are inside of a range (upper and lower limit) or greater/eual a threshold (e.g. for two-body pt). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for $\Delta\eta$ and $\Delta\varphi$ are expressed with a precision of 3^{rd} position after decimal point, for DR, MASS and two-body pt with 1^{st} position after decimal point.

For the VHDL entity declaration of calo muon correlation condition module in `calo_muon_correlation_condition.vhd`, see Listing 9.

Listing 9: Entity declaration of `calo_muon_correlation_condition.vhd`

```
entity calo_muon_correlation_condition is
     generic(
        deta_cut: boolean;
        dphi_cut: boolean;
        dr_cut: boolean;
        mass_cut: boolean;
        mass_type : natural;
        twobody_pt_cut: boolean;

        calo_object_low: natural;
        calo_object_high: natural;
        et_ge_mode_calo: boolean;
        obj_type_calo: natural := EG_TYPE;
        et_threshold_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_calo : natural;
        eta_w1_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_calo: boolean;
        phi_w1_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_calo: boolean;
        phi_w2_upper_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_calo: std_logic_vector(MAX_CALO_TEMPLATES_BITS-1
            downto 0);
        iso_lut_calo: std_logic_vector(2**MAX_CALO_ISO_BITS-1 downto 0);

        muon_object_low: natural;
        muon_object_high: natural;
        pt_ge_mode_muon: boolean;
        pt_threshold_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_muon : natural;
        eta_w1_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
```

```
        eta_w2_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_muon : boolean;
        phi_w1_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_muon : boolean;
        phi_w2_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        requested_charge_muon: string(1 to 3);
        qual_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.qual_high-D_S_I_MUON_V2
            .qual_low+1)-1 downto 0);
        iso_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.iso_high-D_S_I_MUON_V2.
            iso_low+1)-1 downto 0);
        upt_cut_muon : boolean;
        upt_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
        upt_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
        ip_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.ip_high-D_S_I_MUON_V2.
            ip_low+1)-1 downto 0);

        diff_eta_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_eta_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

        diff_phi_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_phi_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

        dr_upper_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0);
        dr_lower_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0);

        mass_upper_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);
```

```
        mass_lower_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);

        pt1_width: positive;
        pt2_width: positive;
        mass_cosh_cos_precision : positive;
        cosh_cos_width: positive;

        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0);
        sin_cos_width: positive;
        pt_sq_sin_cos_precision : positive

    );
    port(
        lhc_clk: in std_logic;
        calo_data_i: in calo_objects_array;
        muon_data_i: in muon_objects_array;
        diff_eta: in deta_dphi_vector_array;
        diff_phi: in deta_dphi_vector_array;
        pt1 : in diff_inputs_array;
        pt2 : in diff_inputs_array;
        cosh_deta : in calo_muon_cosh_cos_vector_array;
        cos_dphi : in calo_muon_cosh_cos_vector_array;
        cos_phi_1_integer : in sin_cos_integer_array;
        cos_phi_2_integer : in sin_cos_integer_array;
        sin_phi_1_integer : in sin_cos_integer_array;
        sin_phi_2_integer : in sin_cos_integer_array;
        condition_o: out std_logic
    );
end calo_muon_correlation_condition;
```

Table 24: Explanation of Listing 9

Item	Explanation
`deta_cut`	boolean for using $\Delta\eta$ cut.
`dphi_cut`	boolean for using $\Delta\varphi$ cut.
`dr_cut`	boolean for using DR cut.
`mass_cut`	boolean for using MASS cut.
`mass_type`	selection of mass type (INVARIANT_MASS_TYPE, INVARIANT_MASS_PT_TYPE, TRANSVERSE_MASS_TYPE or TRANSVERSE_MASS_PT_TYPE are allowed).
`calo_object_low`	low index of object range (valid numbers: 0..11).
`calo_object_high`	high index of object range (valid numbers: 0..11, but greater or equal calo_object_low).
`calo_et_ge_mode_calo`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only).
`obj_type_calo`	selection of calo object type (EG_TYPE, JET_TYPE or TAU_TYPE are allowed)
`et_threshold_calo`	threshold value for comparison in E_{T}.

Table 24: Explanation of Listing 9

Item	Explanation
`nr_eta_windows_calo`	integer value for number of η cuts.
`eta_w1_upper_limit_calo`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_calo`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_calo`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_calo`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_calo`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_calo`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_calo`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_calo`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_calo`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_calo`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_calo`	boolean to set full range of φ.
`phi_w1_upper_limit_calo`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_calo`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_calo`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_calo`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_calo`	"lower limit" of "window"-comparator 2 for φ.
`iso_lut_calo`	content of LUT (4 bits) for isolation comparison.
`muon_object_low`	low index of object range (valid numbers: 0..7).
`muon_object_high`	high index of object range (valid numbers: 0..7, but greater or equal muon_object_low).
`pt_ge_mode_muon`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_threshold_muon`	threshold value for comparison in p_{T}.
`nr_eta_windows_muon`	integer value for number of η cuts.
`eta_w1_upper_limit_muon`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_muon`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_muon`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_muon`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_muon`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_muon`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_muon`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_muon`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_muon`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_muon`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_muon`	boolean to set full range of φ.

Table 24: Explanation of Listing 9

Item	Explanation
`phi_w1_upper_limit_muon`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_muon`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_muon`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_muon`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_muon`	"lower limits" of "window"-comparator 2 for φ.
`requested_charge_muon`	string for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
`qual_lut_muon`	content of LUT (16 bits) for quality comparison.
`iso_lut_muon`	content of LUT (4 bits) for isolation comparison.
`upt_cut_muon`	boolean for using unconstrained p_{T} cuts.
`upt_upper_limit_muon`	"upper limit" of unconstrained p_{T}.
`upt_lower_limit_muon`	"lower limit" of unconstrained p_{T}.
`ip_lut_muon`	content of LUTs (4 bits) for impact parameter.
`diff_eta_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_eta_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_phi_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in φ (hex value).
`diff_phi_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in φ (hex value).
`dr_upper_limit`	"upper limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`dr_lower_limit`	"lower limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`DETA_DPHI_VECTOR_WIDTH`	vector width of $\Delta\eta$ and $\Delta\varphi$ for calculation of ΔR^2.
`DETA_DPHI_PRECISION`	position after decimal point for $\Delta\eta$ and $\Delta\varphi$.
`mass_upper_limit`	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`mass_lower_limit`	"lower limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`MASS_PRECISION`	position after decimal point for $\frac{M^2}{2}$.
`pt1_width`	number of bits of pt1.
`pt2_width`	number of bits of pt2.
`MASS_COSH_COS_PRECISION`	position after decimal point for $\cosh(\Delta\eta)$ and $\cos(\Delta\varphi)$.
`cosh_cos_width`	number of bits for the maximum value in the LUT for $\cosh(\Delta\eta)$.
`pt_sq_threshold`	threshold value for comparison in two-body pt (pt^2).
`sin_cos_width_1`	number of bits for the maximum value in the LUT for $\cos(\varphi)$ and $\sin(\varphi)$ of calos.
`sin_cos_width_2`	number of bits for the maximum value in the LUT for $\cos(\varphi)$ and $\sin(\varphi)$ of muon.

Table 24: Explanation of Listing 9

Item	Explanation
`PT_PRECISION`	position after decimal point for pt^2.
`PT_SQ_SIN_COS_PRECISION`	position after decimal point for $\cos(\varphi)$ and $\sin(\varphi)$.
`lhc_clk`	clock input (LHC clock).
`calo_data_i`	calorimeter input data, structure defined with `obj_type_calo`.
`muon_data_i`	muon input data.
`diff_eta`	differences in η, calculated in an instance of module `sub_eta_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`), see 4.4.5.
`diff_phi`	differences in φ, calculated in an instance of module `sub_phi_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`).
`pt1`	calo E_T values [from LUT, in $GeV\times 10$].[9]
`pt2`	muon p_T values [from LUT, in $GeV\times 10$].
`cosh_deta`	$\cosh(\Delta\eta)$ values [from LUT, $\cosh(\Delta\eta)\times 10000$].[10]
`cos_dphi`	$\cos(\Delta\varphi)$ values [from LUT, $\cosh(\Delta\varphi)\times 10000$].
`cos_phi_1`	$\cos(\varphi)$ values from LUT for calo.
`cos_phi_2`	$\cos(\varphi)$ values from LUT for muon.
`sin_phi_1`	$\sin(\varphi)$ values from LUT for calo.
`sin_phi_2`	$\sin(\varphi)$ values from LUT for muon.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

4.4.12.2.7 Muon Muon Correlation condition module

The muon muon correlation condition module contains two "Single object requirement conditions" for data from different bunch-crossings as one possible mode and a "Double objects requirement condition" for muon objects at same bunch-crossing as a second mode (selection is done by a parameter in the generic list of `muon_muon_correlation_condition.vhd` named "same_bx"). In the case of a "Double objects requirement condition", requirements for "requested charge correlations" are used and a muon charge correlation module (see 4.4.11) is required.

In addition there are "Cuts" for differences in η ($\Delta\eta$) and φ ($\Delta\varphi$), a calculation of ΔR (DR), a calculation of invariant mass with pt or of invariant mass with unconstrained pt (MASS), a calculation of two-body pt.

The differences in η and φ are calculated in bins. These differences in bins are converted to numbers (by LUTs, e.g. MUON_MUON_DIFF_ETA_LUT, MUON_MUON_DIFF_PHI_LUT), which represents values of differences (multiples of units in η and φ). These values

---

[9] value 10 from $10^{\mathrm{CALO\_MUON\_INV\_MASS\_PT\_PRECISION}}$

[10] value 10000 from $10^{\mathrm{CALO\_MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$

given in the LUTs are calculated as floating-point values (based on the scales of η and φ), which are multiplied by a factor and truncated to an integer value. So, in the LUTs we have integer values, the factor is $10^{\text{precision}}$. This "precision" is a parameter given for certain LUTs.

Remark: Definitions of scales (see Tables 9 and 10):

- Muon objects:
- η bin width $= \frac{0.087}{8}$ (bin 0 from $0.5 \times \frac{-0.087}{8}$ to $0.5 \times \frac{+0.087}{8}$)
- ϕ bin width $= \frac{2\pi}{576}$ (bin 0 from 0.0 to $\frac{2\pi}{576}$)

The contents of the LUTs for cosh($\Delta\eta$) (MUON_MUON_COSH_DETA_LUT) and cos($\Delta\varphi$) (MUON_MUON_COS_DPHI_LUT) for invariant mass (formular see 4.4.12.1.2) are created by calculating hyperbolic cosine and cosine, rounding-up at the 4th position after decimal point, and multiplying by 10000 to get integer values.[11]
The contents of the LUTs for cos(φ) (MUON_COS_PHI_LUT) and sin(φ) (MUON_SIN_PHI_LUT) for two-body pt (formular see 4.4.12.1.4) are created by calculating cosine and sine, rounding-up at the 3rd position after decimal point, and multiplying by 1000 to get integer values.
The condition is complied, if at least one comparison between object parameters and requirements is valid for the both "Single object requirement condition" or the "Double objects requirement condition" and the results of selected "Cuts" are inside of a range (upper and lower limit) or greater/eual a threshold (e.g. for two-body pt). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for $\Delta\eta$ and $\Delta\varphi$ are expressed with a precision of 3rd position after decimal point, for DR and MASS with 1st position after decimal point.

For the VHDL entity declaration of muon muon correlation condition module in `muon_muon_correlation_condition.vhd`, see Listing 10.

---

[11] Definition of "`constant MUON_INV_MASS_COSH_COS_PRECISION`" in file `gtl_pkg.vhd`. Value 10000 from $10^{\text{MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$.

Listing 10: Entity declaration of `muon_muon_correlation_condition.vhd`

```
entity muon_muon_correlation_condition is
     generic(
        same_bx: boolean;

        deta_cut: boolean;
        dphi_cut: boolean;
        dr_cut: boolean;
        mass_cut: boolean;
        mass_type : natural;
        twobody_pt_cut: boolean;

        muon1_object_low: natural;
        muon1_object_high: natural;
        pt_ge_mode_muon1: boolean;
        pt_threshold_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_muon1: natural;
        eta_w1_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w2_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_muon1: boolean;
        phi_w1_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_muon1: boolean;
        phi_w2_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        requested_charge_muon1: string(1 to 3);
        qual_lut_muon1: std_logic_vector(2**(D_S_I_MUON_V2.qual_high-
            D_S_I_MUON_V2.qual_low+1)-1 downto 0);
        iso_lut_muon1: std_logic_vector(2**(D_S_I_MUON_V2.iso_high-D_S_I_MUON_V2.
            iso_low+1)-1 downto 0);
        upt_cut_muon1 : boolean;
        upt_upper_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
        upt_lower_limit_muon1: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
```

```
        ip_lut_muon1: std_logic_vector(2**(D_S_I_MUON_V2.ip_high-D_S_I_MUON_V2.
            ip_low+1)-1 downto 0);

        muon2_object_low: natural;
        muon2_object_high: natural;
        pt_ge_mode_muon2: boolean;
        pt_threshold_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto 0);
        nr_eta_windows_muon2: natural;
        eta_w1_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w1_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w2_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w2_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w3_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w4_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        eta_w5_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_full_range_muon2: boolean;
        phi_w1_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_muon2: boolean;
        phi_w2_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
            downto 0);
        requested_charge_muon2: string(1 to 3);
        qual_lut_muon2: std_logic_vector(2**(D_S_I_MUON_V2.qual_high-
            D_S_I_MUON_V2.qual_low+1)-1 downto 0);
        iso_lut_muon2: std_logic_vector(2**(D_S_I_MUON_V2.iso_high-D_S_I_MUON_V2.
            iso_low+1)-1 downto 0);
        upt_cut_muon2 : boolean;
        upt_upper_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
        upt_lower_limit_muon2: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
            0);
        ip_lut_muon2: std_logic_vector(2**(D_S_I_MUON_V2.ip_high-D_S_I_MUON_V2.
            ip_low+1)-1 downto 0);

        requested_charge_correlation: string(1 to 2);

        diff_eta_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_eta_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
```

```
        diff_phi_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_phi_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

        dr_upper_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0);
        dr_lower_limit_vector: std_logic_vector(MAX_WIDTH_DR_LIMIT_VECTOR-1
            downto 0);

        mass_upper_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);
        mass_lower_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);

        pt_width: positive;
        upt_width: positive;
        mass_cosh_cos_precision : positive;
        cosh_cos_width: positive;

        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0);
        sin_cos_width: positive;
        pt_sq_sin_cos_precision : positive

    );
    port(
        lhc_clk: in std_logic;
        muon1_data_i: in muon_objects_array;
        muon2_data_i: in muon_objects_array;
        ls_charcorr_double: in muon_charcorr_double_array;
        os_charcorr_double: in muon_charcorr_double_array;
        diff_eta: in deta_dphi_vector_array;
        diff_phi: in deta_dphi_vector_array;
        pt1 : in diff_inputs_array;
        pt2 : in diff_inputs_array;
        upt1 : in diff_inputs_array;
        upt2 : in diff_inputs_array;
        cosh_deta : in muon_cosh_cos_vector_array;
        cos_dphi : in muon_cosh_cos_vector_array;
        cos_phi_1_integer : in sin_cos_integer_array;
        cos_phi_2_integer : in sin_cos_integer_array;
        sin_phi_1_integer : in sin_cos_integer_array;
        sin_phi_2_integer : in sin_cos_integer_array;
        condition_o: out std_logic
    );
end muon_muon_correlation_condition;
```

Table 25: Explanation of Listing 10

Item	Explanation
`same_bx`	boolean indicating whether data are from same Bx - 'true' for same Bx.
`deta_cut`	boolean for using $\Delta\eta$ cut.
`dphi_cut`	boolean for using $\Delta\varphi$ cut.

Table 25: Explanation of Listing 10

Item	Explanation
`dr_cut`	boolean for using DR cut.
`mass_cut`	boolean for using MASS cut.
`mass_type`	selection of mass type (INVARIANT_MASS_TYPE, INVARIANT_MASS_PT_TYPE, TRANSVERSE_MASS_TYPE or TRANSVERSE_MASS_PT_TYPE are allowed).
`muon_object_low`	low index of object range (valid numbers: 0..7).
`muon_object_high`	high index of object range (valid numbers: 0..7, but greater or equal muon_object_low).
`pt_ge_mode_muon1`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_threshold_muon1`	threshold value for comparison in p_{T}.
`nr_eta_windows_muon1`	integer value for number of η cuts.
`eta_w1_upper_limit_muon1`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_muon1`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_muon1`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_muon1`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_muon1`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_muon1`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_muon1`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_muon1`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_muon1`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_muon1`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_muon1`	boolean to set full range of φ.
`phi_w1_upper_limit_muon1`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_muon1`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_muon1`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_muon1`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_muon1`	"lower limits" of "window"-comparator 2 for φ.
`requested_charge_muon1`	string for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
`qual_lut_muon1`	content of LUT (16 bits) for quality comparison.
`iso_lut_muon1`	content of LUT (4 bits) for isolation comparison.
`upt_cut_muon1`	boolean for using unconstrained p_{T} cuts.
`upt_upper_limit_muon1`	"upper limit" of unconstrained p_{T}.
`upt_lower_limit_muon1`	"lower limit" of unconstrained p_{T}.
`ip_lut_muon1`	content of LUTs (4 bits) for impact parameter.

Table 25: Explanation of Listing 10

Item	Explanation
`pt_ge_mode_muon2`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_threshold_muon2`	threshold value for comparison in p_{T}.
`nr_eta_windows_muon2`	integer value for number of η cuts.
`eta_w1_upper_limit_muon2`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_muon2`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_muon2`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_muon2`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_muon2`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_muon2`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_muon2`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_muon2`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_muon2`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_muon2`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_muon2`	boolean to set full range of φ.
`phi_w1_upper_limit_muon2`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_muon2`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_muon2`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_muon2`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_muon2`	"lower limits" of "window"-comparator 2 for φ.
`requested_charge_muon2`	string for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
`qual_lut_muon2`	content of LUT (16 bits) for quality comparison.
`iso_lut_muon2`	content of LUT (4 bits) for isolation comparison.
`upt_cut_muon2`	boolean for using unconstrained p_{T} cuts.
`upt_upper_limit_muon2`	"upper limit" of unconstrained p_{T}.
`upt_lower_limit_muon2`	"lower limit" of unconstrained p_{T}.
`ip_lut_muon2`	content of LUTs (4 bits) for impact parameter.
`requested_charge_correlation`	string (2 characters) for requested charge correlation ("ls" means "like sign", "os" means "opposite sign" or "ig" means "ignore").
`diff_eta_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_eta_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in η (hex value).
`diff_phi_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in φ (hex value).
`diff_phi_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in φ (hex value).

Table 25: Explanation of Listing 10

Item	Explanation
`dr_upper_limit`	"upper limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`dr_lower_limit`	"lower limit" of "window"-comparator for comparison of ΔR^2 (hex value).
`DETA_DPHI_VECTOR_WIDTH`	vector width of $\Delta\eta$ and $\Delta\varphi$ for calculation of ΔR^2.
`DETA_DPHI_PRECISION`	position after decimal point for $\Delta\eta$ and $\Delta\varphi$.
`mass_upper_limit`	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`mass_lower_limit`	"lower limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`MASS_PRECISION`	position after decimal point for $\frac{M^2}{2}$.
`pt_width`	number of bits of pt.
`MASS_COSH_COS_PRECISION`	position after decimal point for cosh($\Delta\eta$) and cos($\Delta\varphi$).
`cosh_cos_width`	number of bits for the maximum value in the LUT for cosh($\Delta\eta$).
`pt_sq_threshold`	threshold value for comparison in two-body pt (pt^2).
`sin_cos_width`	number of bits for the maximum value in the LUT for cos(φ) and sin(φ).
`PT_PRECISION`	position after decimal point for pt^2.
`PT_SQ_SIN_COS_PRECISION`	position after decimal point for cos(φ) and sin(φ).
`lhc_clk`	clock input (LHC clock).
`muon1_data_i`	muon1 input data.
`muon2_data_i`	muon2 input data.
`ls_charcorr_double`	array of "like sign" charge correlation for double condition.
`os_charcorr_double`	array of "opposite sign" charge correlation for double condition.
`diff_eta`	differences in η, calculated in an instance of module `sub_eta_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`), see 4.4.5.
`diff_phi`	differences in φ, calculated in an instance of module `sub_phi_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`).
`pt1`	muon1 p_{T} values [from LUT, in $GeV\times10$].[12]
`pt2`	muon2 p_{T} values [from LUT, in $GeV\times10$].
`upt1`	muon1 unconstrained p_{T} values [from LUT, in $GeV\times10$].[13]
`upt2`	muon2 unconstrained p_{T} values [from LUT, in $GeV\times10$].
`cosh_deta`	cosh($\Delta\eta$) values [from LUT, cosh($\Delta\eta$)×10000].[14]
`cos_dphi`	cos($\Delta\varphi$) values [from LUT, cosh($\Delta\varphi$)×10000].
`cos_phi_1`	cos(φ) values from LUT for muon.
`cos_phi_2`	cos(φ) values from LUT for muon (different to cos_phi_1, when data from different bunch-crossings).

---

[12] value 10 from $10^{\mathrm{MUON\_MUON\_INV\_MASS\_PT\_PRECISION}}$

[13] value 10 from $10^{\mathrm{MUON\_MUON\_INV\_MASS\_PT\_PRECISION}}$

[14] value 10000 from $10^{\mathrm{MUON\_MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$

Table 25: Explanation of Listing 10

Item	Explanation
`sin_phi_1`	sin(φ) values from LUT for muon.
`sin_phi_2`	sin(φ) values from LUT for muon (different to sin_phi_1, when data from different bunch-crossings).
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

4.4.12.2.8 Muon Muon Correlation condition module for Invariant Mass Divided by ΔR

The muon muon correlation condition module for invariant mass divided by ΔR contains two "Single object requirement conditions" from different bunch-crossings as one possible mode and a "Double objects requirement condition" for objects of same bunch-crossing as a second mode (selection is done by a parameter in the generic list of `muon_muon_mass_div_dr_condition.vhd` named "same_bx").
The calculation of *invariant mass divided by* ΔR *of two objects* is done in an own module outside of the condition (`mass_div_dr_calculator.vhd`), see 4.4.12.1.5. This module is instantiated once for every object type bunch-crossing combination.
The comparison of calculated values and threshold is done inside the module (texttttmuon_muon_mass_div_dr_condition.vhd).
In the TME there is one threshold for M/ΔR: "greater/equal threshold", given in GeV (floating point notation).
The threshold for comparison with $\frac{M^2}{2} \times (1/\Delta R^2)$ (no square root in VHDL) is provided by VHDL-Producer.
No other correlation cuts, except "charge correlation", available in this condition type.

4.4.12.2.9 Muon Correlation condition module for Invariant Mass with Three Objects

The muon correlation condition module for invariant mass with three objects (`muon_mass_3_obj_condition.vhd`) contains a "Triple objects requirement condition" for objects of same type and same bunch-crossing.
In addition a "Cut" for calculation of *invariant mass with three objects* (see 4.4.12.1.6) is mandatory.
No other correlation cuts, except "charge correlation", available in this condition type.

4.4.12.2.10 Muon Esums Correlation condition module

The muon esums correlation condition module (`muon_esums_correlation_condition.vhd`) contains two "Single object requirement conditions", one of muon objects and one of esums (ET_{miss}, ET^{HF}_{miss} or HT_{miss}).
In addition there are "Cuts" for differences in φ ($\Delta\varphi$) or a calculation of mass (MASS) for Transverse mass or Transverse mass with two-body pt.

The differences in φ are calculated in bins. These differences in bins are converted to numbers (by LUTs, e.g. MUON_ETM_DIFF_PHI_LUT, ...), which represents values of differences (multiples of units in φ). These values given in the LUTs are calculated as floating-point values (based on the scales of φ), which are multiplied by a factor and truncated to an integer value. So, in the LUTs we have integer values, the factor is $10^{\text{precision}}$.
Because of the different scales of muon objects and esums in φ, there are LUTs for conversion the esums bins to muon bins (in `gtl_pkg.vhd`: e.g. ETM_PHI_CONV_2_MUON_PHI_-LUT).

Remark:
The center value of bins are used as reference value for conversion. The content of LUT is calculated with formular:
"converted-esums-phi[bin] = esums-phi[bin] $\times$ 4 + 2"
(see Figure 14). The conversion calculations are preliminary, others may be proposed.

Definitions of scales:

- ET_{miss}, ET^{HF}_{miss} or HT_{miss}:
  - ϕ bin width = $\frac{2\pi}{144}$ (bin 0 from 0.0 to $\frac{2\pi}{144}$)
- Muon objects:
  - ϕ bin width = $\frac{2\pi}{576}$ (bin 0 from 0.0 to $\frac{2\pi}{576}$)

The contents of the LUTs for cos($\Delta\varphi$) (MU_ETM_COS_DPHI_LUT, ...) for Transverse mass (formular see 4.4.12.1.3) are created by calculating cosine, rounding-up at the 4th position after decimal point and multiplying by 10000 ($10^{\text{MU\_ETM\_COSH\_COS\_PRECISION}}$) to get integer values.[15]
The contents of the LUTs for cos(φ) (CALO_COS_PHI_LUT and MUON_COS_PHI_LUT) and sin(φ) (CALO_SIN_PHI_LUT and MUON_SIN_PHI_LUT) for two-body pt (formular see 4.4.12.1.4) are created by calculating cosine and sine, rounding-up at the 3rd position after decimal point and multiplying by 1000 to get integer values.
The condition is complied, if at least one comparison between object parameters and requirements is valid for the both "Single object requirement condition" and the results of selected "Cuts" are inside of a range (upper and lower limit). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for $\Delta\varphi$ are expressed with a precision of 3rd position after decimal point, for MASS with 1st position after decimal point.

For VHDL entity declaration for muon esums correlation condition module in `muon_esums_correlation_condition.vhd`, see Listing 11.

Listing 11: Entity declaration of `muon_esums_correlation_condition.vhd`

```
entity muon_esums_correlation_condition is
```

---

[15] Definition of "`constant MU_ETM_COSH_COS_PRECISION` ..." and "`constant CALO_PHI_CONV_2_MUON_PHI_LUT` ..." in file `gtl_pkg.vhd`.

```
generic(

    dphi_cut: boolean;
    mass_cut: boolean;
    mass_type : natural;
    twobody_pt_cut: boolean;

    muon_object_low: natural;
    muon_object_high: natural;
    pt_ge_mode_muon: boolean;
    pt_threshold_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto 0);
    nr_eta_windows_muon : natural;
    eta_w1_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w1_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w2_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w2_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w3_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w3_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w4_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w4_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w5_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    eta_w5_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    phi_full_range_muon : boolean;
    phi_w1_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    phi_w1_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    phi_w2_ignore_muon : boolean;
    phi_w2_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    phi_w2_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1
        downto 0);
    requested_charge_muon: string(1 to 3);
    qual_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.qual_high-D_S_I_MUON_V2
        .qual_low+1)-1 downto 0);
    iso_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.iso_high-D_S_I_MUON_V2.
        iso_low+1)-1 downto 0);
    upt_cut_muon : boolean;
    upt_upper_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
        0);
    upt_lower_limit_muon: std_logic_vector(MAX_MUON_TEMPLATES_BITS-1 downto
        0);
    ip_lut_muon: std_logic_vector(2**(D_S_I_MUON_V2.ip_high-D_S_I_MUON_V2.
        ip_low+1)-1 downto 0);

    et_ge_mode_esums: boolean;
    obj_type_esums: natural := ETM_TYPE;
```

```
        et_threshold_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_full_range_esums: boolean;
        phi_w1_upper_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w1_lower_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w2_ignore_esums: boolean;
        phi_w2_upper_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);
        phi_w2_lower_limit_esums: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1
            downto 0);

        diff_phi_upper_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);
        diff_phi_lower_limit_vector: std_logic_vector(
            MAX_WIDTH_DETA_DPHI_LIMIT_VECTOR-1 downto 0);

        mass_upper_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);
        mass_lower_limit_vector: std_logic_vector(MAX_WIDTH_MASS_LIMIT_VECTOR-1
            downto 0);

        pt1_width: positive;
        pt2_width: positive;
        mass_cosh_cos_precision : positive;
        cosh_cos_width: positive;

        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0);
        sin_cos_width: positive;
        pt_sq_sin_cos_precision : positive

    );
    port(
        lhc_clk: in std_logic;
        muon_data_i: in muon_objects_array;
        esums_data_i: in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        diff_phi: in deta_dphi_vector_array;
        pt1 : in diff_inputs_array;
        pt2 : in diff_inputs_array;
        cos_dphi : in calo_muon_cosh_cos_vector_array;
        cos_phi_1_integer : in sin_cos_integer_array;
        cos_phi_2_integer : in sin_cos_integer_array;
        sin_phi_1_integer : in sin_cos_integer_array;
        sin_phi_2_integer : in sin_cos_integer_array;
        condition_o: out std_logic
    );
end muon_esums_correlation_condition;
```

Table 26: Explanation of Listing 10

Item	Explanation
`dphi_cut`	boolean for using $\Delta\varphi$ cut.
`mass_cut`	boolean for using MASS cut.
`mass_type`	selection of mass type (TRANSVERSE_MASS_TYPE or TRANSVERSE_MASS_PT_TYPE are allowed).
`muon_object_low`	low index of object range (valid numbers: 0..7).
`muon_object_high`	high index of object range (valid numbers: 0..7, but greater or equal muon_object_low).
`pt_ge_mode_muon`	'mode-selection' for the p_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`pt_threshold_muon`	threshold value for comparison in p_{T}.
`nr_eta_windows_muon`	integer value for number of η cuts.
`eta_w1_upper_limit_muon`	"upper limit" of "window"-comparator 1 for η.
`eta_w1_lower_limit_muon`	"lower limit" of "window"-comparator 1 for η.
`eta_w2_upper_limit_muon`	"upper limit" of "window"-comparator 2 for η.
`eta_w2_lower_limit_muon`	"lower limit" of "window"-comparator 2 for η.
`eta_w3_upper_limit_muon`	"upper limit" of "window"-comparator 3 for η.
`eta_w3_lower_limit_muon`	"lower limit" of "window"-comparator 3 for η.
`eta_w4_upper_limit_muon`	"upper limit" of "window"-comparator 4 for η.
`eta_w4_lower_limit_muon`	"lower limit" of "window"-comparator 4 for η.
`eta_w5_upper_limit_muon`	"upper limit" of "window"-comparator 5 for η.
`eta_w5_lower_limit_muon`	"lower limit" of "window"-comparator 5 for η.
`phi_full_range_muon`	boolean to set full range of φ.
`phi_w1_upper_limit_muon`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_muon`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_muon`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_muon`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_muon`	"lower limits" of "window"-comparator 2 for φ.
`requested_charge_muon`	string for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
`qual_lut_muon`	content of LUT (16 bits) for quality comparison.
`iso_lut_muon`	content of LUT (4 bits) for isolation comparison.
`upt_cut_muon`	boolean for using unconstrained p_{T} cuts.
`upt_upper_limit_muon`	"upper limit" of unconstrained p_{T}.
`upt_lower_limit_muon`	"lower limit" of unconstrained p_{T}.
`ip_lut_muon`	content of LUTs (4 bits) for impact parameter.

Table 26: Explanation of Listing 10

Item	Explanation
`et_ge_mode_esums`	'mode-selection' for the E_{T} comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
`obj_type_esums`	selection of esums type (ETM_TYPE or HTM_TYPE are allowed)
`et_threshold_esums`	threshold value for comparison in E_{T}.
`phi_full_range_esums`	boolean to set full range of φ.
`phi_w1_upper_limit_esums`	"upper limit" of "window"-comparator 1 for φ.
`phi_w1_lower_limit_esums`	"lower limit" of "window"-comparator 1 for φ.
`phi_w2_ignore_esums`	boolean to ignore "window"-comparator 2 for φ.
`phi_w2_upper_limit_esums`	"upper limit" of "window"-comparator 2 for φ.
`phi_w2_lower_limit_esums`	"lower limits" of "window"-comparator 2 for φ.
`diff_phi_upper_limit`	"upper limit" of "window"-comparator for comparison of differences in φ (hex value).
`diff_phi_lower_limit`	"lower limit" of "window"-comparator for comparison of differences in φ (hex value).
`DETA_DPHI_VECTOR_WIDTH`	vector width of $\Delta\varphi$.
`DETA_DPHI_PRECISION`	position after decimal point for $\Delta\varphi$.
`mass_upper_limit`	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`mass_lower_limit`	"lower limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).
`MASS_PRECISION`	position after decimal point for $\frac{M^2}{2}$.
`pt1_width`	number of bits of pt1.
`pt2_width`	number of bits of pt2.
`MASS_COSH_COS_PRECISION`	position after decimal point for $\cos(\Delta\varphi)$.
`cosh_cos_width`	number of bits for the maximum value in the LUT for $\cos(\Delta\varphi)$.
`pt_sq_threshold`	threshold value for comparison in two-body pt (pt^2).
`sin_cos_width_1`	number of bits for the maximum value in the LUT for $\cos(\varphi)$ and $\sin(\varphi)$ of muon.
`sin_cos_width_2`	number of bits for the maximum value in the LUT for $\cos(\varphi)$ and $\sin(\varphi)$ of esums.
`PT_PRECISION`	position after decimal point for pt^2.
`PT_SQ_SIN_COS_PRECISION`	position after decimal point for $\cos(\varphi)$ and $\sin(\varphi)$.
`lhc_clk`	clock input (LHC clock).
`muon_data_i`	muon input data.
`esums_data_i`	esums input data, structure defined with `obj_type_esums`.
`diff_phi`	differences in φ, calculated in an instance of module `sub_phi_integer_obj_vs_obj.vhd` in top-of-hierarchy module (`gtl_module.vhd`).
`pt1`	muon E_{T} values [from LUT, in $GeV \times 10$].
`pt2`	esums E_{T} values [from LUT, in $GeV \times 10$].

Table 26: Explanation of Listing 10

Item	Explanation
`cos_dphi`	cos($\Delta\varphi$) values from LUT.
`cos_phi_1`	cos(φ) values from LUT for muon.
`cos_phi_2`	cos(φ) values from LUT for esums.
`sin_phi_1`	sin(φ) values from LUT for muon.
`sin_phi_2`	sin(φ) values from LUT for esums.
`condition_o`	output of condition (routed to Algorithms logic, see 4.4.14).

4.4.13 External Conditions

Maximal 256 External Conditions are possible in Global Trigger. They are provided as inputs in the Algorithms logic of μGTL. External Conditions will include the "Technical Trigger" of the legacy system.

4.4.14 Algorithms logic

The outputs of all the instantiated conditions are combined in the Algorithms logic with boolean algebra given by TME for every single Algorithm. These Algorithms are registered and provided as inputs for Final Decision Logic.

5 Final Desicion Logic

The Final Desicion Logic (μFDL) firmware contains algo-bx-masks, suppression of algos caused by calibration trigger, prescalers, veto-masks and rate-counters ("before prescalers", "after prescalers" and "post dead time") for each Algorithm and the local Final-OR- and veto-logic.

5.1 μFDL Interface

Inputs:

- Algorithms from μGTL
- IPBus interface (for registers, counters and memories)
- LHC-clock
- Reset signal
- BC0, BGo test-enable, L1A

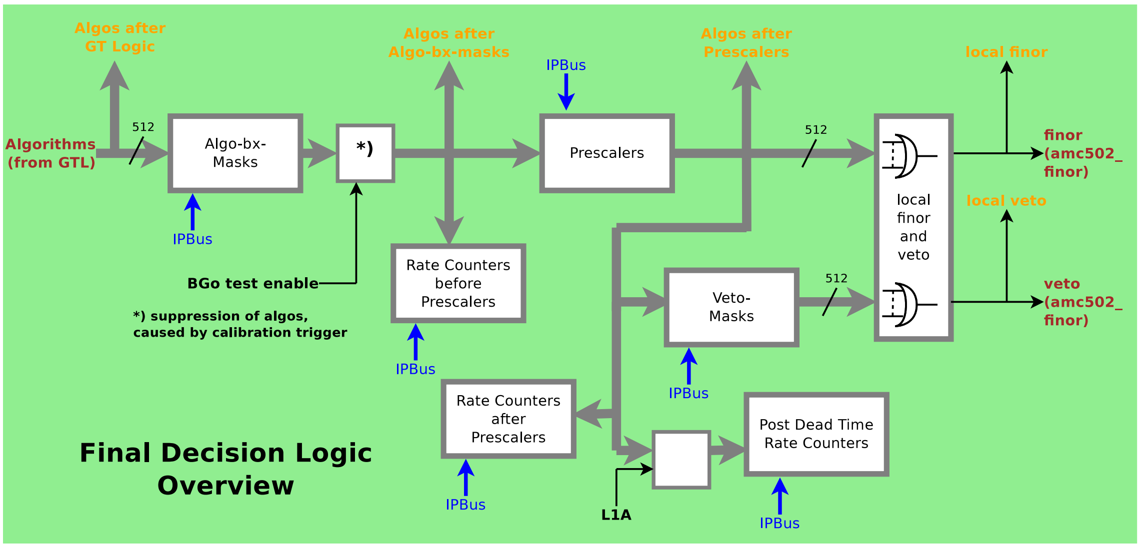


Figure 15: μFDL firmware v1.0.1

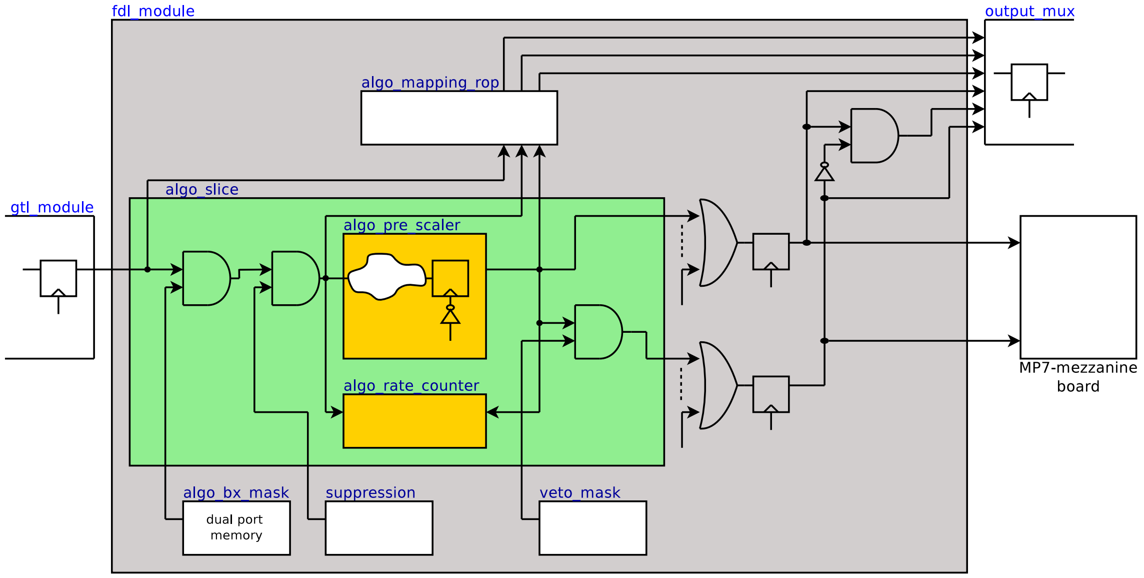
- Begin of lumi-section

Outputs:

- Prescale factor set index to Readout-Process
- Algorithms after GTLogic to Readout-Process
- Algorithms after algo-bx-masks to Readout-Process
- Algorithms after prescalers to Readout-Process
- Algorithms after Final-OR-masks to Readout-Process
- Local Final-OR to Readout-Process
- Local veto to Readout-Process
- Local Final-OR with veto to Readout-Process
- Local Final-OR to mezzanine
- Local veto to mezzanine
- Local Final-OR with veto to mezzanine

5.2 MP7 Final-OR hardware solution

The firmware of μFDL in this document is based on a hardware configuration with maximum 6 μGT modules.



Scheme of the FDL pipeline structure
(for one algo slice)

Figure 16: μFDL pipeline v1.0.1

5.3 Data flow

Every Algorithm, in total 512 coming from μGTL, passes a algo-bx-mask, the logic for suppression of algos caused by calibration trigger and a prescaler, which reduces the trigger rate by a given factor. Prescaled Algorithms signals are combined to a local final-or-signal (Final-OR). For every Algorithm there is a rate-counter before prescaler and after prescaler, which are incremented by LHC-clock if the Algorithm is true. In addition there are post-dead-time counters, one for each Algorithm, which are incremented, if the Algorithm and the L1A-signal are true at the same bunch-crossing. Algorithms after GTLogic, after algo-bx-masks, after prescalers, the local Final-OR- and local veto-signal are provided for read-out-record.
If there are not enough firmware resources in one μGT board, more boards could be used. Therefore the 512 Algorithms are partitioned by TME. TME will set the number of Algorithms as constant in the package module `gtl_pkg.vhd`. This means μGTL and μFDL firmware considered as a unit for synthesis. In the case of more μGT boards, the local Final-OR and local veto are routed via a mezzanine board on MP7 (located on "General Purpose I/O connector") to the FINOR-AMC502 module, where the total Final-OR is created and send to TCDS.
A mapping for Algorithms is provided, to give flexibility for setting the index of Algorithms:

- creating a mapping instance (algo_mapping_rop.vhd) over TME (see **??**), this component will be instantiated in fix part of FDL, and new calculation will done each time over TME.
- TME delivers just the number of Algorithms, which will be built on each card.
- from FDL point of view, FDL see incremented number of Algorithms indexes, e.g. 0, 1, 2, which is e.g. 69, 200, 300.

- TME should take care of assignment of each Algorithm to a number, that means if in card 1 algo_59 is defined, nobody allows to produce the same number again.

5.4 Main parts

The top-of-hierarchy module (`fdl_module.vhd`) contains

- version registers
- a command pulse register
- prescalers for all Algorithms
- registers for prescale factors
- register for prescale factor set index
- rate-counters for all Algorithms, finor, veto, L1A and post-dead-time
- read only registers for rate-counter values
- algo-bx-masks for all Algorithms
- Final-OR-masks for all Algorithms
- veto-masks for all Algorithms
- the Final-OR-logic

5.4.1 Registers and memories

All registers and memories are 32 bits wide. (A first draft of the definition of the relative addresses is shown in Table 27.)

- Dual-port memories for the algo-bx-masks are implemented. For each Algorithm there is a mask bit at every bunch crossing of one orbit. Therefore in total memories of 4096 x 512 bits are implemented. Because of the 32 bit data interface, 16 memories each with a size of 4096 x 32 bits are instantiated.
- Read-only registers for the value of rate-counters (before and after prescalers, post-dead-time counters) are implemented, 512 registers, one for every Algorithm. Rate-counter value has 32 bits.
- Registers for prescale factor of the prescalers are implemented, 512 registers, one for every Algorithm. A prescale factor value has 24 bits.
- Registers for masks (finor- and veto-masks) are implemented, 512 registers.
- One register for prescale factors set index is implemented. This register contains a value, which is unique for a given set of prescale factors. The content of this register is part of Readout-record.

- One register for command pulses is implemented. One bit of this register (bit 0) is used for "setting the request signal for updating prescale factors high", which enables, that the prescale factors and the prescale factor set index are loaded at the begin of a luminosity segment period. (Other bits are not defined yet.)
- One control register is implemented (the content has to be defined).
- 32 register for L1 Trigger Menu name for μGTL is implemented.
- 4 register for L1 Trigger Menu UUID for μGTL is implemented.
- One register for L1 Trigger Menu compiler version is implemented.
- One register for μFDL firmware version is implemented.
- One register for μGTL firmware (fixed code) version is implemented.

5.4.1.1 Register map

The register map for μFDL has a base address of `0x90000000`.

Table 27: μFDL register map

Offset	Register name	Access	Description
0x90000000	`Algo BX masks(0)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 0-31.
0x90001000	`Algo BX masks(1)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 32-63.
...	...	...	...
0x9000F000	`Algo BX masks(15)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 480-511.
0x90010000	`Rate counter before prescaler`	r	512 read-only registers for rate-counter values before prescalers.
0x90010200	`Prescale factors`	r/w	512 registers for prescale factors.
0x90010400	`Rate counter after prescaler`	r	512 read-only registers for rate-counter values after prescalers.
0x90010600	`Rate counter post-dead-time`	r	512 read-only registers for post-dead-time rate-counter values.
0x90010800	`Masks`	r/w	512 registers for finor-masks and veto-masks. Bit 0 = finor-mask, bit 1 = veto-mask.

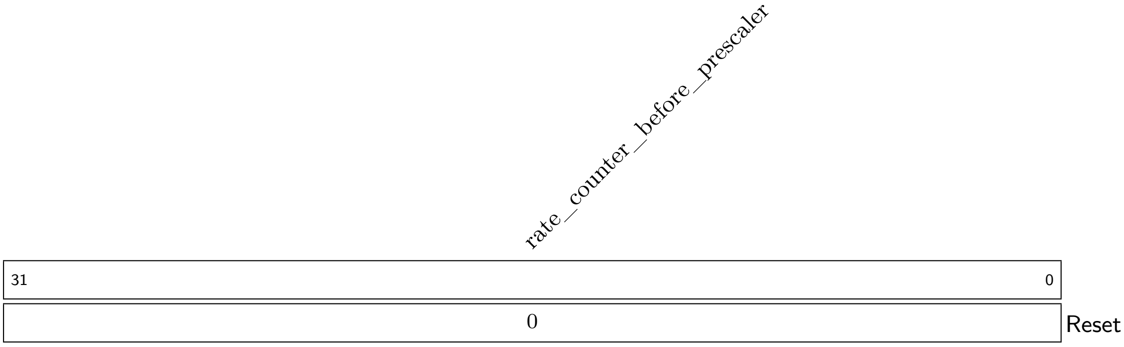
Table 27: μFDL register map

Offset	Register name	Access	Description
0x90091880	`Prescale factors set index`	r/w	Register for prescale factors set index.
0x900918C0	`L1tm name`	r	32 registers for L1 Trigger Menu name for μGTL.
0x900918E0	`L1tm uuid`	r	4 registers for L1 Trigger Menu UUID for μGTL.
0x900918E4	`L1tm compiler version`	r	Register for L1 Trigger Menu compiler version.
0x900918E5	`GTL FW version`	r	Register for firmware version of μGTL VHDL code.
0x900918E6	`FDL FW version`	r	Register for firmware version of μFDL VHDL code.
0x900918E7	`L1tm FW uuid`	r	4 registers for L1 Trigger Menu FW UUID for μGTL.
0x900918EB	`SVN revision number`	r	Register for SVN revision number.
0x900918EC	`L1tm uuid hash`	r	Register for L1 Trigger Menu UUID hash for μGTL.
0x900918ED	`L1tm FW uuid hash`	r	Register for L1 Trigger Menu FW UUID hash for μGTL.
0x900918EE	`Module ID`	r	Register for Module ID of L1 Trigger Menu.
0x90091900	`Command Pulses`	r/w	Register for command pulses.
0x90091980	`Rate counter finor`	r	One read-only registers for finor rate-counter value.
0x90092200	`L1A latency delay`	r/w	Register for L1A latency delay value (used for post-dead-time counter).
0x90093000	`Rate counter L1A`	r	One read-only registers for L1A rate-counter value.
0x90094000	`Rate counter veto`	r	One read-only registers for veto rate-counter value.
0x90095000	`Current prescale set index`	r	Read-only register for prescale factors set index, which was "updated" with begin of current lumi-section ("prescale_factors_set_index_reg_updated(0)" in VHDL).

Table 27: μFDL register map

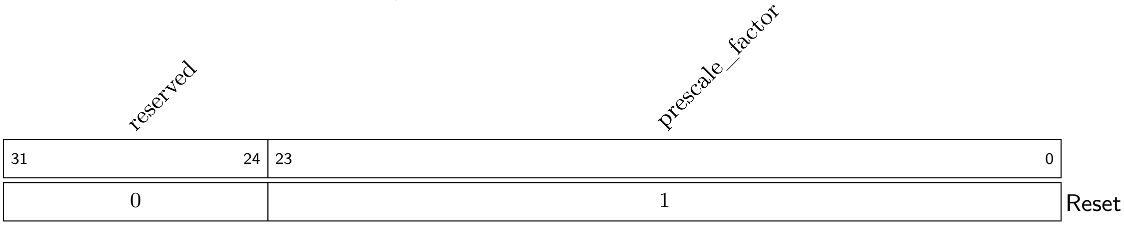
Offset	Register name	Access	Description
0x90095001	`Previous prescale set index`	r	Read-only register for prescale factors set index, which was "updated" with begin of previous lumi-section for monitoring "prescale_factors_set_index_reg_updated(1)" in VHDL).
0x90096000	`Calibration trigger gap`	r/w	Register for begin and end (in Bx) of calibration trigger gap.

Register 5.1: Rate counter before prescaler



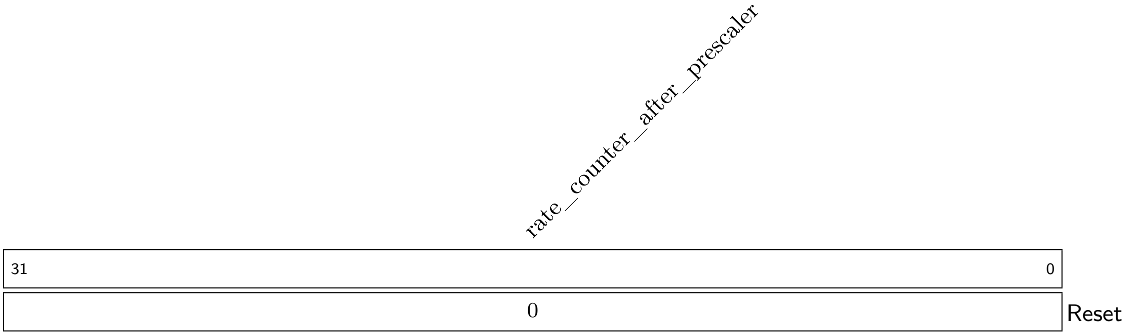
rate_counter_before_prescaler Rate counter before prescaler. Counts the occurancy of an algo (given by register address) in one luminosity segment.

Register 5.2: Prescale factor



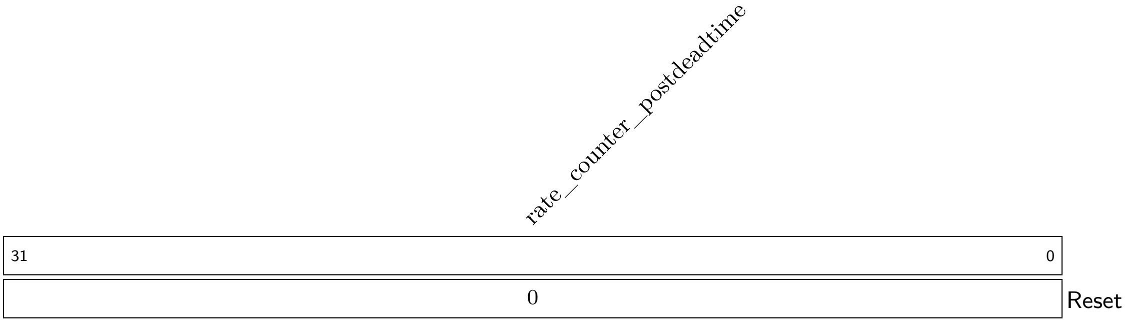
prescale_factor Prescale factor of an algo (given by register address). Prescale factor = 0 means disable algo.

Register 5.3: Rate counter after prescaler



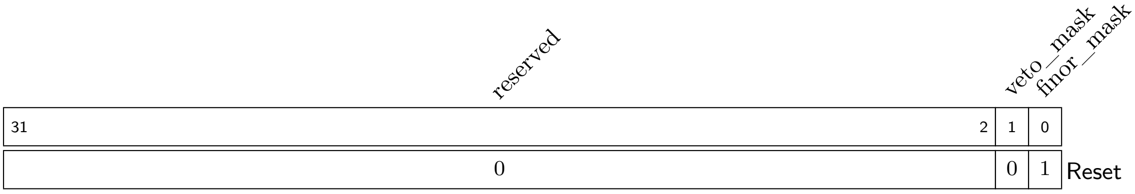
rate_counter_after_prescaler Rate counter after prescaler. Counts the occurancy of an algo (given by register address) in one luminosity segment.

Register 5.4: Rate counter post-dead-time



rate_counter_postdeadtime Rate counter post-dead-time. Counts the occurancy of an algo (given by register address) and L1A at the same bx in one luminosity segment.

Register 5.5: Masks



veto_mask Selection of a veto (by an algo, given by register address) for veto-or.

finor_mask Selection of an algo (given by register address) for final-or.

Register 5.6: Prescale factors set index

reserved	prescale_factor_set_index	
31 8	7 0	
0	0	Reset

prescale_factor_set_index Index for a certain set of prescale factors.

Register 5.7: L1tm compiler version

reserved	major	minor	revision	
31 24	23 16	15 8	7 0	
0	0	0	0	Reset

major Major version of L1tm compiler.

minor Minor version of L1tm compiler.

revision Revision version of L1tm compiler.

Register 5.8: GTL FW version

reserved	major	minor	revision	
31 24	23 16	15 8	7 0	
0	0	0	0	Reset

major Major version of GTL firmware.

minor Minor version of GTL firmware.

revision Revision version of GTL firmware.

Register 5.9: FDL FW version

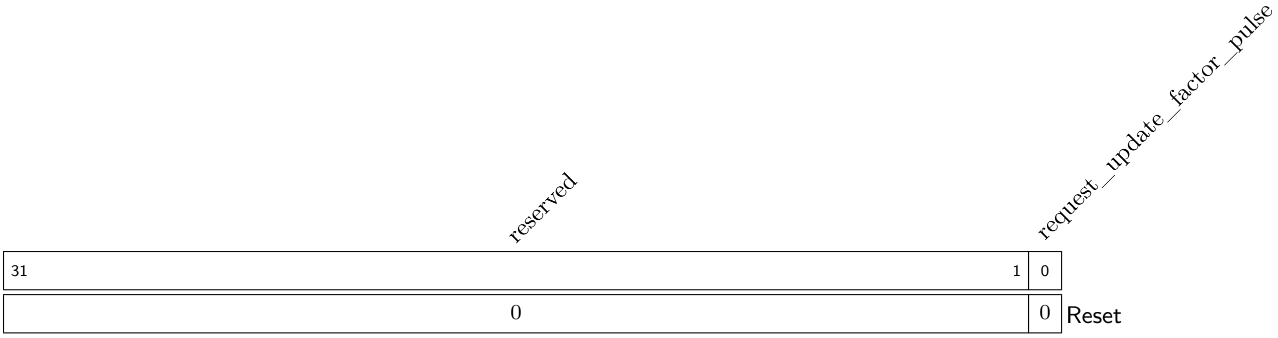
reserved	major	minor	revision	
31 24	23 16	15 8	7 0	
0	0	0	0	Reset

major Major version of FDL firmware.

minor Minor version of FDL firmware.

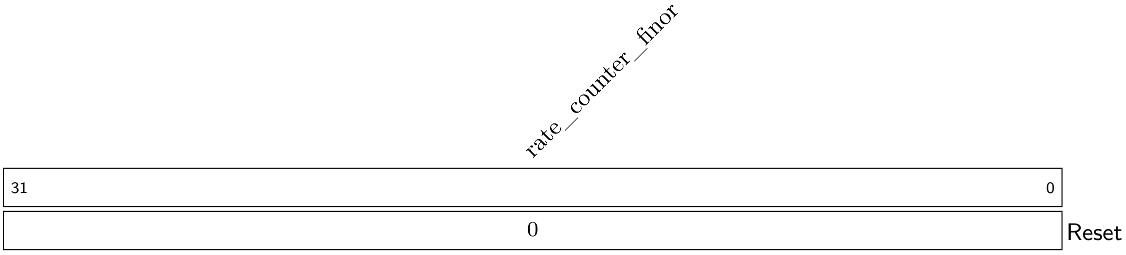
revision Revision version of FDL firmware.

Register 5.10: Command Pulses Register



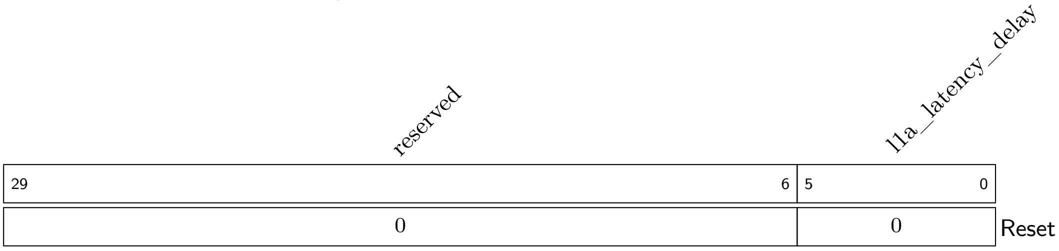
request_update_factor_pulse Sets the request signal for updating prescale factors high. Updating is done at the next "begin of luminosity segment".

Register 5.11: Rate counter finor



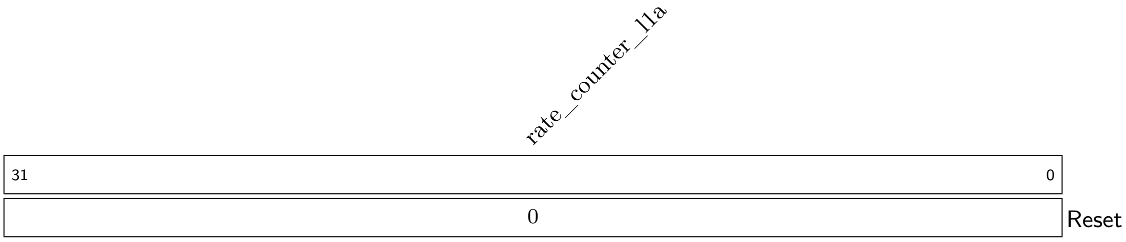
rate_counter_finor Rate counter finor. Counts the occurancy of finor in one luminosity segment.

Register 5.12: L1A latency delay



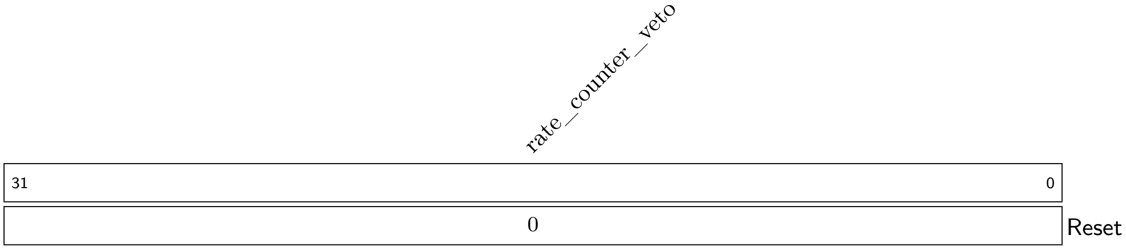
l1a_latency_delay L1A latency delay value (used for post-dead-time counter).

Register 5.13: Rate counter L1A



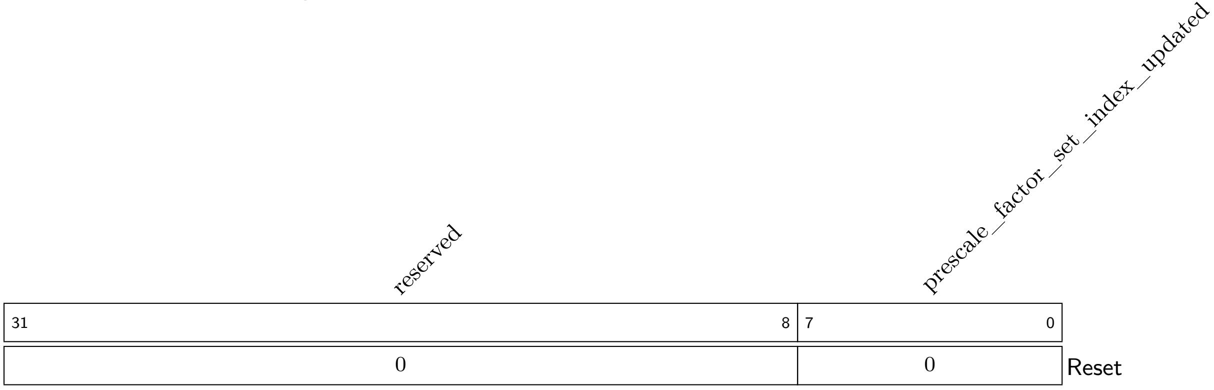
rate_counter_l1a Rate counter L1A. Counts the occurancy of L1A in one luminosity segment.

Register 5.14: Rate counter veto



rate_counter_veto Rate counter veto. Counts the occurancy of veto in one luminosity segment.

Register 5.15: Current prescale set index



prescale_factor_set_index_updated Index for a certain set of prescale factors, which was "updated" with begin of current lumi-section.

Register 5.16: Previous prescale set index

reserved	prescale_factor_set_index_updated	
31 8	7 0	
0	0	Reset

prescale_factor_set_index_updated Index for a certain set of prescale factors, which was "updated" with begin of previous lumi-section.

Register 5.17: Calibration trigger gap

reserved	end_calibration_trigger_gap	reserved	begin_calibration_trigger_gap	
31 28	27 16	15 12	11 0	
0	0	0	0	Reset

begin_calibration_trigger_gap Begin of calibration trigger gap (in Bx).

end_calibration_trigger_gap End of calibration trigger gap (in Bx).

5.4.2 Algo-bx-masks

Every Algorithm passes a logic where at every bunch-crossing of the orbit the Algorithm is enabled (or not). The algo-bx-masks are implemented as dual-port memories and loaded at the begin of run. The size of the algo-bx-masks memory is number of bunch-crossings per orbit for address length and number of Algorithms for data-depth (3564 [4096] x 512 bits). The address (bx-number) of the memory for masking the Algorithm is delivered by an address-counter for algo-bx-masks memory, which is reseted with a delay-able bcres signal, to get the correct relations between Algorithms and masks from memory.

5.4.3 Rate-counters

Every Algorithm has a rate-counters with 32 bits, because of the length of one luminosity segment period. There are counters before and after prescalers and post-dead-time counters. The counters before and after prescalers are incremented, if the Algorithm signal is in high state and a positive edge of LHC-clock occur. The post-dead-time counters are incremented, if the Algorithm signal is in high state (delayed by L1A latency delay), a L1A signal and a positive edge of LHC-clock occur. The content of a counter is updated into a register (for reading the counter value) and is reseted at the begin of a luminosity segment period. So there is one luminosity segment period time to read the registers with the counter values by software.

5.4.4 Prescalers

Every Algorithm has a prescaler with a prescale factor of 24 bits. The prescaler reduces the trigger-rate per Algorithm with a factor, so e.g. a factor of 2 passes through every second trigger. A prescale factor of 0 inhibits all triggers of the certain Algorithm. The factor is loaded into a register by software and updated at begin of a new luminosity segment period, if the update was enabled by software ('request_update_factor_pulse' was set in "command_-pulses" register). The prescaler works with the new factor. A register for "prescale factor set index" contains a value which represents a certain set of prescale factors. The content of this register is seen in the Readout-record too. The "prescale factor set index" is loaded into the register by software and updated at begin of a new luminosity segment period.

5.4.5 Finor-masks

Every Algorithm passes a Final-OR-mask, which enables the Algorithm for Final-OR. The Final-OR-masks are implemented as registers and loaded at the begin of a run.

5.4.6 Veto-masks

Every Algorithm passes a veto-mask, if at least one Algorithm, which is enabled by veto-mask, becomes high state, then Final-OR is disabled as long as the Algorithm is in high state. The veto-masks are implemented as registers and loaded at the begin of a run.

5.4.7 Finor

The Final-OR-signal is a disjunction of all Algorithms passed the Final-OR-bx-masks. An Algorithm enabled by veto-mask, disables the Final-OR. This is done on the FINOR-AMC502 module.

5.5 Implementation in firmware

The entity-declaration of `fdl_module.vhd` is shown in 5.3.

Listing 12 contains the entity-declaration of the `fdl_module.vhd`.

Listing 12: Entity declaration of `fdl_module.vhd`

```
entity fdl_module is
    generic(
        SIM_MODE : boolean := false; -- if SIM_MODE = true, "algo_bx_mask" is
            given by "algo_bx_mask_sim".
        PRESCALE_FACTOR_INIT : ipb_regs_array(0 to MAX_NR_ALGOS-1);
        MASKS_INIT : ipb_regs_array(0 to MAX_NR_ALGOS-1);
        PRESCALE_FACTOR_SET_INDEX_WIDTH : positive := 8;
        PRESCALE_FACTOR_SET_INDEX_REG_INIT : ipb_regs_array(0 to 1) := (others =>
             X"00000000");
        L1A_LATENCY_DELAY_INIT : ipb_regs_array(0 to 1) := (others => X"00000000"
            );
        CNTRL_REG_INIT : ipb_regs_array(0 to 1) := (others => X"00000000");
-- Input flip-flops for algorithms of fdl_module.vhd - used for tests of
    fdl_module.vhd only
        ALGO_INPUTS_FF: boolean := false
    );
    port(
        ipb_clk             : in std_logic;
        ipb_rst             : in std_logic;
        ipb_in              : in ipb_wbus;
        ipb_out             : out ipb_rbus;
-- ==========================================================================
        lhc_clk             : in std_logic;
        lhc_rst             : in std_logic;
        bcres               : in std_logic;
        test_en             : in std_logic;
        l1a                 : in std_logic;
        begin_lumi_section  : in std_logic;
        algo_i              : in std_logic_vector(NR_ALGOS-1 downto 0);
        bx_nr_out : out std_logic_vector(11 downto 0);
        prescale_factor_set_index_rop : out std_logic_vector(
            PRESCALE_FACTOR_SET_INDEX_WIDTH-1 downto 0);
        algo_after_gtLogic_rop   : out std_logic_vector(MAX_NR_ALGOS-1 downto 0);
        algo_after_bxomask_rop      : out std_logic_vector(MAX_NR_ALGOS-1 downto
            0);
        algo_after_prescaler_rop      : out std_logic_vector(MAX_NR_ALGOS-1
            downto 0);
        local_finor_rop     : out std_logic;
        local_veto_rop      : out std_logic;
```

```
        finor_2_mezz_lemo      : out std_logic; -- to LEMO
        finor_preview_2_mezz_lemo      : out std_logic; -- to LEMO
        veto_2_mezz_lemo      : out std_logic; -- to LEMO
        finor_w_veto_2_mezz_lemo      : out std_logic; -- to tp_mux.vhd
        local_finor_with_veto_o       : out std_logic; -- to SPY2_FINOR
-- HB 2016-03-02: v0.0.21 - algo_bx_mask_sim input for simulation use with
    MAX_NR_ALGOS (because of global index).
        algo_bx_mask_sim    : in std_logic_vector(MAX_NR_ALGOS-1 downto 0)
    );
end fdl_module;
```

6 Readout-Process

The readout is done via TX-links of MP7 to AMC13.

7 Glossary

electron/γ = electron/gamma objects over Calo-Layer2 (VHDL: eg)

jet = jet objects over Calo-Layer2 (VHDL: jet)

tau = tau objects over Calo-Layer2 (VHDL: tau)

muon = muon objects over μGMT (VHDL: muon)

ET = Scalar sum of transverse energy components over Calo-Layer2 (VHDL: ett)

ETTEM = Scalar sum of transverse energy components from ECAL only over Calo-Layer2 (VHDL: ettem)

MBTxHFy = Minimum bias HF bits (VHDL: MBT0HFP, MBT0HFM, MBT1HFP, MBT1HFM)

HT = Magnitude of the vectorial sum of transverse energy of jets (hadronic) over Calo-Layer2 (VHDL: htt)

TOWERCOUNT = tower counts (VHDL: towercount)

ET_{miss} = 2-vector sum of transverse energy over Calo-Layer2 (VHDL: etm)

HT_{miss} = Missing Total transverse energy of jets over Calo-Layer2 (VHDL: htm)

$\mathbf{ET}_{miss}^{HF}$ = 2-vector sum of transverse energy including HF over Calo-Layer2 (VHDL: etmhf)

$\mathbf{HT}_{miss}^{HF}$ = Missing Total transverse energy of jets including HF over Calo-Layer2 (VHDL: htmhf)

ASYMET = Asymmetry of ET over Calo-Layer2 (VHDL: asymet)

ASYMHT = Asymmetry of HT over Calo-Layer2 (VHDL: asymht)

ASYMETHF = Asymmetry of ET including HF over Calo-Layer2 (VHDL: asymethf)

ASYMHTHF = Asymmetry of HT including HF over Calo-Layer2 (VHDL: asymhthf)

CENTx = Centrality bits [7:0] over Calo-Layer2 (VHDL: cent7, cent6, ...)

p_{T} = transverse momentum of muon objects(VHDL: pt)

E_{T} = energy of calorimeter objects (VHDL: et)

η = pseudo-rapidity position (VHDL: eta)

φ = azimuth angle position (VHDL: phi)

isolation = isolation information (VHDL: iso)

quality = quality information (VHDL: qual)

Acronyms

DAQ Data Acquisition

DM Delay Manager Module

FDL Final Decision Logic Module

GTL Global Trigger Logic Module

ROP Readout-Process Module

TCM Timing Counter Manager Module

TCS Trigger Control System

GCT Calorimeter Trigger Layer-2

GMT Global Muon Trigger

GT Global Trigger

References