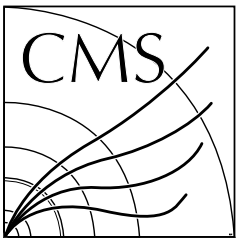


Global Trigger Logic - Description for emulator

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

Institute of High Energy Physics (HEPHY)

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

April 16, 2020

Revision History

Doc Rev	Description of Change	Revision Date
3.9	Inserted text in section 2.4.3.2.3 for Calorimeter Overlap Remover conditions and 2.4.9.2.6 for Calo Calo Overlap Remover Correlation conditions.	2020/04/16
3.8	Updated text in sections 2.4.3, 2.4.8 and 2.4.9 for changes which have been done for GTL VHDL version 1.8.0 (module names without version number, "five eta cuts").	2019/08/13
3.7	Inserted "Asymmetry" and "Centrality" of "Energy sums" (GTL VHDL version 1.6.0). Therefore updated sections 2.1, 2.2.1, 2.4.4 added section "Centrality condition" 2.4.7 and updated Table **??**	2018/08/13
3.6	Updated text in section "Global Trigger Logic" (2) according to firmware version v1.5.0 of gtl_module.vhd	2018/02/21
3.5	Inserted text for "Minimum bias trigger conditions" (2.4.5) and "Towercount condition" (2.4.6). Updated glossary.	2016/11/28
3.4	Updated text in section "Muon Muon Correlation condition module" (2.4.9.2.3).	2016/01/15
3.3	Removed "Double objects requirements condition with spatial correlation", because not used anymore in the future, replaced by Correlation conditions (see sections 2.4.3.2 and 2.4.8.3.	2016/01/08
3.2	Minor changes in text and updated Figure 3.	2016/01/08
3.1	Changed colour in Figure 4 and updated text for correlation conditions (see section 2.4.9.	2016/01/07
3.0	Updated Figures 3 and text 2.4.9.2.1.	2015/12/21
2.9	Inserted drawing of VHDL structure of cuts for correlation conditions (see Figure 5).	2015/11/18
2.8	Updated muon η ranges (Table 9) and inserted correlation conditions (see section 2.4.9). Created scheme for conversion of calorimeter η and φ to muon scale for calo-muon-correlation conditions (see Figure 6).	2015/11/17
2.7	Added text in sections (2.4.3.2.4) and (2.4.8.3.2).	2015/10/08
2.6	Updated Tables 7 and 18. Remaned section "Calorimeter conditions module - version 2" to "Calorimeter conditions module - version 3" (see 2.4.3.2), section "Muon conditions module" to "Muon conditions module - version 2" and section "Muon comparators module" to "Muon comparators module - version 2" (see 2.4.8.3)	2015/10/02
2.5	Updated text and tables of η ranges for Calorimeter objects (see 2.4.3.1).	2015/09/22

Doc Rev	Description of Change	Revision Date
2.4	Corrected calculation of muon η step width (see 2.4.8.1).	2015/09/10
2.3	Edited text in Tables 7, 17 and 18.	2015/08/28
2.2	Updated definition of η ranges for Calorimeter objects and Muon objects (2.4.3.1 and 2.4.8.1).	2015/08/20
2.1	Added section Calo Muon Correlation condition (2.4.9.2.2)	2015/08/19
2.0	Added section "Correlation conditions" (2.4.9).	2015/06/19
1.9	Added tables for calorimeter isolation-bits and for muon quality- and isolation-bits definition (5, 11 and 12). Edited section glossary (3) and acronyms.	2015/05/07
1.8	Added text for "Energy sum conditions" (2.4.4) and updated chapters for "Calorimeter conditions" for version 2. Inserted isolation bits for electron/γ and tau objects (2.4.3.1).	2015/05/06
1.7	Minor changes in sections "Muon data" (2.4.8.1)	2014/11/06
1.6	Minor changes in sections "Calorimeter conditions definition" (2.4.3.2) and "Muon conditions definition" (2.4.8.3)	2014/07/01
1.5	Minor changes	2014/06/12
1.4	Fixed bug in Figure 4	2014/04/30
1.3	Added section "Muon conditions" (2.4.8)	2014/04/22
1.2	Changed Figure 4 and minor changes in text for anti-clockwise behaviour in φ	2014/04/04
1.1	Changed text in section 2.4.3.2.	2014/02/11
1.0	Document created. Description of Calorimeter conditions.	2013/10/15

Contents

List of Figures

List of Tables

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 these 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;
```

2 Global Trigger Logic

Remark:
this description is for version 1.6.0 of Global Trigger Logic.

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

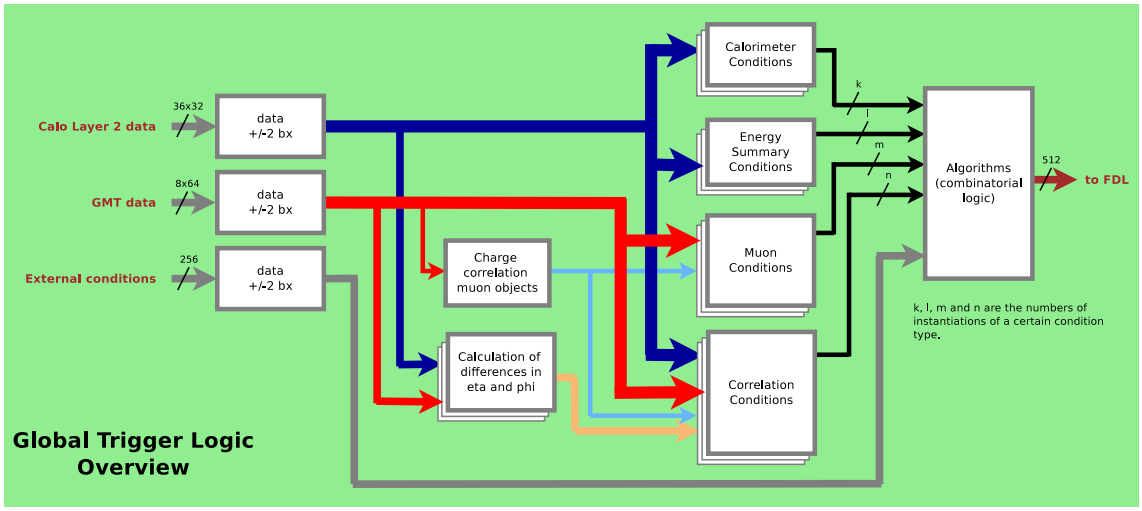


Figure 1: μGTL firmware

2.1 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^{HF}_{miss}), missing Ht objects (HT_{miss}), missing Ht including HF (HT^{HF}_{miss}) and "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
- LHC-clock

Outputs:

- Algorithms

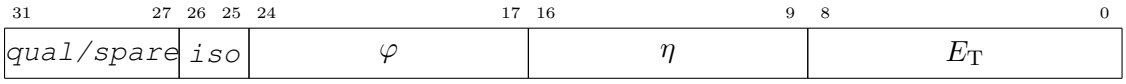
2.2 Definition of optical interfaces

Remark: all definitions in the following chapters are from a CMS Detector Note: "Scales for inputs to μGT" (see actual version in http://globaltrigger.hephy.at/upgrade/ugt/downloads/).

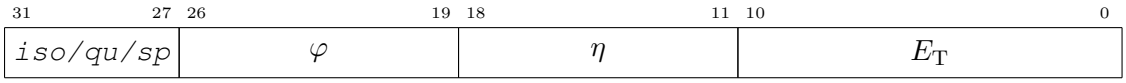
2.2.1 Calo-Layer2 optical interfaces

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

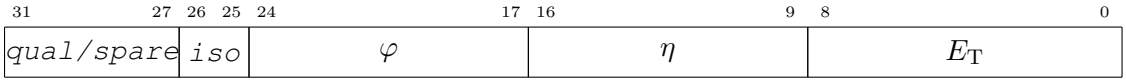
The data structure of an electron/γ object (bits 27..31 are not defined yet, reserved for quality, ...):



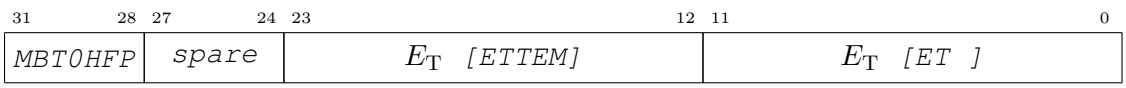
The data structure of a jet object (bits 27..31 are not defined yet, reserved for quality, ...):



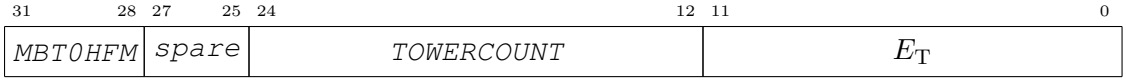
The data structure of a tau object (bits 27..31 are not defined yet, reserved for quality, ...):



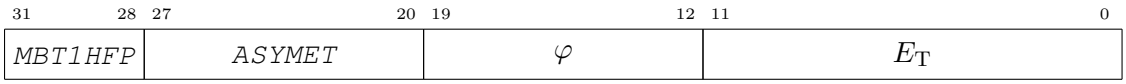
The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits]:



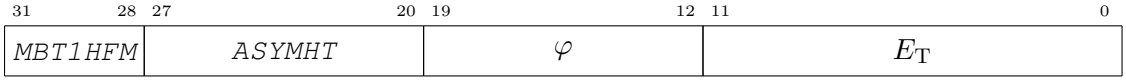
The data structure of "total calibrated Et in jets" (HT) quantity [including "towercount" and "minimum bias HF- threshold 0" bits]:



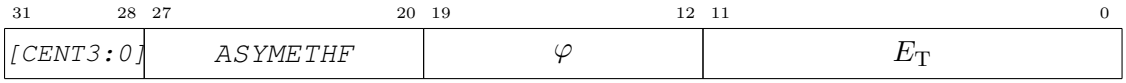
The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits]:



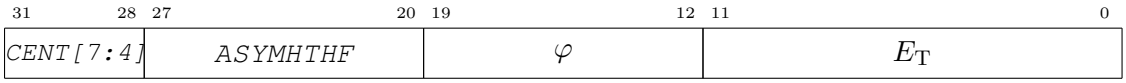
The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits]:



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



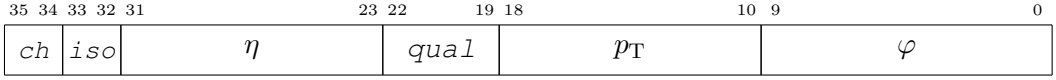
The data structure of "missing Ht including HF" (HT^{HF}_{miss}) quantity [including "Asymmetry" ASYMHTHF and "Centrality" bits (7:4)]:



2.2.2 Global Muon Trigger optical interfaces

The configuration of optical connections from Global Muon Trigger to μGT is shown in Table **??**.

The data structure of a muon object (bit 34 = charge sign, bit 35 = charge valid, bits 36..63 are spare bits):



2.3 Implementation in firmware

Remark: all definitions for scales in the following chapters are from a CMS Detector Note: "Scales for inputs to μGT" (see actual version in http://globaltrigger.hephy.at/upgrade/ugt/downloads/).

The firmware of μGTL consists of two main parts:

- A top-of-hierarchy file (`gtl_module.vhd`), which contains the pipeline for $\pm$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. Both files generated by VHDL Producer for every Trigger Menu (see Figure 2 and Section **??**). In addition, VHDL Producer creates a VHDL files for the mapping of Algorithms (`algo_mapping_rop.vhd`) for μFDL.
- A set of VHDL-files 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.

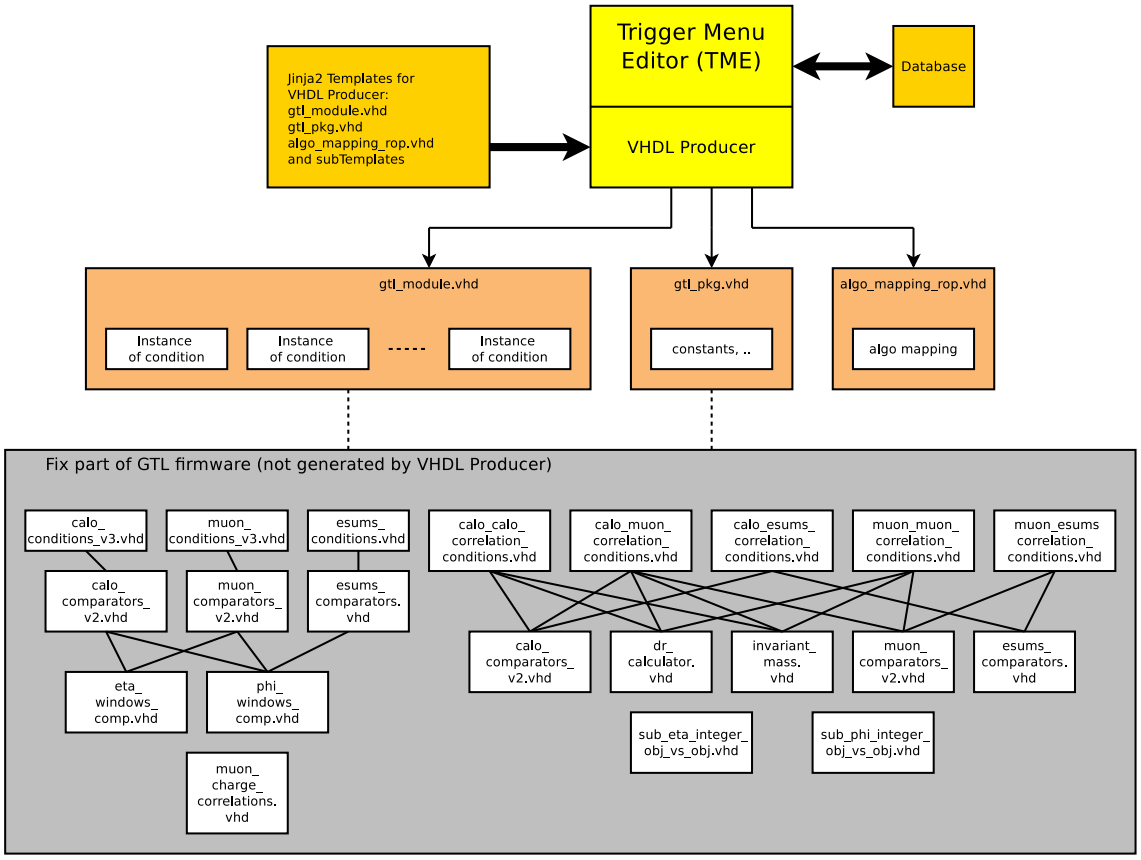


Figure 2: VHDL file generation by VHDL Producer

The latency of μGTL is fixed to 5 bunch crossings, 2 bunch crossings for the pipeline of $\pm$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 3).

If there are not enough firmware resources in FPGA of one AMC board for 512 Algorithms, more boards could be used. Therefore the 512 Algorithms are **partitioned by TME** (e.g. the first board covers Algorithms 0..200 and the second Algorithms 201..511). Trigger Menu Editor (TME) will give the number of Algorithms per module as constant in the package module `gtl_pkg.vhd`. This means μGTL and μFDL firmware considered as a unit for synthesis.

2.3.1 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 2 contains the entity-declaration of the top-of-hierarchy file (`gtl_module.vhd`).

Listing 2: 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);
```

```
--
    ********************************************************************************

        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;
```

2.3.2 Package module

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

2.4 μGTL structure

2.4.1 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 3 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.

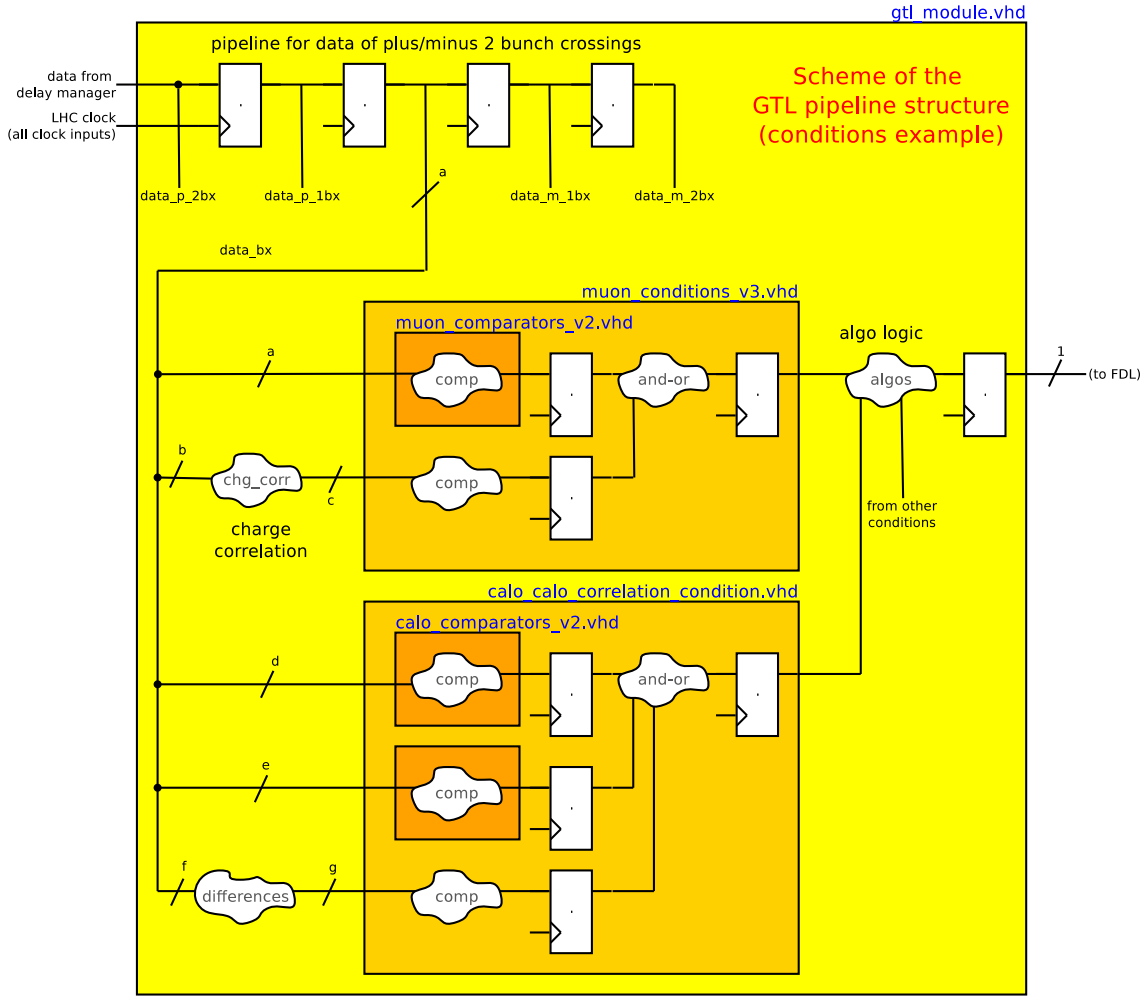


Figure 3: Scheme of μGTL pipeline structure

2.4.2 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 dif-

ferences 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 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_signed_eta.vhd` as many times as the numbers of both objects determine.
In the module `sub_signed_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.

2.4.3 Calorimeter conditions

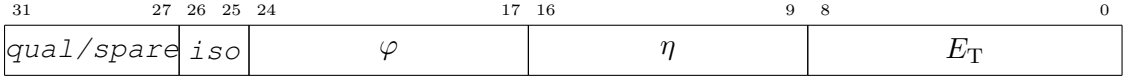
2.4.3.1 Calorimeter data

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 E_{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 = -3.0 to 3.0, 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 (meaning not defined yet!)
- 5 bits quality and spare (not defined yet!)

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



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 E_{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 $\approx 2\pi/144$ ($\triangleq$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 5 bits quality and spare (not defined yet!)

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}

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 E_{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 = -3.0 to 3.0, step = 0.087/2, linear scale, 138 bins (HW index = 0xBC..0x44)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\triangleq$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 2 bits isolation (meaning not defined yet!)
- 5 bits quality and spare (not defined yet!)

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 representation of the 8 bits (called "hardware index [HW index]") in η is expected as Two's Complement notation as shown in Table 3.
The content of this table has to be checked.

The representation of the 8 bits in φ is expected as shown in Table 4.
The content of this table has to be checked.

The representation of the 2 bits for isolation (e/γ and tau) is expected as shown in Table 5.
The content of this table has to be checked.

Table 2: η scale of electron/γ and tau

HW index	η **range**	η **bin**
0x44	68∗0.087/2 to 69∗0.087/2	68
...	...	...
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
...	...	...
0xBC	-68∗0.087/2 to -69∗0.087/2	-69

Table 3: η scale of jet

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

Table 4: φ scale of calorimeter objects

HW index	φ range	φ range [degrees]	φ bin
0x00	0 to 2π/144	0 to 2.5	0
0x01	2π/144 to 2∗2π/144	2.5 to 5.0	1
...	...	...	...
0x8F	143∗2π/144 to 2π	357.5 to 360	143

Table 5: Definition of e/γ and tau isolation bits

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

2.4.3.2 Calorimeter conditions definition

A condition consists of calorimeter objects as input data and a set of requirements, which contain the requirements to be complied.

The requirement for calorimeter conditions contains:
one threshold for E_{T}, ranges for η, φ LUTs for isolation and differences in η and φ. In addition the selection of the "relative bx" of objects is done in the requirement.
The condition is complied, if every comparison between object parameters and requirements is valid for the following equation:

- E_{T} greater-equal or equal threshold
- η in range
- φ in range
- isolation as requested (for electron/γ and tau)

Additional comparisons for "quality information" could be part of the equation - but not defined yet.

There are different types of calorimeter 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 selection of the mode of E_{T}-comparator (greater/equal or equal), the E_{T}-threshold-value, ranges for η and φ set with thresholds, LUTs for isolation and ranges for differences in η and φ set with thresholds are fixed values given by VHDL Producer for every Trigger Menu.

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

2.4.3.2.1 Calorimeter conditions over bx

The description in this chapter is very preliminary and only valid, if the objects coming to μGT do not occur in more than one bx!

If there are **different "relative bx" in the requirements** of a condition type, VHDL Producer creates a set of "sub-conditions", which are combined with an logical "AND", to get a "condition-over-bx". With this method no "special" condition types are needed to create a "condition-over-bx" in VHDL-code. The label of the instance and the signal name of "sub-conditions" in VHDL-code are the same as the condition name in VHDL Producer, but with a suffix as index, which indicates the "sub-conditions" belonging to the "condition-over-bx". The "sub-conditions" created by VHDL Producer are the following:

- "Quad objects requirements condition over bx":
  - Four different bx in the requirements => Four "single object requirement condition" combined by an "AND".
  - Three requirements of same bx and one requirements of other bx => One "triple object requirements condition" and one "single object requirement condition" combined by an "AND".
  - Two requirements of same bx and two requirements of other, but same bx => Two "double object requirements condition" combined by an "AND".
- "Triple objects requirements condition over bx":
  - Three different bx in the requirements => Three "single object requirement condition" combined by an "AND".
  - Two requirements of same bx and one requirements of other bx => One "double object requirements condition" and one "single object requirement condition" combined by an "AND".
- "Double objects requirements condition over bx":
  - Two requirements with different bx => Two "single object requirement condition" combined by an "AND".

2.4.3.2.2 Calorimeter conditions module

The module for conditions with calorimeter objects (`calo_conditions.vhd`) instantiates the calorimeter comparators module (`calo_comparators.vhd`) as many times as the numbers of objects and requirements determine. Depending on the condition-type, different and-or-structures of object vs. requirement are selected. The selection of condition-type and the number of objects is done by parameters in the generic interface list of the module (see Listing 3, see also the explanations below).
For comparison in η, φ and the differences in η and φ, "window"-comparators are used.
In the calorimeter conditions module a cut for two-body pt calculation can be selected (see 2.4.9.1.4). Therefore a threshold value for two-body pt is required.

Listing 3: Entity declaration of `calo_conditions.vhd`

```
entity calo_conditions is
     generic(
        calo_object_slice_1_low: natural;
        calo_object_slice_1_high: natural;
        calo_object_slice_2_low: natural;
        calo_object_slice_2_high: natural;
        calo_object_slice_3_low: natural;
        calo_object_slice_3_high: natural;
        nr_templates: positive;
        et_ge_mode: boolean;
        obj_type : natural := EG_TYPE;
        et_thresholds: calo_templates_array;
        eta_full_range : calo_templates_boolean_array;
        eta_w1_upper_limits: calo_templates_array;
        eta_w1_lower_limits: calo_templates_array;
        eta_w2_ignore : calo_templates_boolean_array;
        eta_w2_upper_limits: calo_templates_array;
        eta_w2_lower_limits: calo_templates_array;
        phi_full_range : calo_templates_boolean_array;
        phi_w1_upper_limits: calo_templates_array;
        phi_w1_lower_limits: calo_templates_array;
        phi_w2_ignore : calo_templates_boolean_array;
        phi_w2_upper_limits: calo_templates_array;
        phi_w2_lower_limits: calo_templates_array;
        iso_luts: calo_templates_iso_array;

        twobody_pt_cut: boolean := false;
        pt_width: positive := 1;
        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0) := (others => '0');
        sin_cos_width: positive := 1;
        pt_sq_sin_cos_precision : positive := 1

    );
    port(
        clk: in std_logic;
        data_i: in calo_objects_array;
        condition_o: out std_logic;
        pt : in diff_inputs_array(0 to MAX_CALO_OBJECTS) := (others => (others =>
             '0'));
        cos_phi_integer : in calo_sin_cos_integer_array(0 to MAX_CALO_OBJECTS) :=
             (others => 0);
        sin_phi_integer : in calo_sin_cos_integer_array(0 to MAX_CALO_OBJECTS) :=
             (others => 0)
   );
end calo_conditions;
```

Table 6: Explanation of Listing 3

Item	Explanation
calo_object_slice_1_low	low value of slice for object 1.
calo_object_slice_1_high	high value of slice for object 1.
calo_object_slice_2_low	low value of slice for object 2.
calo_object_slice_2_high	high value of slice for object 2.
calo_object_slice_3_low	low value of slice for object 3.
calo_object_slice_3_high	high value of slice for object 3.
calo_object_slice_4_low	low value of slice for object 4.
calo_object_slice_4_high	high value of slice for object 4.
nr_templates	valid values are 1 (for single), 2 (double), 3 (triple) and 4 (quad) - depending on condition type.
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 'EG_TYPE', 'JET_TYPE', and 'TAU_TYPE'.
et_thresholds	array of four threshold values for comparison in E_T (four thresholds, because of max. 4 requirements).
nr_eta_windows	array of four integer values for number of η cuts.
eta_w1_upper_limits	array of four "upper limits" of "window"-comparator 1 for η.
eta_w1_lower_limits	array of four "lower limits" of "window"-comparator 1 for η.
eta_w2_upper_limits	array of four "upper limits" of "window"-comparator 2 for η.
eta_w2_lower_limits	array of four "lower limits" of "window"-comparator 2 for η.
eta_w3_upper_limits	array of four "upper limits" of "window"-comparator 3 for η.
eta_w3_lower_limits	array of four "lower limits" of "window"-comparator 3 for η.
eta_w4_upper_limits	array of four "upper limits" of "window"-comparator 4 for η.
eta_w4_lower_limits	array of four "lower limits" of "window"-comparator 4 for η.
eta_w5_upper_limits	array of four "upper limits" of "window"-comparator 5 for η.
eta_w5_lower_limits	array of four "lower limits" of "window"-comparator 5 for η.
phi_full_range	array of four boolean to set full range of φ.
phi_w1_upper_limits	array of four "upper limits" of "window"-comparator 1 for φ.
phi_w1_lower_limits	array of four "lower limits" of "window"-comparator 1 for φ.
phi_w2_ignore	array of four boolean to ignore "window"-comparator 2 for φ.
phi_w2_upper_limits	array of four "upper limits" of "window"-comparator 2 for φ.
phi_w2_lower_limits	array of four "lower limits" of "window"-comparator 2 for φ.
iso_luts	array of four LUTs for comparison of 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).
sin_cos_width	vector length of sine and cosine.
pt_sq_sin_cos_precision	precision of sine and cosine calculation in LUTs.
clk	clock input (LHC clock).
data_i	data, structure defined in obj_type.
condition_o	output of condition (routed to Algorithms logic, see 2.4.11).
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.

2.4.3.2.3 Calorimeter Overlap Remover conditions module

The Calorimeter Overlap Remover conditions consits of a Calorimeter condition (2.4.3.2.2) and a single condition for a different calo object type. A correlation cut for overlap removal is required between objects of Calorimeter condition and objects of the different calo object type. Overlap Remover conditions `calo_conditions_orm.vhd` are implemented only for calo object types.

2.4.3.2.4 Calorimeter comparators module

A comparator between the energy (E_T) and a threshold (et_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 E_T threshold, the 'mode-selection' for the E_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. There are two flags, one for "full-range" and one for "ignore-second-window" for the selection of the ranges.
There are two cases how the limits of one "window"-comparator could be set (see also Figure 4 and Listing 4):

- 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).

The comparison of isolation (for electron/γ and tau) is done with LUTs.

Listing 4: 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';
```

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 and tau) is done with LUTs (see Table 7). Only the least significant 4 bits of LUT are used, because currently 2 isolation bits are defined.

Table 7: LUT contents for isolation comparison of electron/γ and tau objects

LUT content (16 bits)	isolation bits [26..25]	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)

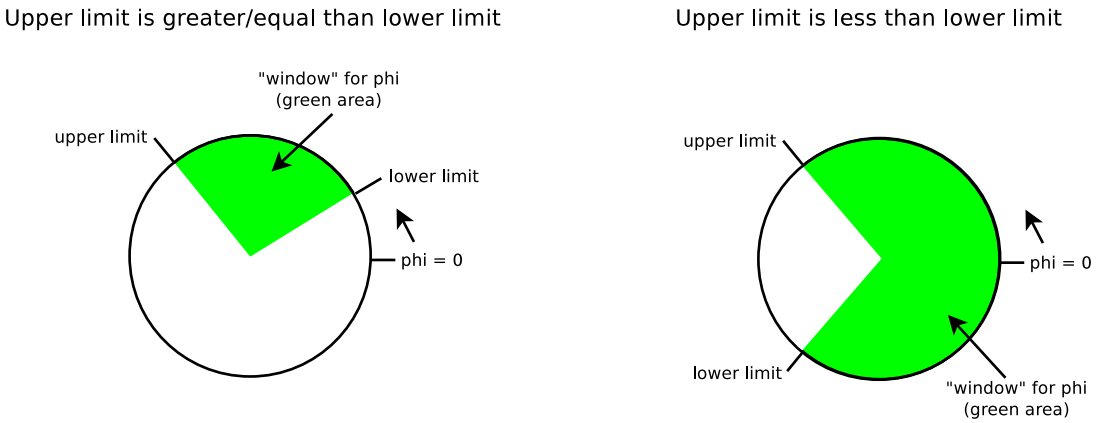


Figure 4: Setting the limits for "window"-comparators for φ

2.4.4 Energy sum quantities conditions

energy sum quantities:
Consists of following quantities (naming convention see 3):

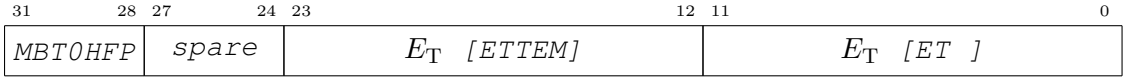
- **ET**
- **HT**
- ET_{miss}
- HT_{miss}
- **ETTEM**
- $\mathbf{ET}^{HF}_{miss}$
- $\mathbf{HT}^{HF}_{miss}$
- **ASYMET**
- **ASYMHT**
- **ASYMETHF**
- **ASYMHTHF**
- **CENT0**
- ..
- **CENT7**

2.4.4.1 Energy sum quantities data

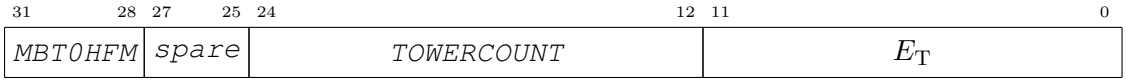
Calo-Layer2 sends 6 frames (each 32 bits) with Energy sum quantities containing the following information:

- 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

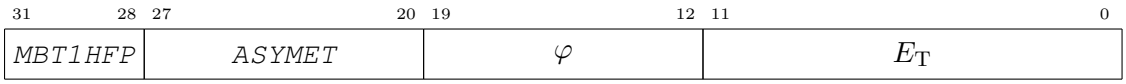
Frame0: The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits]:



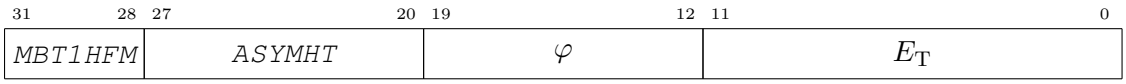
Frame1: The data structure of "total calibrated Et in jets" (HT) quantity [including "towercount" and "minimum bias HF- threshold 0" bits]:



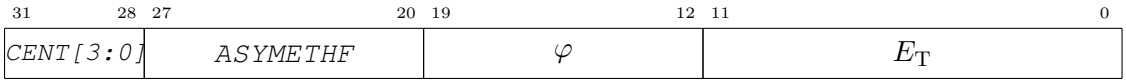
Frame2: The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits]:



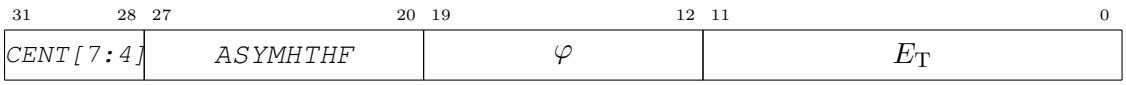
Frame3: The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits]:



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



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



2.4.4.2 Energy sum quantities conditions module

For the entity-declaration of `esums_conditions.vhd`, see Listing 5.

Listing 5: 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 8: Explanation of Listing 5

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 2.4.11).

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 2.4.3.2.4). 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.

2.4.5 Minimum bias trigger conditions

Data for Minimum bias trigger are received on the 4 MSBs of 4 frames used for Energy sum quantities (see 2.4.4).

- 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 the min_bias_hf_conditions.vhd module there is a comparision with a 4-bit threshold (greater-equal or equal).

2.4.6 Towercount condition

Data for Towercount trigger (number of firing HCAL towers) are received on frame HT (see 2.4.4) on bits 24..12 (13 bits) of HT data structure. In the towercount_condition.vhd module there is a comparision with a 13-bit threshold (greater-equal or equal).

2.4.7 Centrality condition

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

2.4.8 Muon conditions

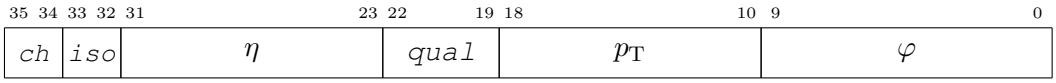
2.4.8.1 Muon data

Eight Muon objects are provided by Global Muon Trigger. One Muon object has a 64 bits data structure with parameters for p_T, for position, charge, quality and isolation information:

- 10 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/576$ ($\hat{=}$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")
- 28 bits tag and spare (not defined yet!)

The data structure of a muon object (bit 34 = charge sign, bit 35 = charge valid, bits 36..63 are spare bits):



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.
The content of this table has to be checked.

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.
The content of this table has to be checked.

The representation of the 4 bits for quality is expected as shown in Table 11.
The content of this table has to be checked.

The representation of the 2 bits for isolation is expected as shown in Table 12.
The content of this table has to be checked.

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

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

2.4.8.2 Muon charge correlation module

For definition of muon charge, see 2.4.8.
In the muon charge correlation module, 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 13: 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 14: 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 15: 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

2.4.8.3 Muon conditions definition

A condition consists of input-data and a set of requirements, which contain the requirements to be complied.

The requirement for muon conditions contains:
a threshold for p_{T}, ranges for η and φ, a LUT for quality, a LUT for isolation, a requsted charge. The condition is complied, if every comparison between object parameters and requirements is valid for the following equation:

- p_{T} greater-equal or equal threshold
- η in range
- φ in range
- requested charge
- quality LUT
- iso LUT

There are different types of calorimeter 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.[2]
- "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.

In addition requested charge correlation must be matched (except for "Single object requirement condition", there is no charge correlation). The calculation of charge correlations is done in an own module in the top-of-hierarchy module (`gtl_module.vhd`).

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

2.4.8.3.1 Muon conditions module

A module for conditions with muon objects (`muon_conditions.vhd`) instantiates the muon comparators module (`muon_comparators.vhd`) as many times as the numbers of objects and requirements determine. Depending on the condition-type different and-or-structures of object vs. requirement are selected. The selection of condition-type and the number of objects is done by parameters in the generic interface list of the module (see the following VHDL entity definition in Listing 6).

Listing 6: Entity declaration of `muon_conditions.vhd`

```
entity muon_conditions is
    generic (
        muon_object_slice_1_low: natural;
        muon_object_slice_1_high: natural;
        muon_object_slice_2_low: natural;
        muon_object_slice_2_high: natural;
        muon_object_slice_3_low: natural;
        muon_object_slice_3_high: natural;
        muon_object_slice_4_low: natural;
        muon_object_slice_4_high: natural;
        nr_templates: positive;
        pt_ge_mode : boolean;
        pt_thresholds: muon_templates_array;
        eta_full_range : muon_templates_boolean_array;
        eta_w1_upper_limits: muon_templates_array;
        eta_w1_lower_limits: muon_templates_array;
        eta_w2_ignore : muon_templates_boolean_array;
        eta_w2_upper_limits: muon_templates_array;
        eta_w2_lower_limits: muon_templates_array;
        phi_full_range : muon_templates_boolean_array;
        phi_w1_upper_limits: muon_templates_array;
        phi_w1_lower_limits: muon_templates_array;
        phi_w2_ignore : muon_templates_boolean_array;
        phi_w2_upper_limits: muon_templates_array;
        phi_w2_lower_limits: muon_templates_array;
        requested_charges: muon_templates_string_array;
        qual_luts: muon_templates_quality_array;
        iso_luts: muon_templates_iso_array;
        requested_charge_correlation: string(1 to 2);

        twobody_pt_cut: boolean := false;
        pt_width: positive := 1;
        pt_sq_threshold_vector: std_logic_vector(MAX_WIDTH_TBPT_LIMIT_VECTOR-1
            downto 0) := (others => '0');
        sin_cos_width: positive := 1;
        pt_sq_sin_cos_precision : positive := 1

    );
    port(
        lhc_clk : in std_logic;
        data_i : in muon_objects_array;
        condition_o : out std_logic;
        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'))));
        pt : in diff_inputs_array(0 to NR_MUON_OBJECTS-1) := (others => (others
            => '0'));
```

```
        cos_phi_integer : in muon_sin_cos_integer_array(0 to NR_MUON_OBJECTS-1)
            := (others => 0);
        sin_phi_integer : in muon_sin_cos_integer_array(0 to NR_MUON_OBJECTS-1)
            := (others => 0)
    );
end muon_conditions;
```

Table 16: Explanation of Listing 6

Item	Explanation
muon_object_slice_1_low	low value of slice for object 1.
muon_object_slice_1_high	high value of slice for object 1.
muon_object_slice_2_low	low value of slice for object 2.
muon_object_slice_2_high	high value of slice for object 2.
muon_object_slice_3_low	low value of slice for object 3.
muon_object_slice_3_high	high value of slice for object 3.
muon_object_slice_4_low	low value of slice for object 4.
muon_object_slice_4_high	high value of slice for object 4.
nr_templates	number of requirements, selector of condition-type. Valid values are 1, 2, 3 and 4.
pt_ge_mode	'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	array of four threshold values for comparison in pt (four threshold, because of max. 4 requirements).
nr_eta_windows	array of four integer values for number of η cuts.
eta_w1_upper_limits	array of four "upper limits" of "window"-comparator 1 for η.
eta_w1_lower_limits	array of four "lower limits" of "window"-comparator 1 for η.
eta_w2_upper_limits	array of four "upper limits" of "window"-comparator 2 for η.
eta_w2_lower_limits	array of four "lower limits" of "window"-comparator 2 for η.
eta_w3_upper_limits	array of four "upper limits" of "window"-comparator 3 for η.
eta_w3_lower_limits	array of four "lower limits" of "window"-comparator 3 for η.
eta_w4_upper_limits	array of four "upper limits" of "window"-comparator 4 for η.
eta_w4_lower_limits	array of four "lower limits" of "window"-comparator 4 for η.
eta_w5_upper_limits	array of four "upper limits" of "window"-comparator 5 for η.
eta_w5_lower_limits	array of four "lower limits" of "window"-comparator 5 for η.
phi_full_range	array of four boolean to set full range of φ.
phi_w1_upper_limits	array of four "upper limits" of "window"-comparator 1 for φ.
phi_w1_lower_limits	array of four "lower limits" of "window"-comparator 1 for φ.
phi_w2_ignore	array of four boolean to ignore "window"-comparator 2 for φ.
phi_w2_upper_limits	array of four "upper limits" of "window"-comparator 2 for φ.
phi_w2_lower_limits	array of four "lower limits" of "window"-comparator 2 for φ.
requested_charges	array of four strings for requested charge ("pos" means "positive charge", "neg" means "negative charge" and "ign" means "ignore charge").
qual_luts	array of four LUTs (16 bits) for quality.
iso_luts	array of four LUTs (4 bits) for isolation.
requested_charge_correlation	string (2 characters) for requested charge correlation ("ls" means "like sign", "os" means "opposite sign" or "ig" means "ignore").

Table 16: Explanation of Listing 6

Item	Explanation
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).
sin_cos_width	vector length of sine and cosine.
pt_sq_sin_cos_precision	precision of sine and cosine calculation in LUTs.
lhc_clk	clock input (LHC clock).
data_i	input data, structure defined in d_s_i.
condition_o	output of condition (routed to Algorithms logic, see 2.4.11).
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.
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.

2.4.8.3.2 Muon comparators module

A comparator between p_T and a threshold (pt_threshold), a comparison in η with five "window"-comparators and φ with two "window"-comparators, a comparison of quality with LUT a comparison of isolation with LUT and a comparison of the requested charge is done in this basic module. The values for p_T threshold, the 'mode-selection' for the p_T comparator, the "limits" of the "window"-comparators, the quality LUTs, the isolation LUTs and the requested charge 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. There are two flags, one for "full-range" and one for "ignore-second-window" for the selection of the ranges.
There are two cases how the limits of one "window"-comparator could be set (see also Figure 4 and Listing 4):

- 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).

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

Charge valid and charge sign bits must be equal to the requested charge.
The comparison of quality is done with LUT. To ignore quality comparison, all bits in the LUT have to be '1'.

Table 17: LUT contents for quality comparison of muon objects

LUT content (16 bits)	quality bits [22..19]	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")

The comparison of isolation is done with LUT. To ignore isolation comparison, all bits in the LUT have to be '1' (see Table 18).

Table 18: LUT contents for isolation comparison of muon objects

LUT content (4 bits)	isolation bits [33..32]	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)

2.4.9 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 requirements" there are cuts for $\Delta\eta$, $\Delta\varphi$, ΔR and mass. Only one correlation cut is allowed in a condition, except the combination of one $\Delta\eta$ and one $\Delta\varphi$ cut.

The following cuts can be used:

- Cuts for $\Delta\eta$ (DETA) and/or $\Delta\varphi$ (DPHI).
- Cut for ΔR (DR).
- Cuts for mass (MASS) of following mass types:
  - Cut for Invariant mass.
  - Cut for Invariant mass with "two-body pt".
  - Cut for Transverse mass.
  - Cut for Transverse mass with "two-body pt".

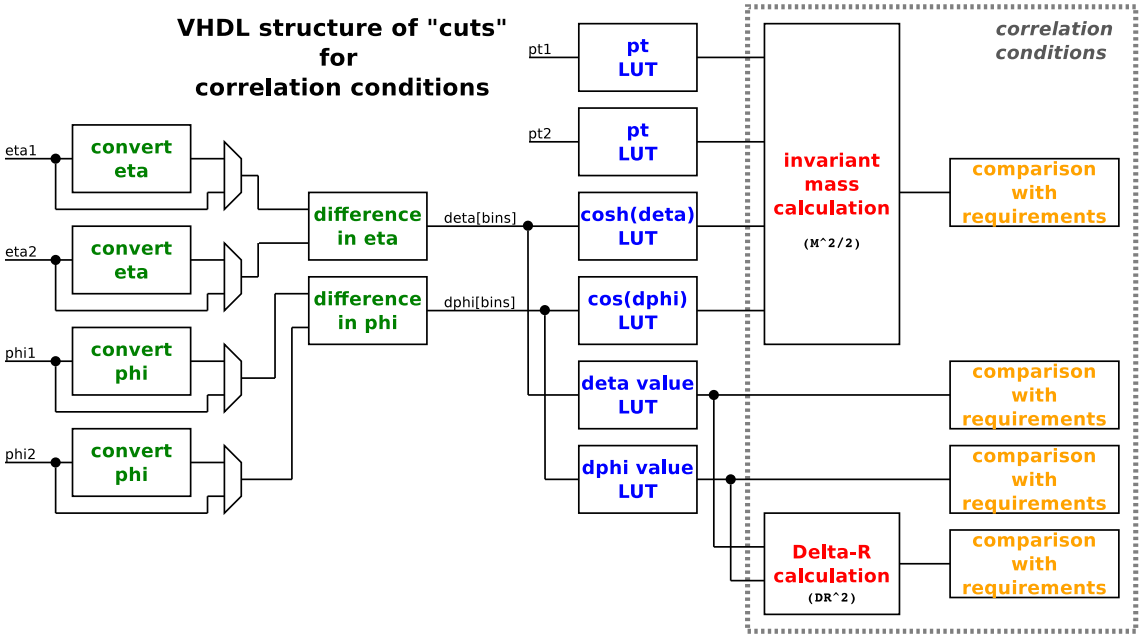


Figure 5: VHDL structure of cuts for correlation conditions

2.4.9.1 Calculation of cuts

Calculation of $\Delta\eta$ and $\Delta\varphi$ see section "Calculation of differences in η and φ" (2.4.2).

2.4.9.1.1 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.

2.4.9.1.2 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.[3] 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.

2.4.9.1.3 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 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.

2.4.9.1.4 Two-body pt calculation

The calculation of two-body pt is done with formula:
$pt=\sqrt{pt_1^2 + pt_2^2 + 2pt_1pt_2(\cos(\varphi_1)\cos(\varphi_2) + \sin(\varphi_1)\sin(\varphi_2))}$
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.

2.4.9.2 Correlation condition modules

As described in section Correlation conditions (2.4.9), correlations of two object types are done. 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.)

[3] 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.

- 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 with calorimeter objects and energy sum quantities (ET_{miss}, ET^{HF}_{miss} 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 muon objects and energy sum quantities (ET_{miss}, ET^{HF}_{miss} and HT_{miss} only) (`muon_esums_correlation_condition.vhd`: muon-etm, muon-etmhf and muon-htm are possible.)

2.4.9.2.1 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 (DETA) and φ (DPHI) or a calculation of ΔR (DR) or a calculation of mass (MASS), see Figure 5.
The cut of mass is available for Invariant mass or Transverse mass or one of both with 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. 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 3, 4, 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 2.4.9.1.2) and Transverse mass (formular see 2.4.9.1.3) 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.[4]
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 2.4.9.1.4) 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). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for DETA and DPHI 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 calo calo correlation condition module (version 4) in `calo_calo_correlation_condition.vhd`, see Listing 7.

[4]Definition of "`constant CALO_INV_MASS_COSH_COS_PRECISION...`" in file `gtl_pkg.vhd`. 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_v4 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);
        eta_full_range_calo1: boolean;
        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_ignore_calo1: boolean;
        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);
        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);
        eta_full_range_calo2: boolean;
        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_ignore_calo2: boolean;
        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);
        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 calo_sin_cos_integer_array;
        cos_phi_2_integer : in calo_sin_cos_integer_array;
        sin_phi_1_integer : in calo_sin_cos_integer_array;
        sin_phi_2_integer : in calo_sin_cos_integer_array;
        condition_o: out std_logic
    );
end calo_calo_correlation_condition_v4;
```

Table 19: 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 DETA cut.
dphi_cut	boolean for using DPHI 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.
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).

Table 19: Explanation of Listing 7

Item	Explanation
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.
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 DETA and DPHI.
mass_upper_limit	"upper limit" of "window"-comparator for comparison of $\frac{M^2}{2}$ (hex value).

Table 19: Explanation of Listing 7

Item	Explanation
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 2.4.2.
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 $GeV \times 10$].[5]
pt2	calo2 E_T values [from LUT, in $GeV \times 10$].
cosh_deta	cosh($\Delta\eta$) values [from LUT, cosh($\Delta\eta$)×1000].[6]
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 2.4.11).

2.4.9.2.2 Calo Muon Correlation condition module

The calo muon correlation condition module 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 with "Cuts" for differences in η (DETA), φ (DPHI) or a calculation of ΔR (DR) or a calculation of mass (MASS).

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

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

The cut of mass is available for Invariant mass or Transverse mass or one of both with 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. EG_MUON_DIFF_ETA_LUT, EG_MUON_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] × 4 + 2",
of EG_PHI_CONV_2_MUON_PHI_LUTwith formular:
"converted-calo-phi[bin] = calo-phi[bin] × 4 + 2".
The conversion calculations are preliminary, others may be proposed.

Definitions of scales (see Tables 3, 4, 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 2.4.9.1.2) and Transverse mass (formular see 2.4.9.1.3) 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.[7]
The contents of the LUTs for $\cos(\varphi)$ (CALO_COS_PHI_LUT and MUON_COS_PHI_LUT) and $\sin(\varphi)$ (CALO_SIN_PHI_LUT and MUON_SIN_PHI_LUT) for two-body pt (formular see 2.4.9.1.4) 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"

[7] 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`.

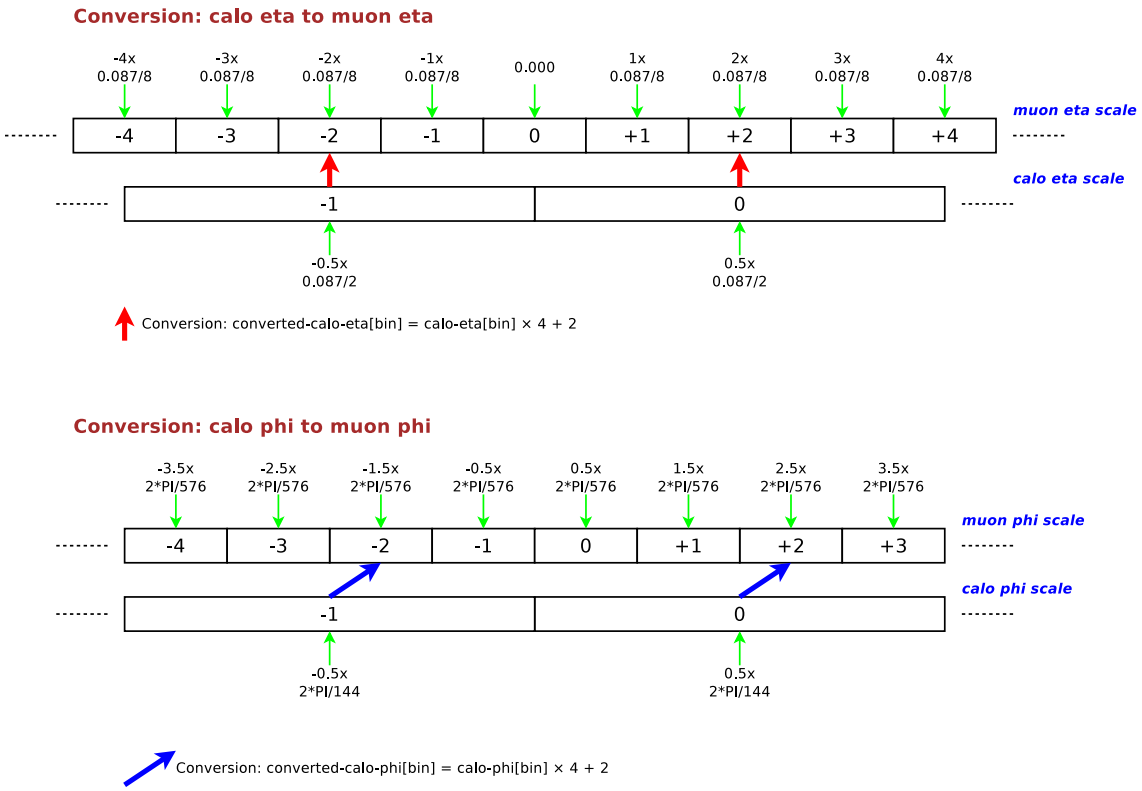


Figure 6: Conversion of calorimeter η and φ to muon scales

list of the entity declaration of the module and are expressed in hex notation. The limits for DETA and DPHI are expressed with the 3rd position after decimal point, for DR and MASS with the 1st position after decimal point.

For the VHDL entity declaration of calo muon correlation condition module (version 3) in `calo_muon_correlation_condition.vhd`, see Listing 8.

Listing 8: 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);
        eta_full_range_calo: boolean;
        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_ignore_calo: boolean;
        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);
        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);
        eta_full_range_muon : boolean;
        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_ignore_muon : boolean;
        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);
        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);

        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 muon_sin_cos_integer_array;
        cos_phi_2_integer : in muon_sin_cos_integer_array;
        sin_phi_1_integer : in muon_sin_cos_integer_array;
        sin_phi_2_integer : in muon_sin_cos_integer_array;
        condition_o: out std_logic
    );
```

```
end calo_muon_correlation_condition;
```

Table 20: Explanation of Listing 8

Item	Explanation
deta_cut	boolean for using DETA cut.
dphi_cut	boolean for using DPHI 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}.
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).

Table 20: Explanation of Listing 8

Item	Explanation
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.
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 DETA and DPHI.

Table 20: Explanation of Listing 8

Item	Explanation
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(φ) and sin(φ) of calos.
sin_cos_width_2	number of bits for the maximum value in the LUT for cos(φ) and sin(φ) of muon.
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).
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 2.4.2.
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 $GeV\times10$].[8]
pt2	muon p_T values [from LUT, in $GeV\times10$].
cosh_deta	cosh($\Delta\eta$) values [from LUT, cosh($\Delta\eta$)×10000].[9]
cos_dphi	cos($\Delta\varphi$) values [from LUT, cosh($\Delta\varphi$)×10000].
cos_phi_1	cos(φ) values from LUT for calo.
cos_phi_2	cos(φ) values from LUT for muon.
sin_phi_1	sin(φ) values from LUT for calo.
sin_phi_2	sin(φ) values from LUT for muon.
condition_o	output of condition (routed to Algorithms logic, see 2.4.11).

[8] value 10 from $10^{\text{CALO\_MUON\_INV\_MASS\_PT\_PRECISION}}$
[9] value 10000 from $10^{\text{CALO\_MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$

2.4.9.2.3 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 2.4.8.2) is required.
In addition there are "Cuts" for differences in η (DETA), φ (DPHI) or a calculation of ΔR (DR) or a calculation of mass (MASS).
The cut of mass is available for Invariant mass or Transverse mass or one of both with 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 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 2.4.9.1.2) and Transverse mass (formular see 2.4.9.1.3) are created by calculating hyperbolic cosine and cosine, rounding-up at the 4^{th} position after decimal point, and multiplying by 10000 to get integer values.[10]
The contents of the LUTs for $\cos(\varphi)$ (MUON_COS_PHI_LUT) and $\sin(\varphi)$ (MUON_SIN_-PHI_LUT) for two-body pt (formular see 2.4.9.1.4) are created by calculating cosine and sine, rounding-up at the 3^{rd} 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). This limits are parts of the "generic" list of the entity declaration of the module and are expressed in hex notation. The limits for DETA and DPHI are expressed with a precision of 3^{rd} position after decimal point, for DR and MASS with 1^{st} position after decimal point.

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

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

Listing 9: Entity declaration of muon_muon_correlation_condition.vhd

```
entity muon_muon_correlation_condition_v4 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);
        eta_full_range_muon1: boolean;
        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_ignore_muon1: boolean;
        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);
        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);

        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);
        eta_full_range_muon2: boolean;
        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_ignore_muon2: boolean;
        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);
        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);

        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;
        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;
```

```
        cosh_deta : in muon_cosh_cos_vector_array;
        cos_dphi : in muon_cosh_cos_vector_array;
        cos_phi_1_integer : in muon_sin_cos_integer_array;
        cos_phi_2_integer : in muon_sin_cos_integer_array;
        sin_phi_1_integer : in muon_sin_cos_integer_array;
        sin_phi_2_integer : in muon_sin_cos_integer_array;
        condition_o: out std_logic
    );
end muon_muon_correlation_condition_v4;
```

Table 21: Explanation of Listing 9

Item	Explanation
same_bx	boolean indicating whether data are from same Bx - 'true' for same Bx.
deta_cut	boolean for using DETA cut.
dphi_cut	boolean for using DPHI 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).
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.

Table 21: Explanation of Listing 9

Item	Explanation
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.
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.
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).

Table 21: Explanation of Listing 9

Item	Explanation
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 DETA and DPHI.
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).
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 2.4.2.
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	calorimeter E_T values [from LUT, in $GeV\times10$].[11]
pt2	muon p_T values [from LUT, in $GeV\times10$].
cosh_deta	cosh($\Delta\eta$) values [from LUT, cosh($\Delta\eta$)×10000].[12]
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).
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).

[11] value 10 from $10^{\text{CALO\_MUON\_INV\_MASS\_PT\_PRECISION}}$
[12] value 10000 from $10^{\text{CALO\_MUON\_INV\_MASS\_COSH\_COS\_PRECISION}}$

Table 21: Explanation of Listing 9

Item	Explanation
`condition_o`	output of condition (routed to Algorithms logic, see 2.4.11).

2.4.9.2.4 Calo Esums Correlation condition module

The calo esums correlation condition module 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 (DPHI) 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 $10^{\text{precision}}$.

The contents of the LUTs $\cos(\Delta\varphi)$ (EG_ETM_COS_DPHI_LUT, ...) for Transverse mass (formular see 2.4.9.1.3) are created by calculating cosine, rounding-up at the 3rd position after decimal point and multiplying by 1000 to get integer values.[13]
The contents of the LUTs for $\cos(\varphi)$ (CALO_COS_PHI_LUT) and $\sin(\varphi)$ (CALO_SIN_PHI_LUT) for two-body pt (formular see 2.4.9.1.4) 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 DPHI 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 10.

Listing 10: Entity declaration of `calo_esums_correlation_condition.vhd`

```
entity calo_esums_correlation_condition is
    generic(

        dphi_cut: boolean;
        mass_cut: boolean;
        mass_type : natural;
```

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

```
        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);
        eta_full_range_calo: boolean;
        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_ignore_calo: boolean;
        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);
        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 calo_sin_cos_integer_array;
        cos_phi_2_integer : in calo_sin_cos_integer_array;
        sin_phi_1_integer : in calo_sin_cos_integer_array;
        sin_phi_2_integer : in calo_sin_cos_integer_array;
        condition_o: out std_logic
    );
end calo_esums_correlation_condition;
```

Table 22: Explanation of Listing 10

Item	Explanation
dphi_cut	boolean for using DPHI 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 22: Explanation of Listing 10

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 DPHI.
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(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).
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 $GeV \times 10$].[14]
pt2	esums E_T values [from LUT, in $GeV \times 10$].
cos_dphi	cos($\Delta\varphi$) values from LUT.
cos_phi_1	cos(φ) values from LUT for calo1.
cos_phi_2	cos(φ) values from LUT for esums.
sin_phi_1	sin(φ) values from LUT for calo1.
sin_phi_2	sin(φ) values from LUT for esums.
condition_o	output of condition (routed to Algorithms logic, see 2.4.11).

[14] value 10 from $10^{\text{CALO\_INV\_MASS\_PT\_PRECISION}}$

2.4.9.2.5 Muon Esums Correlation condition module

The muon esums correlation condition module 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 (DPHI) 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 6). 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 2.4.9.1.3) are created by calculating cosine, rounding-up at the 4^{th} 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 $\cos(\varphi)$ (CALO_COS_PHI_LUT and MUON_COS_PHI_LUT) and $\sin(\varphi)$ (CALO_SIN_PHI_LUT and MUON_SIN_PHI_LUT) for two-body pt (formular see 2.4.9.1.4) are created by calculating cosine and sine, rounding-up at the 3^{rd} 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 DPHI are expressed with a precision of 3^{rd} position after decimal point, for MASS with 1^{st} position after decimal point.

[15] Definition of "`constant MU_ETM_COSH_COS_PRECISION ...`" and "`constant CALO_PHI_-CONV_2_MUON_PHI_LUT ...`" in file `gtl_pkg.vhd`.

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
     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);
        eta_full_range_muon : boolean;
        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_ignore_muon : boolean;
        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);
        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);

        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 muon_sin_cos_integer_array;
        cos_phi_2_integer : in muon_sin_cos_integer_array;
        sin_phi_1_integer : in muon_sin_cos_integer_array;
        sin_phi_2_integer : in muon_sin_cos_integer_array;
        condition_o: out std_logic
    );
end muon_esums_correlation_condition;
```

Table 23: Explanation of Listing 9

Item	Explanation
dphi_cut	boolean for using DPHI 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.
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.

Table 23: Explanation of Listing 9

Item	Explanation
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 DPHI.
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(φ) and sin(φ) of muon.
sin_cos_width_2	number of bits for the maximum value in the LUT for cos(φ) and sin(φ) of esums.
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).
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 $GeV\times10$].
pt2	esums E_{T} values [from LUT, in $GeV\times10$].
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.

Table 23: Explanation of Listing 9

Item	Explanation
`condition_o`	output of condition (routed to Algorithms logic, see 2.4.11).

2.4.9.2.6 Calo Calo Overlap Remover Correlation condition module

The Calo Calo Overlap Remover Correlation conditions consits of a Calo Calo Correlation condition with "Double objects requirement condition" for calo objects of same type and same bunch-crossing (2.4.9.2.1) and a single condition for a different calo object type. There has to be at least one correlation cut or the objects of "Double objects requirement condition" and a correlation cut for overlap removal between objects of "Double objects requirement condition" and objects of the different calo object type. Overlap Remover Correlation conditions `calo_calo_calo_correlation_orm_condition.vhd` are implemented only for calo object types.

2.4.10 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.

2.4.11 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.

3 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)

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

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

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