Copyright 2014 Society of Photo-Optical Instrumentation Engineers.
This paper was published in Proceedings of SPIE (Proc. SPIE Vol. 9290, 929025, DOI: http://dx.doi.org/10.1117/12.2073380) and is made available as an electronic reprint (preprint) with permission of SPIE. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper are prohibited.

FPGA Implementation of Overlap MTF Trigger - preliminary study

Wojciech M. Zabołotny[a,b], Dominik Bartkiewicz[b], Michał Bluj[c], Karol Buńkowski[b], Adrian Byszuk[a]
, Krzysztof Doroba[b], Maciej Górski[c], Artur Kalinowski[b], Krzysztof Kierzkowski[b], Marcin Konecki[b],
Jan Królikowski[b], Wojciech Okliński[b], Michał Olszewski[b], Krzysztof Poźniak[a,b]

[a]Institute of Electronic Systems, Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland

[b]Institute of Experimental Physics, Faculty of Physics, University of Warsaw, ul. Hoża 69, 00-681 Warszawa, Poland

[c]National Centre for Nuclear Research, ul. Andrzeja Sołtana 7, 05-400 Otwock, Świerk, Poland, ul. Hoża 69, 00-681 Warszawa, Poland

ABSTRACT

The paper presents the concept of the Overlap Muon Track Finder (OTF) trigger for the CMS experiment in CERN as a system implemented in the modern FPGA device. The parametrized description of the complex data processing system, allowing further optimization by iterative simulations and recompilations, is presented. Problems associated with synthesis of such complex systems with currently available synthesis tools, and their workarounds are described.

Keywords: FPGA, CERN, CMS, Level-1 trigger, muon trigger, OMTF, Overlap muon track finder

1. INTRODUCTION

The[*] Compact Muon Solenoid (CMS)[3] is a general purpose experiment for physics discoveries at the highest luminosities of the CERN Large Hadron Collider (LHC). The LHC machine delivered about $30\,\mathrm{fb}^{-1}$ of proton-proton data at a center-of-mass energy $\sqrt{s} = 7\,\mathrm{TeV}$ in 2010 and 2011, and at $\sqrt{s} = 8\,\mathrm{TeV}$ until the end of 2012, when proton-proton collisions in LHC Run-1 ended. The CMS experiment had been successfully operated during that time. The collected data allowed physicists to perform a variety of measurements and searches, and led to the discovery of a Higgs particle.[4]

The layout of the CMS detector is shown in Figure 1. In the outer part of the CMS detector, outside the coil, a muon system is placed. It is dedicated for muon reconstruction and identification. The muon system is equipped with gaseous detectors: Drift Tubes (DTs) in the barrel, Cathode Strip Chambers (CSCs) in the endcaps and Resistive Plate Chambers (RPCs) in both barrel and endcaps. The pseudorapidity[†] coverage of the muon system extends to 2.4, but in the case of the RPC system it is restricted to $|\eta| < 1.6$.

The CMS experiment has a two-layer triggering system designed to reduce a 40 MHz LHC input rate down to a few hundreds events per second suitable for storage and offline analyses. The Level-1 Trigger[5] is the first layer. It is built by using custom-made, partially programmable hardware devices (special-purpose ASICs and FPGAs). It analyses coarsely segmented data only from the calorimeter and muon systems. The Level-1 Trigger selection is based on inclusive single- and multi-object triggers with a threshold relying on estimated (transverse) momenta and energies. It reduces the event rate below 100 kHz, which is the maximal input rate capable to be handled by the second triggering layer: the High-Level Trigger (HLT) system. The HLT is implemented in the computer farm, consisting of commercial CPU processors. The HLT is further reducing the output rate to match available bandwidth, storage and computer resources available for prompt event reconstructions.

Further author information: (Send correspondence to W.M.Z.)
W.M.Z.: E-mail: wzab@ise.pw.edu.pl, Telephone: +48 22 234 7717

[*]The compact overview of the CMS detector, highlights of physics results, the RPC trigger system and perspective of CMS upgrades are given in Ref. 1, Ref. 2 and the DSc Thesis of Marcin Konecki. The first sections of this paper are based on these materials.

[†]Pseudorapidity $\eta = -\ln\tan\theta/2$, where θ is the polar angle, measured from beam axis (see Figure 1).

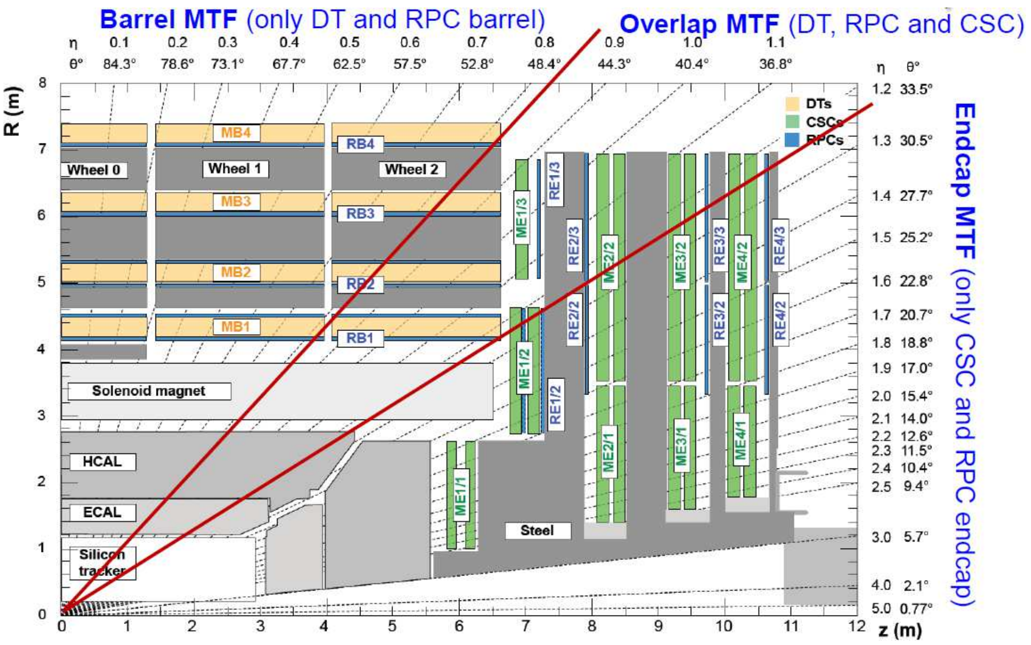


Figure 1. The longitudinal, schematic view of the quarter of CMS detector. The muon detectors RPC, CSC and DT are marked. In addition the approximate segmentation into barrel, overlap and endcap partitions of the new Muon Track Finder (discussed in Section 2) is shown.

Muon reconstruction at Level-1 trigger in LHC Run-1 is performed by dedicated sub-triggers. The DT Track Finder (DTTF) and CSC Track Finder (CSCTF) reconstruct trigger candidates using pre-processed track segments delivered by DT and CSC on-detector electronics. The Pattern Comparator Trigger (PACT) reconstructs candidates directly from pattern of hits[‡]. delivered by RPC chambers. Sub-trigger candidates are combined by the Global Muon Trigger (GMT), and the resulting Level-1 muon trigger candidate is used by algorithms implemented in the CMS Level-1 Global Trigger.

2. MUON TRIGGER UPGRADE

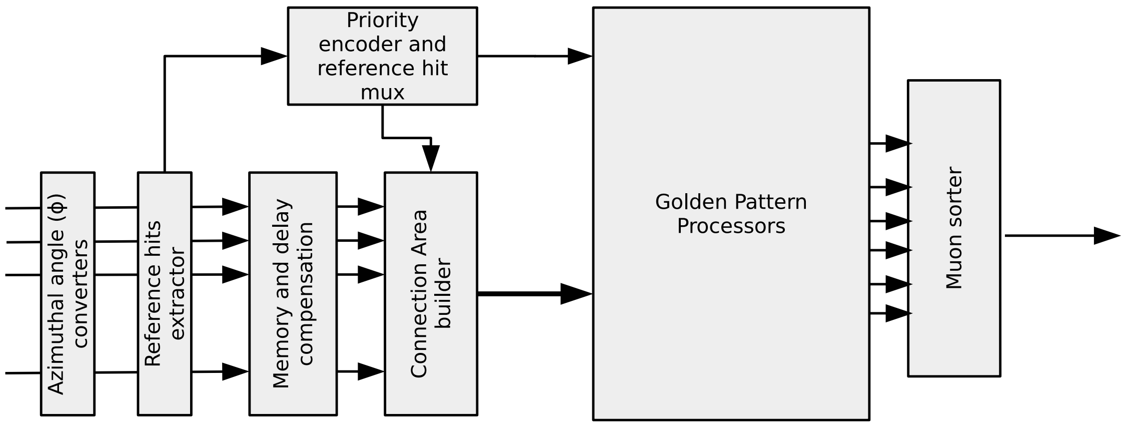
The LHC Run-1 ended in the beginning of 2013 and LHC entered Long Shutdown I (LS1). During the period of LS1 the LHC is improving the magnet interconnections. These improvements and the dipole magnet training program will allow LHC to provide proton-proton collisions at an energy of $\sqrt{s} = 13\,\text{TeV}$. Until the end of LHC Phase-1, planned for 2022, LHC will deliver over 300 fb^{-1} of proton-proton data. The instantaneous luminosity may reach values close to $2 \cdot 10^{34}\,\text{cm}^{-2}\text{s}^{-1}$.

The modernizations of the LHC machine will be accompanied by necessary upgrades[6] of the CMS apparatus to deal with more difficult experimental conditions, radiation, detector aging, but also to improve the detector performance. The increase in accelerator energy and luminosity impose high purity triggering already at Level-1. To improve purity, the Level-1 system will be rebuilt with a modern electronics[7] to enable more advanced algorithm execution.

Within the new system the muon trigger, will undergo major logic changes. In the new system the legacy PACT, DTTF and CSCTF sub-triggers and the GMT will be replaced by a new muon trigger, called the Muon Track Finder (MTF). Within each MTF module the locally available underlying detector signals will be delivered to one data processing unit and the resulting muon candidate will take into account all the available information from RPC, DT and CSC.

Such an approach is especially important in the barrel-endcap transition region where number of chamber layers from each individual sub-system is limited, thus where signals from all three types of muon detectors matters. Since the detector layout is changing with pseudorapidity the system will be partitioned into barrel-, overlap- and endcap-part, enabling different algorithm optimization depending on the detector region (see Figure 1).

[‡]Hit is a signal caused by muon crossing the given detector.



Priority encoder and reference hit mux
Azimuthal angle (ϕ) converters
Reference hits extractor
Memory and delay compensation
Connection Area builder
Golden Pattern Processors
Muon sorter

Figure 2. General structure of the OMTF Processor (OMTFP).

3. THE OVERLAP MUON TRACK FINDER

Following the MTF design, the overall strategy for its overlap part, named Overlap Muon Track Finder (OMTF), is to use available information from the RPC, DT and CSC muon detectors as the input of algorithm to combine them in a optimal way. As a consequence of a larger (than in case of each sub-systems) number of contributing measurement points one may expect improved muon transverse momentum p_T resolution in the new muon Level-1 trigger. It is especially important since a significant contribution to the rate of muon triggers in legacy system is found to originate from events in which the p_T of the Level-1 muon is over-estimated, promoting the low p_T muons above the Level-1 Trigger treshold.

In the overlap we face complicated geometry and magnetic field which, taking into account limited hardware resources, prevents us from using propagation method to build track candidates with CSC, RPC and DT measurements, in a similar way to legacy CSCTF and DTTF sub-triggers. The pattern based track building, in a way similar to PACT design cannot be used as well, due to huge number of expected hit patterns.

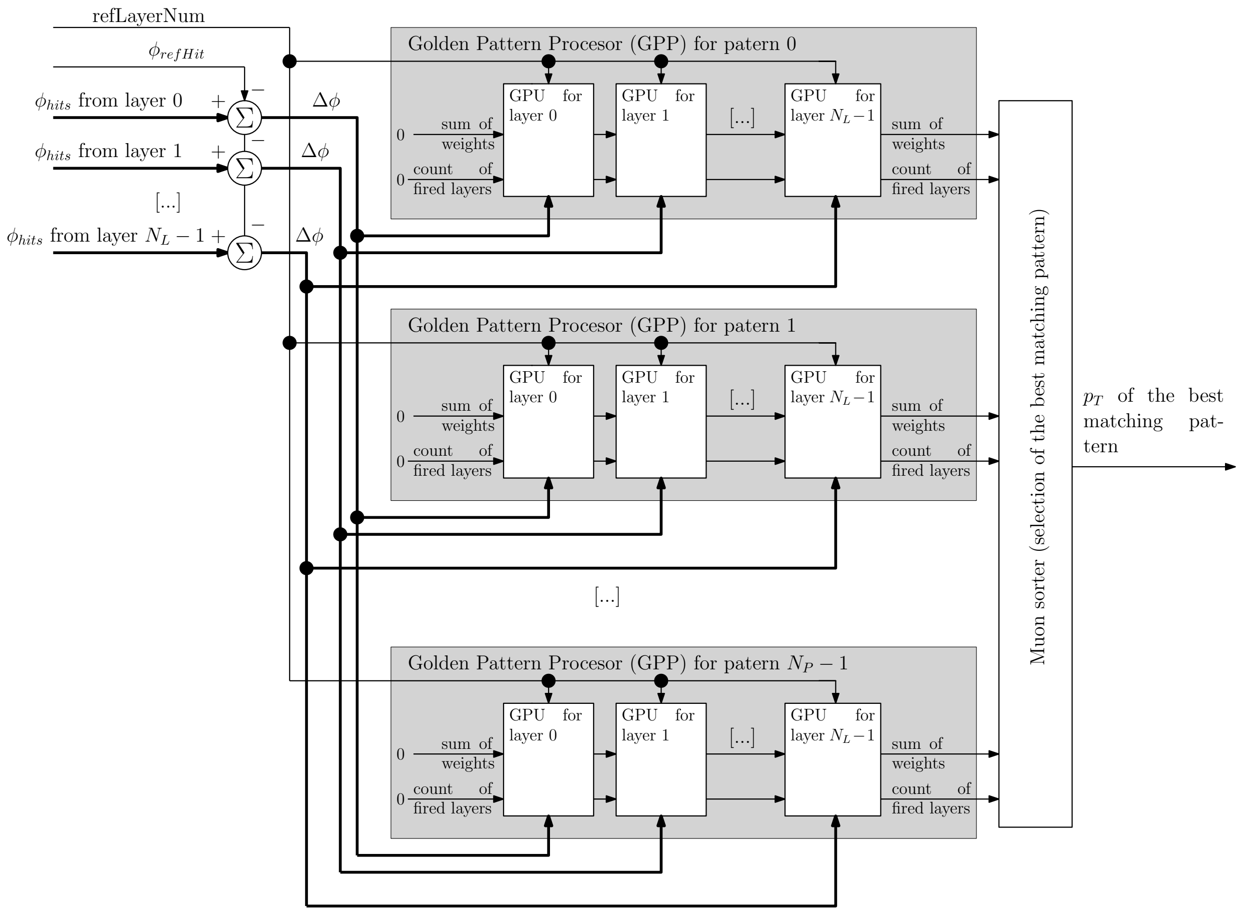
Thus as a baseline we consider complex-pattern, algorithm, called Golden Pattern algorithm. It assumes that each set of hits in the event is compared with predefined Golden Patterns. The Golden Pattern itself is defined for an “average track” of a given p_T. Distribution of muon positions in each layer provides the probability density functions (pdf) associated to the Golden Pattern. For each place in the detector there is a limited number of Golden Patterns, defined by the Level-1 p_T scale and possible choice of parameters internal to the algorithm. A comparison of measurements with all Golden Pattern, will select the best one, providing muon momentum measurement. The best Golden Pattern selection can be made on the basis of matched pdfs and number of contributing layers.

Since a set of Golden Patterns is associated with the particular place in a detector, a number of patters in the full overlap coverage may became significant, preventing to implement algorithms in todays FPGAs. A solution proposed to cope with a hardware limitation is to explore the cylindrical symmetry of the detector. In this approach each Golden Pattern can be implemented in terms of position changes between the layers rather than in terms of absolute position measurements.

4. THE OVERLAP MTF ALGORITHM

To utilize the cylindrical symmetry, all hits transmitted from the subdetectors must be converted first to their azimuthal angle ϕ. After that conversion, the hits data from each overlap regions will be processed be six OMTF processors (OMTFP) (there are two overlap regions). Each OMTFP covers the 70°range of ϕ (assuming 10°overlap between processors). The general structure of the OMTF processor is shown in Figure 2.

The first task of the OMTFP is to check, whether the input data contain any potentially interesting hits. Some of the layers of the detector, which provide the most valuable measurements for the p_T assignment, are selected as so called “reference layers” (current implementation accepts up to 8 “reference layers”). Hits in these layers, in range of angles



refLayerNum
ϕ_{refHit}
ϕ_{hits} from layer 0
ϕ_{hits} from layer 1
ϕ_{hits} from layer $N_L - 1$
$\Delta\phi$
Golden Pattern Procesor (GPP) for patern 0
Golden Pattern Procesor (GPP) for patern 1
Golden Pattern Procesor (GPP) for patern $N_P - 1$
GPU for layer 0
GPU for layer 1
GPU for layer N_L-1
sum of weights
count of fired layers
Muon sorter (selection of the best matching pattern)
p_T of the best matching pattern

Figure 3. Structure of the parallel system consisting of multiple Golden Pattern Processors.

covered by the particular OMTFP are called “reference hits” (up to 80 hits may be selected in the current implementation). To ensure the best trigger performance, the “reference hits” are prioritized basing on their positions, using Monte Carlo (MC) simulations. Depending on the ϕ of the “reference hit”, different set of OMTFP inputs must be scanned for hits potentially belonging to the muon track associated with that “reference hit”. These inputs create so called “Connection Area”. The data belonging to the particular “Connection Area” are finally transmitted to the Golden Pattern Units, which calculate the overall value called “sum of weights”, describing how well the input data matches the Golden Pattern.

The incoming data are first preprocessed. The azimuthal angle of the “reference hit” ϕ_{refHit} is subtracted from the hit angle ϕ_{hit}, creating the relative angle $\Delta\phi$. The data are then delivered in parallel to the Golden Pattern Processors (GPPs) (each associated with one Golden Pattern), and data originating from consecutive layers are processed in sequentially connected Golden Pattern Units (GPUs), as shown in Figure 3. The structure of the Golden Pattern Unit processing data for a single layer is shown in Figure 4.

For each layer, from the inputs belonging to the “Connection Area” in the particular layer, the hit best matching the given Golden Pattern is selected. The “best matching hit” is the one with ϕ which is the nearest to the $\Delta\phi_{mean}$ of the Golden Pattern for that particular layer. The $\Delta\phi_{mean}$ is the angular position difference between the center of muon position distributions in the given layer and in the reference layer. Thus the best matching hit selected is the closest one to the center of the position distribution in that layer. As the track of a given p_{T}, defining the particular Golden Pattern is bent, the $\Delta\phi_{mean}$ depends both on the current layer number, and on the number of layer in which the “reference hit” was triggered (the “reference layer”). Therefore this value is read from an array defined for the particular GPU, indexed with the number of the “reference layer”. The difference $\Delta\phi - \Delta\phi_{mean} = \phi_{dist}$ is called a “distance in ϕ”. Matching of the particular Golden Pattern with the input data is described with the sum of “weights” calculated for each detector layer, and then summed

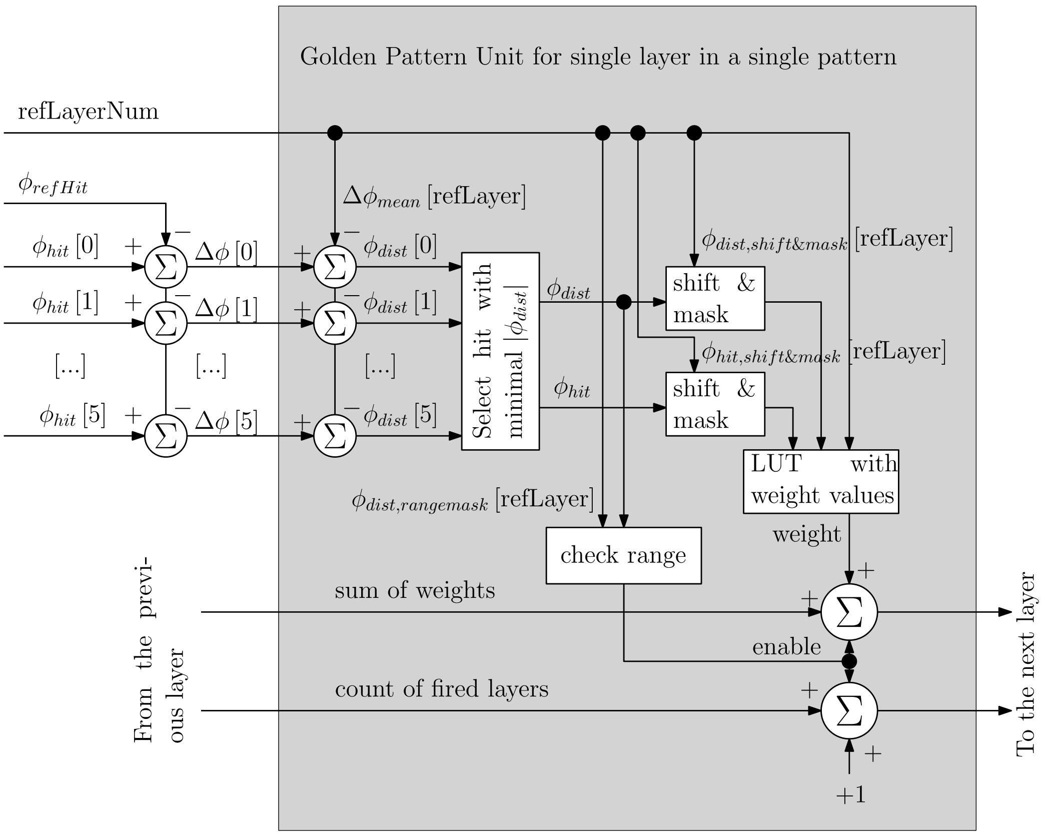


Figure 4. Implementation of the Golden Pattern Unit, processing single layer inputs for a single Golden Pattern. To better describe the algorithm, the figure shows also the preprocessing step.

together. The value of the "weight" for the particular layer is a function of the ϕ_{dist}, more significant bits of the ϕ_{hit} (which allows compensating of minor deviations from the cylindrical symmetry of the CMS detector) and the number of the "reference layer". To maximally speed up calculation of this function, its values are stored in the look-up table, addressed with the word created from listed function arguments. The contents of the weight look-up tables is generated on the basis of the probability density function values (described in Section 3), obtained from Monte Carlo simulation. As the width of distribution used in Golden Patterns pdfs depends on the p_{T} value (patterns for higher p_{T} values are narrower), it is possible to select which bits of the ϕ_{dist} and ϕ_{hit} are used to address the LUT. Such a selection is necessary due to limited width of the LUT address. Of course if we are using only less significant bits of the mentioned values, it is necessary to check, whether the more significant bits are equal to zero (the "check range" block in Figure 4). For each pattern the sum of weights from all layers is calculated, and also the layers in which matching hits were found ("fired layers") are counted. The resulting values are delivered to the muon sorter (see Figure3), which selects the pattern for which the highest number of fired layers and the highest total "sum of weights" was achieved.

5. IMPLEMENTATION OF THE OMTF TRIGGER ALGORITHM IN FPGA

The first step in the implementation of the OMTF trigger algorithm is the conversion of the data delivered by the particular subdetector to the format acceptable for the OMTFP. This process may be best described with an example of the processing of the RPC data, as this is probably the most complex case.

5.1 Conversion of strip clusters into angle coordinates for the RPC detector

Conversion of strip clusters into angle coordinates is done in parallel for all RPC chambers connected to the overlap region. Information about active strips is transmitted to the OMTF system. The general scheme of connections of hit strips to angles

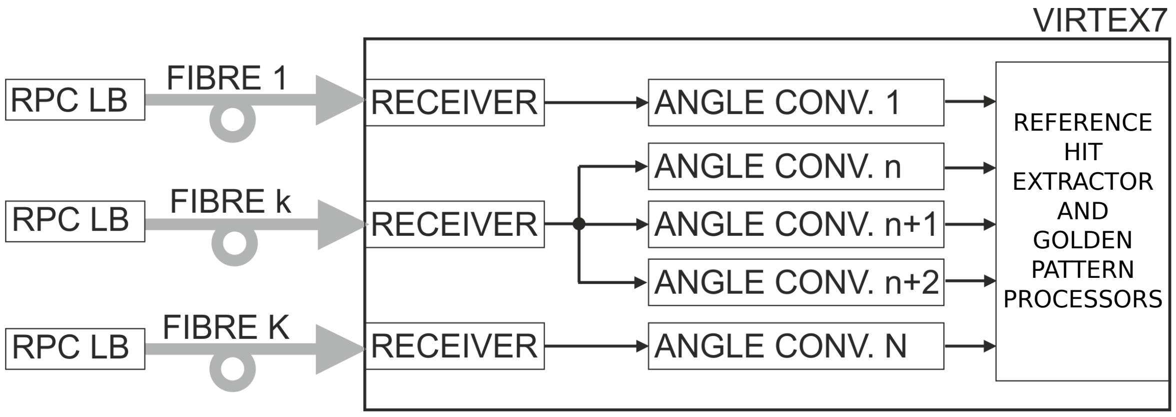


Figure 5. General scheme of conversion of active strips into angles for single trigger board.

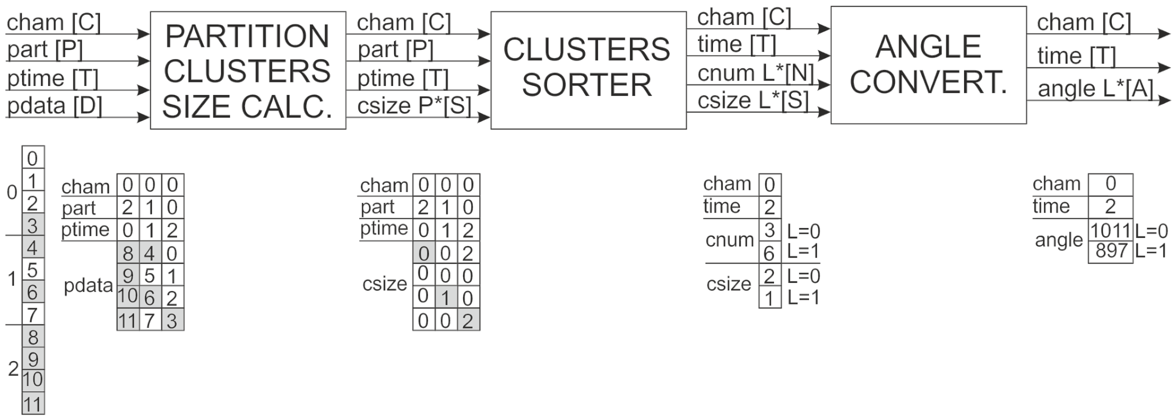


Figure 6. Structure, and principle of operation of the converter.

converters is shown in Figure 5. To the particular OMTFP (analyzing the corresponding area of the CMS detector), the required number of optical links (denoted as K) is connected. Synchronization of the data stream, and its deserialization is accomplished in dedicated blocks of gigabit receivers (SERDES). During that the stream of compressed data, transmitted from RPC chambers via LINK BOX blocks (RPC LB) is recovered.[8] Connection of angle converters to the streams of compressed data is not uniform. Number of independent angle converters depends on number of pseudorapidity ranges, covered by a single chamber (even up to 3 ranges in the endcap area). Due to this fact, each angle converter extracts from the compressed data stream only information associated with the analyzed pseudorapidity range. Structure and principle of operation of the angle converter is shown in Figure 6. The upper part of the figure shows the block diagram of the converter. The block PARTITION CLUSTERS SIZE CALC detects all strip clusters for consecutive partitions of the compressed stream (however not bigger, than the defined maximum size of the cluster). The block CLUSTERS SORTER selects the biggest clusters and outputs their base strip numbers. The block ANGLE CONVERT converts the center position of sorted clusters into angle values.

The converter is implemented as a parametrized block in VHDL language. The following parameters are defined (default values are given):

C=2: number of bits of the chumber number transmitted via optical link from LB

P=4: number of bits of the partition number of the the chamber data transmitted via optical link from LB

T=3: number of bits of delay of the the chamber data transmitted via optical link from LB

D=8: number of bits of data from single partition transmitted via optical link from LB

S=2: number of bits of width of a single cluster

L=2: number of biggest clusters after sorting

M=3: maximum width of a single cluster

N=2: number of bits of the width of a single cluster (associated with the M value)

A=10: number of bits of the width of a single angle value in the relative format

In the lower part of the figure, there is a simplified example of conversion of active strips to the angles for a single 12-strip RPC chamber (cham=0), divided into partitions containing 4 strips per partition. The active strips are marked as gray. The compressed data contain sequentially send partitions with the appropriate delay time (ptime=0,1,2) and partition number (part=2,1,0). In the first block there are three clusters detected, however one of them is 4 strips wide, and exceeds the allowed size (M=3 strips). As a result, its size is set to 0. On the output of the sorter block there are selected two (L=2) widest clusters, together with the numbers of their base strips and common delay time (time=3). In the pipelined process the angles values are calculated (the latency in this process is constant, so the "time" value does not change). The analysis takes into consideration the chamber number (cham).

5.2 Implementation of the data preprocessing before the Golden Pattern Processors

The next stage is the preprocessing of data before they enter the set of GPP units. This step is performed in the same way for signals originating from different subdetectors. The hit data transmitted in each input channel are scanned (regarding their ϕ values) whether they match the "reference hits" definitions. In case of match, the corresponding bit in the "reference hits" vector is set. The "reference hits" vector is then delivered to the priority encoder, which outputs the position of the highest priority set bit. Unless the new "reference hits" vector is loaded, in the next clock period, the priority encoder outputs the position of the next highest priority set bit (after the previous one is cleared). This allows to process the active "reference hits" in order of their priority. The number of reference hits is parametrized (and currently set to 80). To ensure high clock frequency, the priority encoder is implemented as a hierarchical two-level pipelined structure. The "reference hits" definitions are based on result of MC simulations, and are declared in a dedicated VHDL record type constant, in a special VHDL package describing the structure of the OMTFP.

Basing on the number of the "reference hit" provided by the priority encoder, the Connection Area Builder selects the input channels in each layer, which are transferred in parallel to the GPP blocks. This block is a standard multiplexer with registered output. The structure of the Connection Area is also based on MC simulations and described in a dedicated VHDL record type constant in the mentioned VHDL package.

Before the data are delivered to the GPP blocks, the ϕ of the "reference hit" (ϕ_{refHit}) is subtracted from the ϕ_{hit}, creating the $\Delta\phi$, because due to cylindrical symmetry only the angle differences are meaningful. However the original ϕ_{hit} values are also delivered, as they can be used to compensate minor deviations in the cylindrical symmetry (see Section 4).

5.3 Implementation of the Golden Pattern Processors

The most complex part of the OMTFP is the block of Golden Pattern Processors. The biggest problem associated with implementation of this block is the high number of Golden Patterns (in the current implementation we assume 50 Golden Patterns) and the high number of Golden Pattern Units (one unit per Golden Pattern per a detector layer, which results in total number of 950 GPUs). It means, that implementation of the single GPU must be highly optimized regarding the FPGA resources consumption.

Another requirement for the GPP blocks is the high speed of operation. The new data are delivered to the OMTFP every bunch crossing (BX) in the LHC, which means every 25 ns (1/40 MHz). It is impossible to accomplish the algorithm described in Section 4 in such a short time. Therefore a pipelined implementation is necessary. If we are going to process multiple "reference hits" for each BX, the processing speed must be even higher. In the current implementation we intend to process up to 8 "reference hits" in a single BX, so the desired clock speed is equal to 320 MHz.

As can be seen in Figure 4, in each GPU block we must perform multiple subtractions. From the $\Delta\phi$ for each input (up to 6 inputs) a $\Delta\phi_{mean}$ for the particular layer must be subtracted. It means, that up to 5700 subtractions may be needed in all GPP blocks. Let's compare this assessment with the number of resources in the available FPGA chip.

The hardware platform for the OMTFP will be the MTF7 board,[9] which contains (in the extended version) the Xilinx Virtex 7 XC7VX690T FPGA. This chip contains the following resources:[10]

- 693,120 Logic cells
- 108,300 Slices
- 10,888 Kb of Distributed RAM
- 3600 DSP Slices
- 2940 18Kb Block RAMs
- 80 GTH transceivers

According to the description of the DSP48E1 arithmetical blocks (DSP slices) available in the Virtex 7 chips,[11] these blocks are capable to perform simultaneously two 24-bit additions/subtractions or four 12-bit additions/subtractions. Therefore the XC7VX690T chip is able to perform up to 7200 24-bit additions/subtractions or up to 14400 12-bit additions/subtractions. It means, that the MTF7 board should be able to provide necessary arithmetic resources for the OMTFP. In case if the number of Golden Pattern should be significantly increased, we can use a special approach developed to allow a single long-word adder/subtractor to perform multiple operations with shorter words.[12] Additionally, for very short words, the arithmetic operations may also be implemented in the general purpose FPGA logic resources.

Another limiting factor is the number of Block RAMs available for implementations of look-up tables. For 950 GPUs, we can theoretically use even 3 Block RAMs per GPU, but assuming that some BRAMs may also be needed for input and output data buffering, we can safely assume availability of 2 BRAMs per single GPU, which should be sufficient to implement the "weight" look-up tables. Usage of BRAMs as a look-up tables also allows some flexibility. Generally each BRAM[13] is a dual port memory which stores 1024 18-bit words. If we need to use them as a ROM, in the single BRAM we can implement two independent ROMS, storing 512 18-bit words. On the other hand, if the "weight" may be only a 9-bit value, we can use single BRAM as a memory storing 2048 9-bit words (using one address bit to select upper of lower half of the word). As we can see, available amount of BRAMs allows us to significantly increase the number of GPUs.

The structure of Golden Pattern Processors block is defined basing on results of MC simulations. Configurable items include:

- Number of inputs in each layer of each GPP (ie. for each GPU)
- $\Delta\phi_{mean}$ for each layer of each GPP (ie. for each GPU), as the function of "reference layer" number
- Number of least significant bits ignored in the ϕ_{dist} for each layer of each GPP (ie. for each GPU) when calculating the LUT address
- Number of least significant bits ignored in the ϕ_{hit} for each layer of each GPP (ie. for each GPU) when calculating the LUT address
- Number of most significant bits in the ϕ_{hit} for range checking for each layer of each GPP (ie. for each GPU)
- Contents of the "weights" look-up table for each layer of each GPP (ie. for each GPU)

The above configuration data (except of contents of the "weight" LUTs) are stored in the constant two-dimensional array of VHDL record type in the VHDL package describing the structure of the OMTFP. Contents of the LUTs must be stored in separate binary files, as explained in Section 5.5.

To allow easy modification of the implementation parameters, the OMTFP has been implemented in the VHDL in a parametrized way. The modifiable parameters are stored in two VHDL packages:

- the first one describes the constraints (e.g. number of detector layers, number of "reference hits", number of "reference layers" etc.) and the VHDL types,

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;

library work;

package golden_proc_pkg is

  -- Maximum number of inputs in a layer
  constant GP_MAX_INS_IN_LAYER : integer := 14;
  -- Maximum number of outputs per layer in the connection area
  constant GP_MAX_CONAR_OUTS    : integer := 6;
  -- Maximum number of layers
  constant GP_MAX_LAYERS       : integer := 19;
  -- Maximum number of connection areas
  constant GP_N_OF_CONARS : integer := 8;

  subtype T_GP_CONAR_NR is integer range 0 to GP_N_OF_CONARS-1;
  subtype T_GP_LAYER_NR is integer range 0 to GP_MAX_LAYERS-1;
  subtype T_GP_ENTRY_NR is integer range 0 to GP_MAX_INS_IN_LAYER-1;

  constant GP_PHI_WIDTH : integer := 10;

  subtype T_GP_HIT_PHI is signed(GP_PHI_WIDTH-1 downto 0);
  subtype T_GP_HIT_DIST_PHI is signed(GP_PHI_WIDTH-1 downto 0);

  type T_GP_HIT is record
    active : std_logic;
    phi      : T_GP_HIT_PHI;
    dist_phi : T_GP_HIT_DIST_PHI;
  end record T_GP_HIT;

  -- Definition of all hit inputs
  type T_GP_LAYER is array (0 to GP_MAX_INS_IN_LAYER-1)
          of T_GP_HIT;
  type T_GP_HITS_ARRAY is array (0 to GP_MAX_LAYERS-1)
          of T_GP_LAYER;

  -- Definition of connection areas outputs
  type T_GP_CONAR_LAYER is array (0 to GP_MAX_CONAR_OUTS-1)
          of T_GP_HIT;
  type T_GP_CONAR_OUTS is array (0 to GP_MAX_LAYERS-1)
          of T_GP_CONAR_LAYER;

  type T_GP_CONAR_LAYER_DEF is record
    first : integer range 0 to GP_MAX_INS_IN_LAYER-1;
    len   : integer range 1 to GP_MAX_INS_IN_LAYER;
  end record T_GP_CONAR_LAYER_DEF;

  -- Types for definition of all connection areas
  type T_GP_CONAR_DEF is array (0 to GP_MAX_LAYERS-1)
          of T_GP_CONAR_LAYER_DEF;
  type T_GP_ALL_CONARS_DEF is array (0 to GP_N_OF_CONARS-1)
          of T_GP_CONAR_DEF;

  [...]
end package body golden_proc_pkg;
```

Figure 7. Sample definitions of constants and types used to describe the OMTFP structure in a parametrized way. In this particular part the record types describing the Connection Areas definitions are declared. The T_GP_CONAR_LAYER_DEF type describes a single layer in the area. The T_GP_CONAR_DEF describes single Connection Area. Finally the T_GP_ALL_CONARS_DEF is a type used to store definitions of all Connection Areas. The constant of the T_GP_ALL_CONARS_DEF type is defined in the second VHDL package, and initialized with values resulting from MC simulations.

- the second contains the constant values resulting from MC simulations, describing "reference hits" definitions, Connection Areas connections, and Golden Patterns.

Such approach allows to separate description of the logic implementing the OMTF algorithm from the parameters describing e.g. number of bits in particular values or definition of bit fields. A part of the first VHDL package defining the necessary constants and types is shown in Figure 7.

5.4 Implementation of the muon sorter

The block of GPPs outputs a total "sum of weights" and number of active layers for each Golden Pattern. The last block is the muon sorter which must select the Golden Pattern which best matches the input data - i.e. which has the highest count of active layers and the highest total "sum of weights". Implementation of this operation in a block operating with clock frequency of 320 MHz required a hierarchical multilevel tree of sorters. However it appeared, that acceptable number of inputs in each level depends on the length of the word describing the total "sum of weights". As width of a single "weight" is parametrized, the width of "sum of weights" also varies, and therefore a special parametrized implementation[14] of the final sorter was used, which automatically finds the number of levels and implements the tree of sorters for given number of Golden Patterns and number of inputs in the elementary sorter.

5.5 Problems faced during the implementation

The structure of the OMTFP (definition of "reference hits", structure of Connection Areas associated with particular "reference hits", contents of the weight LUTs) depends on results of the physical simulations. Therefore it may be subject to significant changes during the development and testing. In the first attempt, it was supposed that the VHDL packages should be generated directly by the code performing the simulations. However during the development it appeared that some details of implementation must be changed to obtain required performance (eg. clock speed and resources consumption), and therefore finally another solution was chosen. In this approach the MC simulations generate configuration data in a simple intermediate format. This data are later on read by the Python scripts, which check their correctness and generate the appropriate VHDL packages. With this solution even the significant change in implementation does not involve persons responsible for physical simulations. It is only necessary to modify the Python scripts so, that they generate new VHDL packages, compatible with the new implementation.

Additionally this approach allows independent testing of implementation's feasibility for particular parameters. It is likely, that regularity found in the results of real MC simulations, will result in a structure, which may be significantly simplified during the optimizations performed by the synthesis tools. To check synthesizeability of the design for the worst case, a special set of scripts was written in the Python language, which generates random (but reasonable) configuration data. Due to lack of any regularity, successful compilation of the OMTFP with such a random data may be a strong suggestion, that it should also compile with any configuration data based on real MC simulations.

Certain problems were also found during the synthesis of the design. As described in Section 5.3, the content of the "weights" look-up tables must be defined in the configuration data. The most natural solution was to include them in the records defining structures of individual GPUs. However it appeared, that such complex record type constants are very inefficiently handled by available synthesis tools (Xilinx Vivado 2013.4), which resulted in a huge memory consumption and unacceptable compilation times. The described problem was finally successfully solved, by storing the contents of each LUT in a separate file with the name containing the number of the Golden Pattern and the number of the layer. During the synthesis, the name of each file was created using the index of current pattern and layer, and the contents of the appropriate file was used to initialize the particular BRAM. Such non-standard approach resulted in significant reduction of compilation time and allowed to successfully implement the design.

6. RESULTS

At the current stage of the development all components of the OMTFP are implemented in the VHDL, may be successfully synthesized, and may work with clock frequency of 320 MHz. The synthesis of complete OMTF for randomly generated configuration data was successful (for different random configurations) but the achievable clock frequency was below 320 MHz, due to timing problems which are not resolved yet. For 50 Golden Patterns and 19 detector layers the OMTFP fits in the XC7VX690T chip, however no integration with the input preprocessing codes was done yet. At this step resources necessary for services and signal preprocessing are not yet taken into account, however it is expected to fit to remaining resources.

7. CONCLUSIONS

The presented Overlap Muon Track Finder (OMTF) algorithm for the CMS detector has been successfully described in a synthesizable VHDL code. The resulting code is parametrized and the structure and parameters of implemented system may be easily adjusted according to the results of physical Monte Carlo simulations. This approach is especially useful for designing systems, where the final configuration must be a result of an iterative process of finding optimal structure and parameters. Parametrized implementation of such complex systems is a good opportunity to identify limitations of the synthesis tools and to elaborate viable workarounds, which may contribute both to development of the synthesis tools, and to methodology of systems description.

ACKNOWLEDGMENTS

The work on OMTF is supported by Polish NCN grant 2013/11/B/ST2/04202

REFERENCES

[1] Konecki, M., "The RPC based trigger for the CMS experiment at the LHC," *Journal of Instrumentation* **9**(07), C07002 (2014).

[2] Konecki, M., "CMS: Performance, physics, perspectives," *Acta Physica Polonica B* **45**(07), 1427–1445 (2014).

[3] CMS Collaboration, "The CMS experiment at the CERN LHC," *JINST* **3**, S08004 (2008). Also published by CERN Geneva in 2010,.

[4] CMS Collaboration, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Phys.Lett.* **B716**, 30–61 (2012).

[5] CMS Collaboration, [*CMS. The TriDAS project. Technical Design Report, Volume 1: The Trigger Systems*], no. CERN/LHCC 2000-038, CMS TDR 6.1 in Technical Design Report CMS, CERN, Geneva (2000). http://cds.cern.ch/record/706847.

[6] CMS Collaboration, "Technical proposal for the upgrade of the CMS detector through 2020." CERN report: CERN-LHCC-2011-006 ; CMS-UG-TP-1 ; LHCC-P-004 available at `https://cds.cern.ch/record/1355706` (June 2011). [Online; accessed 29-June-2014].

[7] CMS Collaboration, [*CMS Technical Design Report for the Level-1 Trigger Upgrade*], no. CERN-LHCC-2013-011. CMS-TDR-12 in Technical Design Report CMS (Jun 2013).

[8] Gorski, M., Kudla, I., and Pozniak, K., "Resistive plate chamber (RPC) based muon trigger system for the CMS experiment - data compression/decompression system," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **419**(2-3), 701–706 (1998).

[9] Acosta, D. et al., "The CMS modular track finder boards, MTF6 and MTF7," *JINST* **8**(12), C12034 (2013).

[10] "7 Series FPGAs Overview." `http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf` (February 2014). [Online; accessed 29-June-2014].

[11] "7 Series DSP48E1 Slice User Guide." `http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf` (May 2014). [Online; accessed 29-June-2014].

[12] Zabolotny, W., "Generator of VHDL code for parallel adder (using long-word hardware adders to perform multiple additions in parallel)." Online post to alt.sources, also available at `http://ftp.funet.fi/pub/archive/alt.sources/2818.gz` (November 2013). [Online; accessed 29-June-2014].

[13] "7 Series FPGAs Memory Resources User Guide." `http://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf` (May 2014). [Online; accessed 29-June-2014].

[14] Zabolotny, W., "VHDL sources of the parametrized pipelined block for finding the maximum value." Online post to alt.sources, also available at `http://ftp.funet.fi/pub/archive/alt.sources/2851.gz` (June 2014). [Online; accessed 29-June-2014].