A Highly Parallel FPGA Implementation of a 2D-Clustering Algorithm for the ATLAS Fast TracKer (FTK) Processor

N. Kimura, A. Annovi, M. Beretta, M. Gatta, S. Gkaitatzis, T. Iizawa, K. Kordas, T. Korikawa, S. Nikolaidis, C. Petridou, C-L. Sotiropoulou, K. Yorita and G. Volpi

***Abstract*– The highly parallel 2D-clustering FPGA implementation used for the input system of the Fast TracKer (FTK) processor for the ATLAS experiment of the Large Hadron Collider (LHC) at CERN is presented. The LHC after the 2013-2014 shutdown periods is planned to have increased luminosity, which will make it more difficult to have efficient online selection of rare events due to the increase of the overlapping collisions. FTK is a highly-parallelized hardware system that improves the online selection by executing real time track finding using the information from the silicon inner detector. The FTK system requires fast and robust clustering of the hits retrieved from the silicon detector on FPGA devices. We show the development of the original input boards and the implemented clustering algorithm. For the complicated 2D-clustering, a moving window technique is used to minimize the use of FPGA resources. The combination of custom developed boards and implementation of the clustering algorithm provides sufficient processing power to meet the specifications for the silicon inner detector of ATLAS up to the maximum LHC luminosity planned until 2022. The developed algorithm is easily adjustable to other image processing applications that require real-time 2D-clustering.**

ATL-DAQ-PROC-2014-004
04/07/2014

I. Introduction

Online selection of interesting events in the ATLAS experiment [2] is a very challenging task. After the 2013-2014 shutdown periods, the LHC will run with energy of 13 or 14 TeV and with instantaneous luminosities which could exceed 10^{34} cm^{-2} s^{-1}, with a bunch crossing period of 25 ns. It is expected that the pileup will reach 60 or more proton-proton collisions, which will render more difficult to have efficient online selection of rare events. The LHC experiments will need to adapt to the more crowded events, maintaining the physics output and the quality of the final results. FTK [1] is an approved ATLAS trigger upgrade project, and it is developed to reconstruct tracks with transverse momentum above 1 GeV for any event accepted by level 1 trigger of up to 100 kHz. The system is specialized in performing tracking through custom and commercial electronics, using a multistage parallel algorithm. The fast full tracking of FTK makes possible new trigger selections, which are robust against pile-up. A 2D clustering algorithm is implemented on the FTK input mezzanine board (FTK IM) that receives the data from the ATLAS silicon inner track detector. The ATLA silicon Inner Detector [3] consists of Pixel modules (344x128 pixels) [4] (Fig. 1) and Strip modules (768 micro-strips) [5].

Manuscript received May 22, 2014. This work was supported from Istituto Nazionale di Fisica Nucleare; Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science and MEXT, Japan; and the European community FP7 People grants ARTLHCFE 254410 FP7-PEOPLE-2009-IOF and FTK 324318 FP7-PEOPLE-2012-IAPP.

Naoki Kimura, Tomoya Iizawa, Tomohiro korikawa and Kohei Yorita are with the Department of Physis of the Waseda University, 3-4-1 Okubo Shinjuku Tokyo, Japan

S. Gkaitatzis, Calliope-Louisa Sotiropoulou, Kostas Kordas, Spiridon Nikolaidis and Chara Petridou are with the Department of Physics of the Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece (emails: lsoti@physics.auth.gr, snikolaid@physics.auth.gr).

Alberto Annovi, Matteo Beretta, Maurizio Gatta and Guido Volpi are with the INFN National
Laboratory, Italy (emails: alberto.annovi@cern.ch, matteo.beretta@lnf.infn.it, guido.volpi@pi.infn.it).

II. FTK IM

The FTK IM is the most upstream input mezzanine board of the FTK system. The board has 12 PCB layers, and it is equipped with two Xilinx Spartan 6 LX150T FPGAs [6], 4 S-LINK receiver modules, which will receive the input from the ATLAS inner track detector, a memory chip to store the pseudo data for test purposes, a FMC connector to the Data Formatter [1] motherboard and other basic parts (Fig. 2). The data from the ATLAS inner track detector Read Out Drivers (RODs) are received by more than 300 S-LINKs [7]

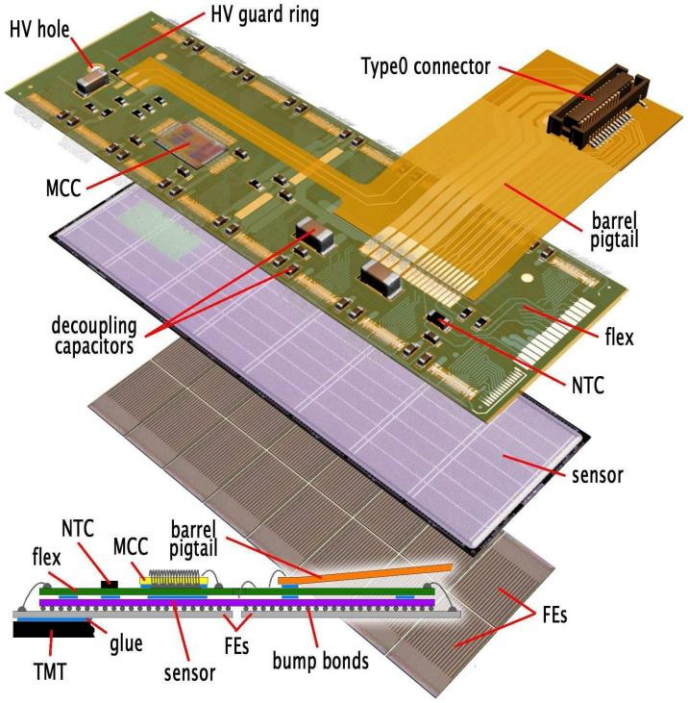

Fig. 1. ATLAS Pixel Module [4]

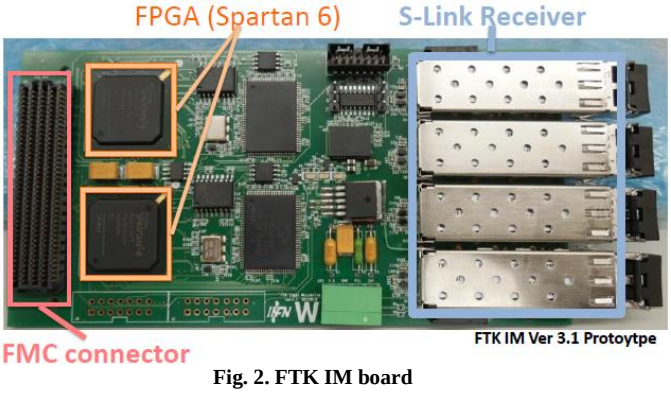

Fig. 2. FTK IM board

running at 2.0 Gbps. One FPGA on the FTK IM receives data from 2 RODs. Data from each ROD arrives as 32 bit words at a maximum 40 MHz after the S-LINK decoding. At most 1 Pixel ROD is connected to each FPGA for efficient usage of FPGA resources. Finally, the FTK IM sends the clustered hit data to the next board by using 16 DDR lines working at 200 MHz, allowing a data throughput of 32 bits words at 50 MHz. Both input and output speeds are confirmed by board testing.

III. CLUSTERING

Clustering signifies identification of the group of contiguous hits in the data from inner track detector. Data from Strip modules are 1D hits position that arrives almost sorted by position, so clustering algorithm is relatively simple. Therefore this paper focuses on the complicated 2D Pixel clustering implementation, and the common logic functions shared among Strip and Pixel clustering. The data from the Pixel modules is received by S-LINK receiver and data is de-serialized to 32 bits data and forwarded to input FIFO. Then event information as exemplified by event headers/trailers is converted to FTK format and only pixel hit information are sent to clustering part.

The clustering implementation is designed in three separate processing modules: a) the hit decoder module, b) the grid clustering module and c) the centroid calculation module.

A. Hit Decoder Module

The hit decoder transforms the incoming data from the ATLAS format to a format useful to the following processing step, the grid clustering module. It is a pre-processing step that selects, formats and organizes the information that is used by the clustering algorithm such as start /end event words (the flag words that mark the beginning and the end of an event), module headers/trailers (the flag words that mark the beginning and the end of this from one pixel module as well as the module number) and of course the pixel hits. The code is robust against bit errors in the input data. In the rare case when the arriving hit data are not identified by a start event word or a module header the data are dropped. In addition the hit decoder can reintroduce missing control word such as end event words and module trailers in order to guarantee a valid data stream to the next modules.

The most important role of the hit decoder module is to properly align all the incoming data. The ATLAS pixel module's 16 front-end (FE) chips are arranged as a 2x8 grid on the module surface and they are numbered in anti-clockwise. The hits data are readout in the same FE sequence. This means that half of the pixel module data arrive in reverse column order than the other half. The hit decoder module needs to restore the order of the hits since the clustering algorithm is based on the assumption that hits are ordered by increasing column number sequence.

To achieve a Last In First Out (LIFO) is used to store all the hits that arrive from read out chips with number from 0 up to 7. When a hit arrives from a read out chips with number from 8 up to 15 it is stored in a separate register. The value of the register is compared with the last value stored in the LIFO and the hit with the smallest column value is propagated to the next processing module. In this way increasing column sequence is restored. Two small FIFOs are added as input and output buffering for synchronization purposes.

B. Grid Clustering Module

The grid clustering module is the one that actually identifies the clusters and it is the most computationally intensive block of the implementation. The module uses a "moving window" technique to minimize computational time per cluster identification as well as needed FPGA resources. The "window" is actually a rectangular grid of pixel cells of generic size. Its size depends on the maximum expected cluster size per application and it must be big enough to fit this cluster size. The "window" is "moving" in the sense that during the several passes of the cluster identification process it is virtually placed in different positions of the pixel module and every time it is filled with data from different areas of the pixel module plane.

On the starting of a module processing it is filled with data around the first received hit. This hit is used as a reference hit and it is placed on the middle row of either column 0 or column 1 of the window. The two alignment options are required because of the double column scrambling of the data. A first hit from an odd column in the pixel module is placed in column 1 inside the clustering grid in order to allow for one column space for hits from the previous (even) column in case they arrive later. The hits are read from the input until the first hit with a column beyond the column range spanned by the "window" arrives. This hit is kept in the input FIFO and processed later. At this point, all the hits that belong to the "window" are loaded to the grid, while the hits that do not belong to the window but are within the window column span (above or below it) are stored in a separate circular buffer. The cluster identification process begins by selecting two grid pixel cells as "seeds" (column 0 and column 1 on the middle row) (blue colored cells – Fig. 3, a). The "seed" cells that contain a hit when selected change their state to "selected". The "selected" state is propagated on the next clock cycle to all neighboring hits (arrow – Fig. 3, b and c). On the same cycle a hit that was previously selected is now read out (black colored cells – Fig. 3, c and d).

When a hit is read out the cell returns to an "empty" state (grey colored cell – Fig. 3, d) using the same process all the

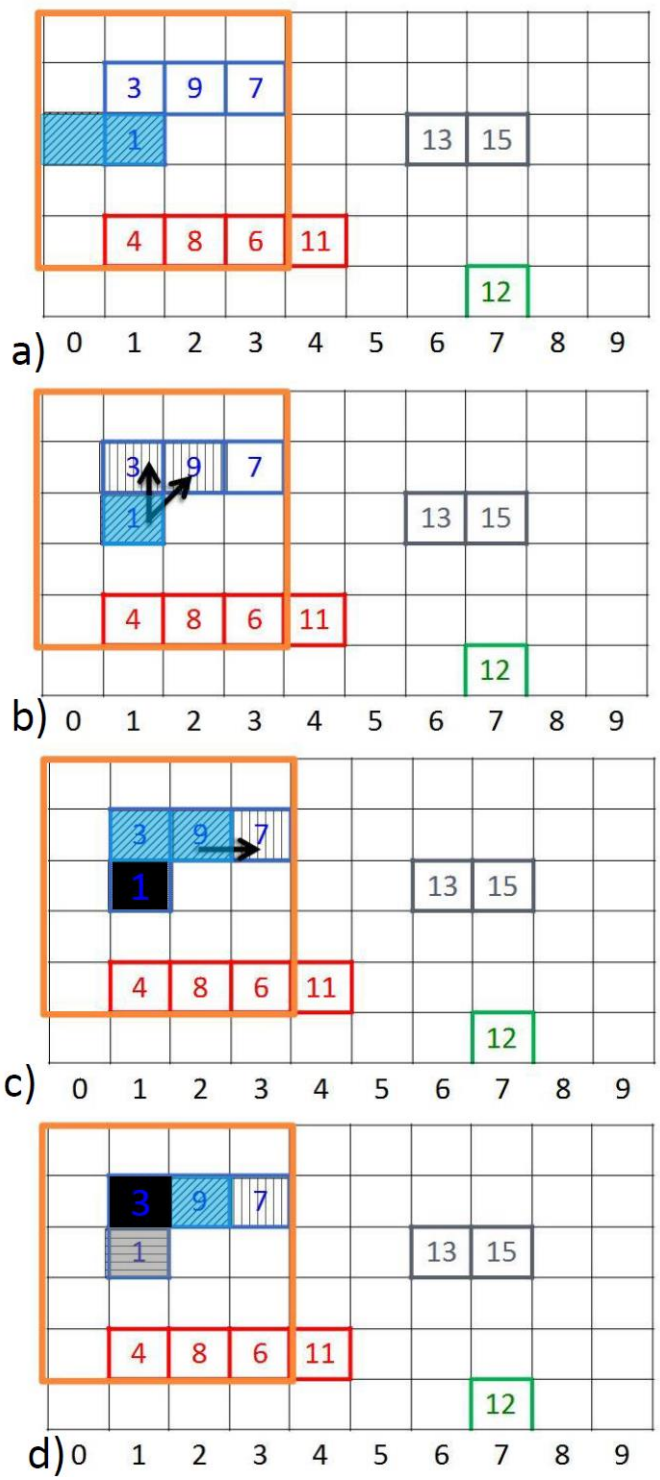

Fig. 3. Cluster Readout Process

hits that form a cluster are read out. The hit information that is read out of the grid is propagated to the next processing module in its relative coordinates with respect to the reference hit. After the cluster hits are all read out, a cluster flag word is sent to the next module that contains the absolute coordinates of the reference hit. The hits that remain in the grid and that do not belong to the identified cluster are also read out in the next processing step and they are saved in the circular buffer in their absolute coordinates. The hits that are recovered from the grid to be stored in the circular buffer are not in column sequence with the previous hits of the circular buffer.

On the next run of the clustering module the grid is loaded with hits from the circular buffer. The leftmost hit stored in the circular buffer is chosen as a new reference hit. This hit value is stored in a separate register, called the "leftmost register", as the circular buffer is being filled. While reading from the circular buffer to load the grid, hits that do not belong to the grid need to be saved again in the circular buffer. Extra functionality had to be added to the circular buffer to control simultaneous reading and writing of hits without accessing twice the same data. If after reading the circular buffer there are hits in the input FIFO that belong to the columns of the circular buffer these hits are read until a hit with a column number outside the grid arrives at the input. The clusters are identified using the same process used on the first run. A clustering module process all the data related to a pixel detector module, so the cycle is repeated until a pixel detector module trailer word is received from the clustering input and the circular buffer is empty. For the current clustering module implementation a "window" of 8x21 pixels is used (8 for the *z* or η direction and 21 for the *r-Φ*). The most common cluster size in the ATLAS pixel module is of 2x3 pixels. The bigger grid is used to allow identification of the rarer but still existing bigger clusters, or clusters generated by merging hits from two or more clusters. Clusters of bigger size than the grid size, which means clusters extending from the reference hit beyond one of the grid edges, will be split. Clusters that touch a grid edge will be identified by a flag in the output.

The algorithm is executed in a pipeline, which means that clusters are identified and read out from the clustering module and then processed by the centroid module simultaneously. Different numbers of clustering modules can be implemented at the output of the hit decoder to identify clusters in parallel. These modules will work independently on different pixel module data. Therefore, the implementation offers the versatility to choose the best performance over FPGA resource tradeoff.

C. Centroid Calculation Module

The centroid calculation module is the post-processing step in the clustering implementation that performs the data reduction process, and it is currently under development. It is the module where the cluster data is replaced with one set of coordinates, the centroid coordinates. For each cluster a centroid value is calculated. The centroid is then corrected by a variable calculated by taking into account the absolute pixel position in the detector as well as the charge deposition in each cluster measured by the Time-over-Threshold (ToT) information as measured from the FE. The ToT value for each hit is stored in the same word as the hit coordinates and while the hits are placed in the clustering window of the grid clustering module these values are stored in a separate memory (ToT memory) and are recovered while the cluster hits are read out.

One fundamental characteristic of the 2D clustering implementation is that different clustering engine can work independently and in parallel to identify different clusters, therefore increasing performance while exploiting more FPGA resources. However, the pixel data are received through S-LINK as a single data stream and the processing units that

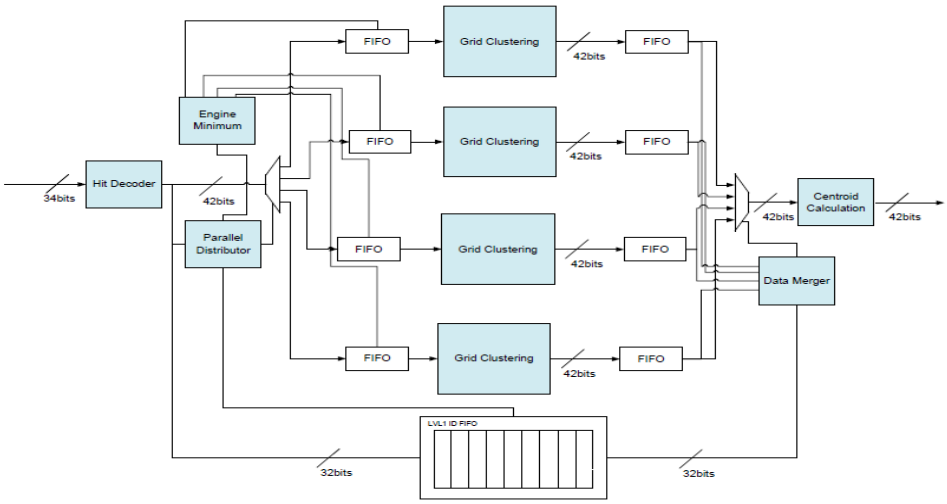

Fig. 4. Clustering Implementation

follow the clustering implementation also require a single data stream, therefore data parallelizing and serializing modules are introduced in order to interface the parallel clustering modules with the single input and output data streams. A parallel distributor module was developed that splits the data stream to the different engines by choosing for each arriving module the least busy one. Start event and end event words are propagated to each clustering engine that is used for a given event in order to keep track of the event boundaries. In order to guarantee that the event sequence is maintained, the sequence of arriving Level-1 IDs (event identifiers) is propagated through a FIFO to the data merger. The data merger module is used to serialize the data output. It restores the original event sequence. Fig. 4 shows three separate processing modules for the clustering in an example that is implemented with four parallel engines.

IV. Results

We produced the 10 prototypes of FTK IM boards. The boards passed basic hardware tested after production. The boards work well with a 2.0 Gbps input over optical fibers that are internally decoded to 32 bits words at 40 MHz, and 200 MHz DDR output rate. Data communication tests have been performed. No bit errors have been observed after checking 10^{16} bits, which meets the requirement from the ATLAS experiment. The clustering algorithm was tested in firmware simulation with realistic Monte Carlo data.

Ten boards where then taken to ATLAS and tested with data from the real RODs. The FTK IM board received hits data from ROD correctly and clustered outputs were consistent with expectation. The single flow 2D clustering was used for the test, and the algorithm works perfectly with real ROD data.

The 2D clustering algorithm was expanded to parallel version, and it is working well in firmware simulation. The implementation of 16 parallel engines has achieved a 65 MHz maximum clock frequency and occupies 40 % of a FPGA resource of FTK IM. This parallelized 2D clustering implementation has a more than enough processing power required for the Pixel detector even under the worst case condition of 80 overlapping pp collisions per bunch crossing that correspond to the maximum LHC luminosity planned until 2022.

References

[1] ATLAS collaboration, Fast TracKer (FTK) Techinical Design Report, CERN-LHCC-2013-007 ATLAS-TDR-021

[2] ATLAS collaboration, G.Aad et al., The ATLAS Experiment at the CERN Large Hadron Colooider, 2008 INST 3 S08003

[3] ATLAS collaboration, ATLAS Inner Detector Techical Design Report, CERN-LHCC-97-16 and CERN-LHCC-97-17

[4] ATLAS collaboration, G. Aad et al, ATLAS pixel detector electronics and sensors, JINST 3(2008) P07007

[5] Y. Unno, Nucl. Instr. and Meth. A 453 (2000) 109.

[6] Xilinx Inc, Spartan-6 Family Overview, : http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[7] E. Van der Bij, R. McLaren and Z. Meggyesi, S-LINK: A Prototype of the ATLAS Read-out Link., 4th Workshop on Electronics for LHC Experiments, Rome, Italy, 21 - 25 Sep 1998, pp.375-379