A Multi-Core FPGA-based Clustering Algorithm for Real-Time Image Processing

C.-L. Sotiropoulou1, A. Annovi2, M. Beretta2, P. Luciano3, S. Nikolidakis1, G. Volpi2

1 Aristotle University of Thessaloniki, 2INFN-Frascati, 3University of Pisa

*Presenter email: Lsoti@physics.auth.gr

Abstract

A multi-core FPGA-based 2D-clustering algorithm for real-time image processing is presented. The algorithm uses a moving window technique adjustable to the cluster size in order to minimize the data required for cluster identification. The window size is generic and application dependent (size/shape of clusters in the input images). A key element of this algorithm is the possibility to instantiate multiple clustering cores working on different windows that can be used in parallel to increase performance exploiting more resources on the FPGA device. In addition to the offered parallelism, the algorithm is executed in a pipeline, thus allowing the cluster readout to be performed in parallel with the identification. The algorithm is developed for the Fast Tracker processor for the trigger upgrade of the ATLAS experiment but is easily adjustable to other image processing applications which require real-time pixel clustering.

ATLAS Fast Tracker

- The Fast Tracker receives data from pixel and microstrip ATLAS detectors for all events accepted by the Level 1 trigger at 100 kHz event rate. It will reconstruct all events to provide in few dozens of microseconds all the tracks on transverse momentum above 1GeV/c to the Level 2 trigger.
- The clustering implementation is part of the FTK input Mezzanine located on the Data Formatting motherboard (DF). The FTK Ms will receive data from 256 S-Links running at 1.2 Gbps each. The total input rate will be 300 Gbps. Each hit is a 32-bit word received by the module at a 40 MHz rate.

ATLAS Pixel Module•

- 320kx144 pixels readout by 16 Front End (FE) chips
- Each module is comprised of 2 rows of 8 FE chips
- Each FE reads out pixel hits by double column
- The hits in the double column are scrambled

Clustering Module

- It groups the detector hits into clusters.
- It uses moving window technique where the window is a grid of clustering cells of generic size, determined by the type and size of the clusters in the application.
- The grid is aligned to place on the middle row of either column 0 or 1 (depending on whether it belongs to an even or odd module column) the first hit pixel (hit) that arrives from the hit decoder. This is due to the fact that hits are scrambled within each pair of columns in the FE output as well as in the hit decoder output.
- Hits are read from the input until a hit that belongs to a column beyond the clustering window is identified.
- The hits that belong to the same columns as the clustering window but they are not within the row range of the window are stored in a separate circular buffer.

Clustering Readout

1. Two cells in the clustering window are selected as “seeds” (coordinates (0, mid-row) and (1, mid-row)). The “seeds” are set in a “selected” state → blue colour
2. The “selected” state is propagated to the neighboring hits → arrow
3. The previously selected hit is now being read out to the next processing module → black colour
4. When a hit is read out it returns to an “empty state” → purple colour

The hits that do not belong to the cluster are recovered (read out) after the cluster is and they are written to the circular buffer in their absolute coordinates.

The pipeline algorithm allows the readout to be executed in parallel with the identification of the clusters.

After the hits are read out, the clustering window is reloaded with data from the circular buffer, thus “moving” the window to a new reference point. The “seeds” are selected and a new cluster is again read out. The process continues until a module trailer (the word which defines the end of the pixel module) is received and the circular buffer is empty.

For the ATLAS Fast Tracker application we are planning to use a grid size of 8x21 pixels (21 for the n direction and 8 for the x or eta direction). The most common cluster size is of two or three pixels. A larger grid is used in order to allow for bigger clusters or clusters that derive from merging neighboring clusters to be also identified. Clusters that, starting from the seed cells, propagate beyond the edges of the grid will be split.

Hit Decoder

- The Hit Decoder decodes the ATLAS bytestream format to a format useful for the clustering module.
- It is robust against errors in input data format: it can drop all unnecessary data (data from undefined events and modules) and propagate only useful hits and control words. When possible it introduces missing control words.
- It aligns incoming data in column sequence: The FE chips in the pixel module are numbered from the bottom left corner to the upper left corner and are read in the same sequence → the first half of the data arrives in reverse order. A LIFO is used to store the hits from FE chips 0-7. When hits from FE chips 8-15 arrive each hit is temporarily stored on a register and compared to the last data in the LIFO. The data with the smallest column number is propagated to the output first.

Centroid Calculation

- The post-processing step in the Clustering system is the one that actually performs the data reduction → it reduces the set of cluster coordinates to one.
- The FTK cluster centroid is calculated as linear interpolation of deposited charge. The charge imbalance between the two sides of the cluster is calculated and also the absolute pixel position is taken into account.
- The Centroid Calculation step will output the centroid cluster and also the cluster size in two coordinates.

Centroid Calculation:

\[
\text{ColMin} = \text{average} \times \text{ColMax} + \text{ColMin}
\]

\[
\text{RowMin} = \text{average} \times \text{RowMax} + \text{RowMin}
\]

\[
\text{ColMax} \times \text{RowMax} \times \text{average} = \text{ColMin} + \text{ColMax} \times \text{RowMin} + \text{RowMax} \times \text{average}
\]

\[
\text{RowMax} \times \text{ColMax} \times \text{average} = \text{RowMin} + \text{ColMax} \times \text{RowMax} + \text{RowMin} \times \text{average}
\]

Constant \( c \) is a function of pixel position

For the 3x2 cluster in the example the centroid coordinates would be (1.15, 2.5)

Clustering Module

- The clustering module is a data reduction process that identifies the clusters composed of pixel hits in the ATLAS pixel detector. For each cluster the output data is reduced to a single pair of coordinates, plus a few bits of cluster size information.
- The data from the pixel and strips readout drivers (RODs) is transmitted to FTK over an S-link channel. An S-Link decoder receives the data and it encodes it on a first data formatting procedure. The data is sent to an S-Link FIFO that is the source of data for the Clustering implementation.
- The Clustering implementation has three computational stages: a) a hit decoder module, b) a clustering module and c) a centroid calculation module.

Results

- For a window size of 8x21 pixels a 1% FPGA area is used (slice registers usage) on a Spartan 6 1x152 FPGA device.
- For the same device we have verified the functionality using a post-place and route simulation with a 12ns clock.
- A realistic hit occupancy shows that the processing time is roughly 10 clock cycles per hit. (The numbers will be verified by FTK simulation in the future)
- A parallel implementation with 10 cores working in parallel on the same device should achieve 80 MHz hit processing speed using less than 20% of the available FPGA resources.

- In the parallel implementation each core will process data from a whole pixel module so that they work independently. New logic will be developed to distribute the data to the clustering cores.
- Data are tagged by their event and module number in order to allow the final stream to be recomposed recovering the proper event sequence.

Acknowledgement

This work receives support 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 ARTICHOCF 254140 FPT-PEOPLE-2009-IF and FTK 320418 FPT-PEOPLE-2012-APP.