A Parallel FPGA Implementation for Real-Time 2D Pixel Clustering for the ATLAS Fast Tracker Processor

C.-L. Sotiropoulou, S.Gkaitatzis, A. Annovi, M. Beretta,
K. Kordas, S. Nikolaidis, C. Petridou, G. Volpi

Workshop on Intelligent Tracking
Philadelphia, 14-16 May 2014

Aristotle University of
Thessaloniki, Greece

2D Clustering

- Pixel detectors are used in a variety of applications, high-energy physics among them
- The high resolution of contemporary pixel detectors leads to increased demand for effective data reduction techniques
- Our clustering implementation targets pixels with direct proximity and it is developed for the ATLAS Fast Tracker (FTK) processor

Target: A system performing data clustering on the fly

Fast Tracking in the Inner Detector

Example:
R-phi view of Barrel region:

- Approved ATLAS upgrade for Run-II, installation starting on 2015
- Receives data over ~380 S-LINKS from
 - Pixel Detector
(also Insertable B-Layer - IBL)
 - Semiconductor Tracker (SCT)
- Process every event accepted by Level-1 trigger, up to 100kHz
- Provides a complete list of tracks for $p_T > 1$ GeV to the ATLAS high-level trigger (HLT)

FTK Architecture

- Input Mezzanine (IM) and Data Formatter (DF)
 - Cluster finding
 - Data reorganization in towers
 - Transmission of data to Data Organizers
- Data Organizer (DO)
- Associative Memory (AM)
 - Contains up to 10^9 preloaded patterns
- Track Fitter (8 Layer Fit)
 - Performs the fit using hits from the DO and the AM
- Hit Warrior
 - Removes duplicate tracks
- Second stage fitter (All 12 layers)
- FLIC (Format Data for HLT)

FTK Track Performance

d0 distributions FTK & offline

Primary vertex z resolution

- $\langle \mu \rangle$ is average pile-up
- Close to offline d0 distributions for b-tagging
- Improved performance with IBL under study

The ATLAS Pixel Module

- Module dimensions: 328x144 pixels
- Module readout performed by 16 Front End (FE) chips
- Each module is comprised of 2 rows of 8 Front End chips
- Pixel dimensions: $50\mu\text{m} \times 400\mu\text{m}$

G. Aad et al., "ATLAS pixel detector electronics and sensors", Journal of Instrumentation, Vol. 3, P07007, July 2008.

Clustering – Main Concept

	3	9	7						
	1					13	15		
	4	8	6	11					
							12		
2	5	10					14		

Goal: Keep up with
the 40MHz input rate

- Hits arrive scrambled by dual column...

- Loop over the list of hits
- Associate hits from the same cluster
- Execution time is non-linear and can have large event-by-event fluctuations

Clustering Implementation

Designed with three separate modules

- Hit Decoder module
- Grid clustering module
- Centroid calculation module

Hit Decoder

- Pixel hits arrive as 32 bit words clockwise from FE 0 to FE 15
 - A word includes the hit coordinates (column, row) and Time-Over-Threshold
 - Hits from FE < 8 come in reverse column order
 - Hits from FE ≥ 8 come in correct column order
- Hit Decoder fixes possible data stream errors
- Main Task for Hit Decoder: align the data stream from left to right

Grid Clustering – Concept

- Start reading from input FIFO
- First hit is the reference hit
- Place grid window (e.g. 4x5 – real 8x21)
 - Reference is on the middle row of grid window and
- Column 1 if it belongs to an odd column or
- Column 0 if it belongs to an even column

- Continue reading hits and place them on
 - Grid window, or
 - Circular buffer
- Reading stops when a hit which has a column higher than the grid window's maximum column

Grid Clustering – Selection & Readout

- Two cells in the cluster window are selected as “seeds”

Grid Clustering – Selection & Readout

- Two cells in the cluster window are selected as “seeds”
- The “selected” state is propagated to the neighbouring hits

Grid Clustering – Selection & Readout

- Two cells in the cluster window are selected as “seeds”
- The “selected” state is propagated to the neighbouring hits
- Previous hit is readout

Grid Clustering – Selection & Readout

- Two cells in the cluster window are selected as “seeds”
- The “selected” state is propagated to the neighbouring hits
- Previous hit is readout
- Selecting and reading out the cluster is executed in parallel

Grid Clustering – Selection & Readout

- Two cells in the cluster window are selected as “seeds”
- The “selected” state is propagated to the neighbouring hits
- Previous hit is readout
- Selecting and reading out the cluster is executed in parallel

- Hits that do not belong to the cluster are moved back to the circular buffer

Grid Clustering – Output

- Hits sent from grid to circular buffer and back until all clusters are identified
- Execution time is data dependent
- Current grid size: 8x21

Grid Clustering Output

- Reference hit is sent out in absolute coordinates in the end cluster word
- The other hits are sent out in local coordinates with respect to the reference hit
 - Less bits for calculation in the next processing element

Split clusters

- On potentially split clusters, the cluster might extend beyond the grid window
- On a sample with 80 pile-up proton-proton collisions and 8x21 grid window, potentially split clusters account for 1.5% of all clusters
- Grid window size can be modified to change the percentage

Centroid Calculation

The Centroid Calculation outputs the cluster centroid and the cluster size in two coordinates. One cluster is reduced to one word, reducing the data

0					
1					
2	30	70	80		
3	40	75	60		
4					
	0	1	2	3	4

$$qRatio = \frac{qColMax}{qColMin + qColMax} = \frac{80 + 60}{(30 + 40) + (80 + 60)} = \frac{140}{210} = \frac{2}{3}$$

$$x_average = \frac{ColMin + ColMax}{2} + a \times (qRatio - 0.5)$$

a is a function of pixel position

- Calculate charge imbalance between the two sides of the cluster
- Apply a correction with respect to the centroid
- For the above example cluster, centroid coordinates would be (1.15,2.5)

Parallel Implementation of Clustering

- A fundamental characteristic is that different clustering engines can work independently in parallel on different data
 - Performance increase due to greater FPGA usage
- A parallelization strategy was chosen:
 - Instantiate multiple clustering engines (grid clustering modules) that work independently on data from separate pixel modules
 - To achieve this, data parallelizing (demultiplexing) and data serializing (multiplexing) modules are necessary

Clustering – Parallel Version

Test Results

- Successful integration on hardware (FTK_IM board) for both Single Flow and 4 Parallel Engine Flow implementations
- Testing performed with events of 80 pile-up pp collisions
- The output for each implementation was compared with the bit-by-bit simulation with excellent accuracy, no differences encountered

Testing on a Spartan-6 LX150T-3

	FPGA Area (%)	Maximum frequency (MHz)
Single Flow	2.4	81.5
4x Parallel Flow	11.1	80.5
16x Parallel Flow	38.4	65.0

Test Results – Single Flow

- For the Single Flow the average processing time is 84.15ns per word
- Maximum input is one word every 25ns (40MHz)
- $84.15/25 = 3.36$ clock cycles of 40MHz are required for one data word processing
- It is anticipated that 4 engines will suffice for the worst case scenario once enough buffering is included

Test Results – Parallel Flow

- The 4 Parallel Engine Flow fully covers pixel specifications for 80 pile up events

Conclusions – Remarks

- A high performance FPGA-based implementation
- Uses a moving grid window technique to identify the clusters
 - Reduces required resources and data for clustering identification
- Fully generic, can be parallelized
- Achieves enough processing power for the ATLAS FTK processor, with an input rate of 40MHz using 4x parallelization and less than 15% FPGA usage

Thank you!