A 2D FPGA-based clustering algorithm for the LHCb silicon pixel detector running at 30 MHz

Giovanni Bassi on behalf of the LHCb RTA project
giovanni.bassi@cern.ch

2019 IEEE Nuclear Science Symposium (NSS) and Medical Imaging Conference (MIC), Manchester, UK, 26 October - 2 November 2019
Motivation

- After Upgrade I (2020), the LHCb readout system will deal with a 30 MHz data processing rate (average LHC bunch crossing rate)
Motivation

- After Upgrade I (2020), the **LHCb** readout system will deal with a **30 MHz data processing rate** (average LHC bunch crossing rate)
- Addressing low-level time-consuming tasks at early DAQ stages can lead to a significant **reduction of computing resources**
- **FPGA** accelerators are well suited to address, in a high parallel way, logically simple and repetitive processes
Motivation

- After Upgrade I (2020), the LHCb readout system will deal with a 30 MHz data processing rate (average LHC bunch crossing rate)
- Addressing low-level time-consuming tasks at early DAQ stages can lead to a significant reduction of computing resources
- FPGA accelerators are well suited to address, in a high parallel way, logically simple and repetitive processes
- Grouping contiguous pixels in single hits (clustering) is both a time demanding (2D pixel geometry) and a repetitive task
Motivation

- After Upgrade I (2020), the LHCb readout system will deal with a 30 MHz data processing rate (average LHC bunch crossing rate)
- Addressing low-level time-consuming tasks at early DAQ stages can lead to a significant reduction of computing resources
- FPGA accelerators are well suited to address, in a high parallel way, logically simple and repetitive processes
- Grouping contiguous pixels in single hits (clustering) is both a time demanding (2D pixel geometry) and a repetitive task
- We developed a FPGA-friendly clustering algorithm to tackle 2D clustering during early DAQ stages, while keeping essentially the same physics performances wrt CPU clustering
Physics Performances

<table>
<thead>
<tr>
<th></th>
<th>VeloClusterTracking (CPU)</th>
<th>FPGA + VeloClusterTracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velo tracks</td>
<td>efficiency 95,519% ± 0,014%</td>
<td>95,258% ± 0,014%</td>
</tr>
<tr>
<td></td>
<td>clone 1,365% ± 0,008%</td>
<td>1,421% ± 0,008%</td>
</tr>
<tr>
<td></td>
<td>hitEffFirst3 95,21%</td>
<td>94,51%</td>
</tr>
<tr>
<td>Long tracks</td>
<td>efficiency 97,705% ± 0,013%</td>
<td>97,493% ± 0,014%</td>
</tr>
<tr>
<td></td>
<td>clone 1,291% ± 0,010%</td>
<td>1,361% ± 0,010%</td>
</tr>
<tr>
<td></td>
<td>hitEffFirst3 95,70%</td>
<td>94,96%</td>
</tr>
<tr>
<td>Ghost tracks</td>
<td>0,8745% ± 0,0041%</td>
<td>1,0217% ± 0,0045%</td>
</tr>
</tbody>
</table>

Summary of the physics performances comparing CPU and FPGA clustering algorithms on commonly-used LHCb tracks (see backup for definitions and efficiency plots).
Background

- The clustering algorithm has been tailored for the LHCb Vertex Locator (VELO):
  - 26 layers, 2 modules each, both in the forward (19) and backward (7) regions wrt interaction point
  - Each module is read by a DAQ card and consists of 4 sensors, grouped in pairs (0-1, 2-3)
  - 41 M pixels in total
- Pixels are read in groups of 2x4 pixels (SuperPixels)
- During readout process each SP is flagged with an isolation bit (isolated = none of the 8 neighbors SPs have any active pixel)
- VELO clusters are typically made of few pixels (1-4)
Algorithm overview

- Isolated SPs are resolved with a Look Up Table (LUT)
- LUT connects each of the 256 ($2^8$) possible pixel configurations inside a SP to the center of mass of the cluster/s (if two clusters are generated)
- The two LUT outputs are then merged and stored in a FiFo

![Diagram showing a 2D FPGA-based clustering algorithm]

Giovanni Bassi

A 2D FPGA-based clustering algorithm
Algorithm overview

- SPs with neighbors fill a set of matrices, 5x3 SPs each (10x12 pixels).
- First SP filling a matrix determines the position of the matrix in the detector.
- If a SP belongs to a matrix it fills it, otherwise it moves forward, checking the next matrix or filling a blank one in the center.

Giovanni Bassi

A 2D FPGA-based clustering algorithm
Algorithm overview

- At the end of each event, in a fully parallel way, each pixel checks if it belongs to one of the following patterns, if so a cluster candidate is identified.

- Each cluster candidate is resolved using a LUT.

- The absolute cluster position is obtained as a “vector sum” of the matrix position wrt the detector, checking-pixel position wrt the matrix and cluster position wrt the checking pixel.

- Reconstructed clusters are then collected and stored in a FiFo.
Algorithm overview

- Algorithm parameters:
  - Matrix shape and size → average number of neighbor SPs and their arrangement
  - Number of matrices → distribution of total number of not isolated SP per event
  - Cluster maximum dimension (3x3 pixels) → distribution of cluster sizes
Algorithm overview

- Algorithm parameters:
  - Matrix shape and size
  - Number of matrices
  - Cluster maximum dimension (3x3 pixels)

- Examples of peculiar behaviors of the algorithm:

Giovanni Bassi
A 2D FPGA-based clustering algorithm
Basic blocks of the clustering firmware (SOP: Start Of Package, EOP: End Of Package, TFC: Timing & Fast Control)

A 2D FPGA-based clustering algorithm

Giovanni Bassi
Decoding

Basic blocks of the clustering firmware (SOP: Start Of Package, EOP: End Of Package, TFC: Timing&Fast Control)

Giovanni Bassi

A 2D FPGA-based clustering algorithm
Decoding and data structure

- Input data are organized in **256-bit words**, each containing 8 SPs (32 bits each).
- A decoder splits 256-bit words into **8 32-bit streams**.
- All SPs generated in a given event (p-p collision) are flagged with the same **timing signal** (TFC).
- With the first (last) word of an event a **SOP**, Start Of Package (**EOP**, End Of Package) signal is received.
- Starting from SOP and EOP signals, internally, EndEvent (**EE**) signals are generated to separate pixels originated from different events.
- The firmware has to deal with SPs coming from both the sensors in a couple and isolated/with-neighbors SPs, a **switch** is introduced to arrange SPs by sensor and by isolation flag, feeding different cluster reconstructing blocks.
Basic blocks of the clustering firmware (SOP: Start Of Package, EOP: End Of Package, TFC: Timing&Fast Control)
Switch

- The switch arranges SP by isolation flag (IF) and by sensor
- The splitter (1 in, 2 out) sends input data to one of the two outputs according to their IF/sensor
- The merger (2 in, 1 out) routes two inputs in a single output
- The 2in2 dispatcher contains 2 splitters and 2 mergers
- For an N-input switch \((N/2)^2\) dispatchers are needed

A 2D FPGA-based clustering algorithm

Giovanni Bassi
Clustering with matrices

Basic blocks of the clustering firmware (SOP: Start Of Package, EOP: End Of Package, TFC: Timing&Fast Control)

Giovanni Bassi

A 2D FPGA-based clustering algorithm
Each matrix is fed with **two data lines** coming from the switch (non isolated SPs).

At the end of each event, matrices are read and clusters are reconstructed.

Clusters are merged into a single line and output to a FiFo.

A **back-pressure** mechanism controls the data transfer, avoiding data losses.

Giovanni Bassi

A 2D FPGA-based clustering algorithm
A 2D FPGA-based clustering algorithm

- Pixels inside each matrix are checked according to the patterns specified by the algorithm.
- If there is a matching, a bit in the pixel flag word is raised, corresponding to the pixel position.
- An encoder passes the pixel address to a mux to extract the 3x3 cluster.
- The checked pixel is erased from the pixel flag word.
- The CM of the cluster is extracted with a LUT and the full cluster word is built.
Giovanni Bassi

A 2D FPGA-based clustering algorithm
Encoder

- The encoder takes 8 separate 32-bit streams and builds 256-bit words.
- The basic encoder block puts together DATA IN 0 and DATA IN 1 (N + N bits) into DATA OUT (2N bits).
- Input data come from differently populated FiFos.
- If the number of 32-bit input data is odd, zero-padded data are added to the output.
- Bandwidth losses are limited using registers (R0, R1 and R3) and contained within ~10%.
Hardware setup

- We tested the firmware on a prototyping board with:
  - 2 Stratix V FPGAs
  - 952k logic elements / FPGA
  - 650 MHz maximum clock

- The FPGA used for testing is comparable to the FPGA used on the LHCb DAQ board (Arria 10)

- The firmware requires (wrt Arria 10):
  - 24% of available logic elements
  - 6% of available M20K memory blocks
  - 350 MHz clock
LHC average bunch crossing rate is 30 MHz
We measured the firmware throughput with simulated data on the sensor with the highest occupancy, both with:
  ○ Minimum bias events, **throughput of 38.9 MHz**
  ○ High multiplicity $B_s \rightarrow \phi\phi$ events, **throughput of 30.9 MHz**
Simulated data are produced at a luminosity $L = 2 \cdot 10^{33} cm^{-2}s^{-1}$
Summary

- We developed a clustering algorithm that can process events from the most populated VELO sensor at 38.9 MHz, thus the clustering on FPGA can run in real-time.

- The FPGA-based algorithm shows essentially indistinguishable physics performances wrt CPU clustering, while reducing the workload on CPU farms.

- Dropping SuperPixels we can also reduce VELO readout bandwidth by ~14% with benefits to the data distribution system.

- The clustering firmware needs a relatively small amount of logic and memory, allowing the integration of the clustering firmware inside the DAQ VELO firmware, while maximizing the use of existing FPGA boards.
Physics performances are evaluated using LHCb MonteCarlo (MC) simulation, comparing the output of the track reconstruction using FPGA and CPU clustering algorithms.

**LHCb track classification**

- **Velo**: track with hits on the VELO
- **Long**: track with hits on the three LHCb tracker (VELO - UT - Scifi)
- **Ghost**: a reconstructed track not associated to a MC track
- **Clone**: a second reconstructed track associated to the same MC track
- **HitEffFirst3**: percentage of tracks reconstructed using the first 3 hits in the VELO
Efficiency studies

Comparison between CPU- and FPGA-clustering tracking efficiencies. Long tracks are used as an example: all kind of long tracks (left), long tracks from electrons (right)
Splitter working principle

● The splitter sends input data to one of the two outputs (or both), according to a pre-set **splitting condition** (a specific bit of the input word is checked)

● For each input:
  ○ If the condition is not met the word is sent to the top output line
  ○ If the condition is met the word is sent to the bottom output line
  ○ If the input word is an EE then it is sent to both output lines
Legend:
- R0: register
- LE: writing enable signal
- Sout: next register state
- Sin: current register state
- Control: mux input select
The merger routes two independent input lines in a single output line:
- If a SP is received on one of the two input lines is sent as output
- If two SPs are received they are sent one after the other
- If the same EE arrives on both inputs it is sent as output
- If two different EE arrive on the inputs an error signal is raised
- If a single EE arrives on one of the inputs it is written in a register, waiting for the other EE to arrive
Merger schematic

Legend:
- R0, R1: registers
- LE: writing enable signals
- Sout: next register state
- Sin: current register state
- Control: mux input select

A 2D FPGA-based clustering algorithm
Test details

- Since the number of events that we can process per unit of time is proportional to: $EVT_{rate} \propto \frac{1}{\#SPs}$ we tested the firmware with SPs from the most populated sensor.
- Reconstructed clusters are compared with the output of an high-level C++ simulation to ensure matching.
Error handling

- Within the firmware two kind of errors can show up:
  - **Data loss** occurs when an entity receives valid data in input and the register in which data should be written is already full
  - **EE mismatch** occurs when input data lose event synchronization and SPs from different events are mixed
- Several entities can generate errors:
  - Switch splitter outputs an error if there is a data loss
  - Switch merger outputs an error if there is a data loss or an EE mismatch between inputs
  - Matrix chain outputs an error if there is an EE mismatch between the two input lines
  - Matrix chain merger outputs an error if there is an EE mismatch between the input lines
  - Encoder outputs an error if there is a data loss or an EE mismatch between the inputs
- Errors are propagated upstream to an **error handling register**, that allows the identification of the error source, simplifying debugging
Data format

- 32 bit SuperPixel word

- 32 bit Cluster word

Giovanni Bassi  A 2D FPGA-based clustering algorithm
Bandwidth reduction

- The main challenge for the trigger-less readout is to build a cost-effective system that can handle the sizable **bandwidth of 40 Tbps** (whole LHCb).
- Average 2254 SuperPixels/event (VELO) → Required bandwidth: 32 bit/SP * 2254 SuperPixel/evt * 30 MHz = **2,16 Tbps**
- Average 1709 clusters/event (VELO) → Required bandwidth: 32 bit/cluster * 1709 cluster/evt * 30 MHz = **1,64 Tbps**
- Given the quality of the results, it seems sensible to discard SPs in favor of clusters → **amount of data reduced** ∼ **24 %**
- Due to encoder bandwidth losses (∼10 %), the total reduction of VELO bandwidth is **∼14 %**
Throughput evolution

A 2D FPGA-based clustering algorithm