Readout firmware of the Vertex Locator for LHCb Run 3 and beyond

Karol Hennessy
on behalf of LHCb
University of Liverpool / CERN
October 14, 2020
LHCb VELO

- LHCb - Flavour physics detector
- Excellent **vertexing** resolution and Particle ID
- LHCb has triggerless readout - full detector readout @ 40 MHz

- Vertex Locator (VELO)
- Silicon pixel modules around the LHC collision region
  - 50fb$^{-1}$ integrated luminosity for LHC Runs 3 & 4
  - Very high radiation environment
  - In vacuum and under active cooling
LHCb VELO

- LHCb - Flavour physics detector
- Excellent **vertexing** resolution and Particle ID
- LHCb has triggerless readout - full detector readout @ 40 MHz

- Vertex Locator (VELO)
- Silicon pixel modules around the LHC collision region
  - 50fb$^{-1}$ integrated luminosity for LHC Runs 3 & 4
  - Very high radiation environment
  - In vacuum and under active cooling
VELO Module

- Whole VELO = two halves of 26 modules
- Four sensors per module
  - 2 front
  - 2 back
- 3 VeloPix per sensor (i.e., 12 total)
- 20 high speed readout links
  - Chips closer to beam see more hits, and need more bandwidth
VELO Module

- Whole VELO = two halves of 26 modules
- Four sensors per module
  - 2 front
  - 2 back
- 3 VeloPix per sensor (i.e., 12 total)
- 20 high speed readout links
  - Chips closer to beam see more hits, and need more bandwidth
VeloPix ASIC

- Front-end ASIC driving the design of the VELO data acquisition system
- Part of the MediPix/TimePix family
- 130 nm CMOS technology
- 256 × 256 pixels of 55 × 55 µm²
- Clocked at 40 MHz
- Sends binary hit information (reducing bandwidth requirement)
  - Full signal amplitude (ToT) available via slow readout for calibration

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
** VELO Electronics and DAQ **

---

**a slice of the VELO readout system**

- See Flavio’s talk on Friday for a fuller description of the LHCb DAQ

---

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
Why an FPGA?

- **Lots of data!**
- **VeloPix is optimised for high speed readout**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak hit rate</td>
<td>900 Mhits/s/ASIC</td>
</tr>
<tr>
<td>Max data rate</td>
<td>19.2 Gb/s</td>
</tr>
<tr>
<td>Total VELO</td>
<td>2.85 Tb/s</td>
</tr>
</tbody>
</table>
Readout Board - PCIe40/TELL40

- Single control and readout board for the entire experiment
- Can be used for Timing, Slow Control, DAQ or all
- Common hardware, shared firmware components
- PCIe Gen3 x16
- Intel Arria10 FPGA (10AX115S4F45E3SG)
- 1 TELL40 = 1 VELO module

- up to 48 bi-directional links @ ~5 Gb/s
- Output bandwidth 100 Gb/s (measured).
So what does the VELO Firmware have to do?
What does VeloPix produce?

- **Time unordered data**

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
What does VeloPix produce?

- Time unordered data
- Custom transmission Protocol

- Custom serializer - **Gigabit Wireline Transmitter (GWT)**
  - Chosen for low power - 60 mW
  - 5.12 Gb/s line rate (slightly higher than 4.8 Gb/s of GBT)

- GWT protocol
  - scrambled data (30 bit multiplicative)
  - parity check, no error recovery
  - low tolerance for header errors
What does VeloPix produce?

- Time unordered data
- Custom transmission Protocol
- **SuperPixels**
  - Pixel data is aggregated into groups of $2 \times 4$ called **SuperPixels**
    - 30% reduction in data size
  - Timestamp stored in SuperPixel data packet

```text

<table>
<thead>
<tr>
<th>SuperPixel Address</th>
<th>Timestamp</th>
<th>Hitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

timestamp: $t = 12$
```
What’s in the TELL40 Firmware?

- **Convert Electrical to Optical for Long distance**
- **FPGA with firmware to process VELO data**
- **TELL40**
- **Data Processing**
- **Output Formatting**
- **Deserialisation & Decoding**
- **Timing and Controls input from LHCb**
- **High Speed Electrical Transmission**
- **Opto-Power Interface Board**
- **20 fibres @ 5.13 Gbps over 300m**
- **Network @ ~100 Gbps**
- **Event Building, Reconstruction, Find Physics...**
- **Fragments**
- **Tracks**
- **Particles**
- **B**
- **K**
- **pi**
• Actually, it’s two parallel streams for PCIe bandwidth optimisation
• But it’s simpler to describe just one
Handling VeloPix data

Going back to our list:

1. Custom transmission Protocol
2. Time unordered data
   SuperPixels
3. *is a generic component and won’t be discussed here*
Handling VeloPix data - Deserialisation & Decoding

Deserialisation & Decoding

Made using Quartus Platform Designer

search for 1010 header pattern and lock after N repetitions

Serial Input @ 5.12 Gbps

RX Transceiver IP

32b @ 160 MHz

GWT Frame Alignment

31b

Retiming to 40 MHz

30 bit scrambling \(x^{30} + x^{16} + x^{15} + x + 1\)

consequence of shared codebase

124b to data processing

Made using Quartus Platform Designer

30 bit scrambling

\(x^{30} + x^{16} + x^{15} + x + 1\)

consequence of shared codebase

Timing and Controls Input from LHCb

Data Processing

Output Formatting

TELL40
Handling VeloPix data

Going back to our list:
1. Custom transmission Protocol
2. Time unordered data
   SuperPixels
3. *is a generic component and won’t be discussed here*
Handling VeloPix data - SuperPixel Extraction

SuperPixel Extraction

<table>
<thead>
<tr>
<th>30b</th>
<th>30b</th>
<th>30b</th>
<th>30b</th>
<th>4b</th>
<th>4b</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPP3</td>
<td>SPP2</td>
<td>SPP1</td>
<td>SPP0</td>
<td>PAR</td>
<td>HDR</td>
</tr>
</tbody>
</table>

GWT Format

- split frame of 4 SPPs into individual packets and quadruple clock speed
- VeloPix encodes timestamp in Gray Code

Data from GWT RX:

- 124 b @ 40 MHz
- SPP Extraction
  - 30b @ 160 MHz
- Gray to binary
- Data Processing
  - Timestamp Sorting
  - Timing Alignment
  - Clusterisation

Output Formatting

TELL40

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
Handling VeloPix data - Time Reordering

- Timestamps are sorted 1 bit at a time in several layers
  - First column is MSB...
- Fifos are needed to avoid collisions
- Data are stored in RAMs at the end of the routing
- The whole reordering consumes a large amount of the FPGAs memory
Handling VeloPix data - Time Reordering

- After the routing, the RAM address is equivalent to the timestamp
- SuperPixel Packets are stored for a maximum latency of 512 clock cycles
- A swinging buffer is used to maximise bandwidth
Handling VeloPix data - Time Alignment

• After reordering, data must align to the rest of LHCb.
• **Timing and Fast Control** (TFC) system provides LHCb timing metadata.
Handling VeloPix data - Clusterisation

- Velo Data 256 bit
  - (Avalon Streaming)
  - Decoder
    - Isolation Flagger
      - Splits Velo Data in 8 streams
      - Adds End Event
  - SuperPixel Extraction
  - Timestamp Sorting
  - Timing Alignment
  - Clusterisation

- TFC
  - Metadata
  - Bypass of special data
  - TFC Forwarding

- Switch
  - Arranges SPs by sensor and Isolation Flag
  - Clustering isolated
    - Clustering matrices
      - RC

- Encoder
  - Arranges clusters in 256 bit packages
  - Velo Data 256 bit
  - (Avalon Streaming)

- Bypass of special data depending if isolated or not (Avalon Streaming)
Handling VeloPix data - Clusterisation

- Splits Velo Data in 8 streams
- Adds End Event

Switch
- Arranges SPs by sensor and Isolation Flag

Clustering Matrices

- Arranges clusters in 256 bit packages

Colour represents the SP position: SPs with the same colour are neighbours

Velo Data 256 bit
Decoder
ICF
- Splits Velo Data in 8 streams
- Adds End Event

Switch
- Arranges SPs by sensor and Isolation Flag

Clusterisation

- Direct data depending if isolated or not

Isolated Cluster Flagging
Not Isolated
Isolated

Isolated Cluster Flags

Avalon Streaming

Clusterisation Matrices

0
Inactive Pixel
1
Active Pixel
Don't Care

Colour represents the SP position: SPs with the same colour are neighbours
Challenges
Deserialisation & Decoding - Congestion

- v1
- v2 Improved frame aligner
- v3 Improved bit slipping
Full Data Processing

Now adding the full Time Reordering and Alignment

Timing closure becomes tricky
Resource Estimate

- Very preliminary estimate for now (sum of individual compilations - not the output of a complete build)

<table>
<thead>
<tr>
<th>Logic (ALMs)</th>
<th>M20K RAMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp Reordering</td>
<td>39</td>
</tr>
<tr>
<td>Clustering</td>
<td>31</td>
</tr>
<tr>
<td>Total</td>
<td>70</td>
</tr>
</tbody>
</table>

- Looks like it will fit, but *congestion* and *timing closure* are the major challenges ahead
A working firmware...

- Full data processing is not complete today
- A bypass is used for production and testing
  - Sorting/processing is done on CPU
  - Rate limited
- Can also be used to check data processing (send same data to both and check)
Tools/Organisation

- Typical combination of Questasim and Quartus
- LHCb employs gitlab pipelines for checking new releases
  - Sim-checker injects files into firmware and verifies on output
  - Can add additional testbenches and cross-checks à la “nightlies”
  - Strict versioning and tracking
- VELO makes stable releases for production testing
...and beyond
• HL-LHC (2028) will provide $7.5 \times$ luminosity
• Meaning $7.5 \times$ tracks/hits...
• Meaning we need a new VELO to go from this

• to this
...and beyond

- Add extra timing precision (necessary for vertex/tracking)
- Bandwidth increase $O(10)$

<table>
<thead>
<tr>
<th>Process</th>
<th>Arria10</th>
<th>Agilex*</th>
<th>Factor Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Elements (k)</td>
<td>1150</td>
<td>2692</td>
<td>$\sim 2$</td>
</tr>
<tr>
<td>M20k Memory (Mb)</td>
<td>53</td>
<td>259</td>
<td>4.9</td>
</tr>
<tr>
<td>DSP</td>
<td>1518</td>
<td>17056*</td>
<td>$\sim 16$</td>
</tr>
</tbody>
</table>

Table: Comparison of FPGA resources for VELO U1b and a candidate for U2.

- Next gen FPGAs not quite scaling with the needs of the experiment!
- What can we do with all these DSPs?
Concluding Remarks

- LHCb VELO firmware on track to process VeloPix data
- Validating Time Reordering and Clustering
- Several challenges in terms of FPGA resources and timing closure
  - Confident we can solve these
  - We welcome any clever suggestions/tips
- Learning techniques to optimise the next generation of the experiment
- Need to adapt to the changing landscape of heterogenous computing
backup
BXID Router

- Time-ordering SuperPixel data
  - 9-bit router sorts data 1 bit at a time
  - Extensive simulation required - both to maximise speed (>160 MHz) and minimise FPGA resource usage
  - Latency limit < 512 clock cycles

Efficiency from Simulation: 99.99%

* no analog pile-up included

9-bit BXID
(512 clks)

sufficient!
Special VeloPix
A lot of non-standard DAQ elements...

- VeloPix has **NO SCA**
  - SLVS communication component required
  - extra SOL40 firmware
  - extra SOL40 software

- VeloPix **does NOT use GBT** for DAQ
  - uses GWT
  - different frequency - 5.12 Gbps
  - special VELO LLI firmware component
  - special firmware decoding, clocking...
  - special VELO LLI software

- VeloPix sends data **“unsynchronised”**
  - Firmware re-aligns data
  - Cannot filter events pre-alignment
  - Special dataflow monitoring needed

- Big effort from Online, Annecy, Marseille to help integrate into the standard firmware and software. Must remain vigilant and ensure “special cases” are tested as standard.
Bypass detail

Bypass

10 input links
124 @ 40 MHz

select non-empty fifo

rate limited; fifo overflow possible

to PCIe output
256b @ 250 MHz

TFC Metadata

Deserialisation &
Decoding

Processing

Output Formatting

Timing and
Controls Input
from LHCb
Isolated Cluster Flagging

ICF - Isolated Cluster Flagging

Decoder & ICF

SOP & EOP to End Event logic

BYPASS FIFO

Isolated

Not Isolated

DATA

TFC

COMMANDS

BYPASS LOGIC

BYPASS DATA

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
Isolated Clustering

### Clustering isolated

```
0 1
0 1
0 0
1 1
```

<table>
<thead>
<tr>
<th>Inactive Pixel</th>
<th>Active Pixel</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- SP col
- SP row
- Pixels
- Cluster position

- Cluster 0
  - Cluster0 col
  - Cluster0 row
- Cluster 1
  - Cluster1 col
  - Cluster1 row

- # Clusters > 1

---

Readout firmware of the Vertex Locator — October 14, 2020 — K. Hennessy
Clustering Matrices

Clustering Matrices

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0: Inactive Pixel
1: Active Pixel
Don't Care

Colour represents the SP position: SPs with the same colour are neighbours

Checking pixel
Cluster candidate

Clustering Matrices

colour represents the SP position: SPs with the same colour are neighbours

Cluster candidate
Active Pixel
Inactive Pixel
Checking pixel
Don't Care