The RD53 Data Format and Communication Efficiency

Michael McGinnis and Anders Ryd

Cornell University

On behalf of RD53 Collaboration
Outline

- RD53 Overview
- Data formatting task
  - CMS Pixel Detector for HL-LHC
    - Cluster shapes
    - Information content – optimal data volumes
- Proposed readout formatting
- Performance
- Outlook and Summary
RD53 Collaboration and RD53A Chip

- Joint CMS+ATLAS Pixel Readout Chip Development
  - 24 Institutions
  - 65 nm – high logic density (50×50 μm² pixels), low power
  - 4000 fb⁻¹, 500 Mrad tolerance
  - <PU>=200, 3 GHz/cm² rate, up to 1 MHz trigger rate
  - Low threshold – 600 e⁻

- RD53A Prototype Chip
  - Large Scale Demonstrator
    - 20×11.5 mm² (half of full chip)
  - Three different frontends
  - Chip submitted Aug. 2017
    - First tests in Dec. 2017
  - RD53A Manual: [https://cds.cern.ch/record/2287593/](https://cds.cern.ch/record/2287593/)

Overall RD53A performance good and provides solid baseline for final chip
Readout Architecture

- Pixel matrix built from 8×8 pixel cores
  - Built from 2×2 pixel regions
  - All cores are identical
- FEs are placed in ‘analog islands’ surrounded by the digital logic
- Trigger data is moved to the chip periphery within the pixel core column
  - Each core column moves the data independently of other core columns
- The formatting of the readout data was not developed for the RD53A prototype

This presentation describes a proposed data formatting scheme that is currently being implemented for RD53B
The main goal of the data formatting is to allow the readout within the available chip bandwidth

- In the innermost region of the detector the bandwidth limitation is the bottleneck
- Further away from the IP we like to minimize the number of links in order to reduce material and power

The formatting should be lossless

Since data is processed independently in each pixel core bus (PCB) the readout order of the PCBs should be independent

Same formatting to be used in all regions of the detector

- The ROC can be configured to use a different number of readout links, or have multiple ROCs share one link
  - These links operate at 1.28 Gbits/s

We will not consider link formatting here, e.g. Aurora 64/66
- The CMS detector layout

- The highest occupancies are near the beam line
  - However using varying number of links to read out modules the link load is balanced such that it is important to reduce the payload in each detector region
    - E.g. in CMS we use three (1.28 Gbps) links per readout chip in the inner layers while in the outer layers four readout chips share one link
Cluster Shapes

- The hit patterns look very different
  - The highest occupancy regions are the most forward modules in the inner layers
    - Clusters are very long
  - In the disks the clusters are small with just a few pixels

### Formatting needs to work efficiently in both configurations

- `<# clusters/ROC>=25`
- `<# pixels/cluster>=16`
- `<# pixels/ROC>=415`

- `<# clusters/ROC>=57`
- `<# pixels/cluster>=2.1`
- `<# pixels/ROC>=125`
We can use the entropy of the hits to estimate the amount of information we have in the pixels. This allows us to calculate a lower bound for how much data, counted in bits, we would need to read out for an optimal formatting of the data.

This has been studied by M. Garcia-Sciveres and X. Wang[1]. We will compare the performance of the implementation that is presented here to the calculated data volume.

The calculated minimum data volume considers that clusters are identified and then considers the number of bits needed to encode:

- Cluster address
- Cluster shape (and size)
- Charge information

- **Address encoding**
  - For a cluster occupancy of 0.025% we need about 14 bits per address

- **Cluster encoding**
  - For small clusters (2-3 pixels) we would need about 5 bits on average

- **Charge encoding**
  - Optimal encoding would use \( \sim 2.2 \) bits/per pixel for charge fraction and \( \sim 4 \) bits for cluster charge
  - We do not format the charge
    - 4 bits per pixel is always used

---

**Estimate of Optimal Data Size**

\[
H = -\sum_i p_i \log_2 p_i
\]

RD53 Readout Chip

- The readout chip (studied for the CMS detector) has 440 columns \( \times \) 328 rows
  - This is a total of 144320 pixels
    - 18 bits are required to encode the address of a pixel. With 4 bits for the charge the most naïve format uses 22 bits/pixel

- The 440 columns are organized as 55 ‘pixel core buses’ (PCB) of each 8 pixels
  - The data are read out independently in each of these 55 core buses

- The formatting scheme should work independently in each of the core buses
  - The overhead with correlating hits in neighboring core busses is too large
    - Two core busses may be reading out data from different events at the same time, and aligning which hits belong to one event before formatting would be very resource expensive

- In general the encoding has to be reasonably simple
  - We transmit the charge as 4 bits/pixel
The readout of the 55 pixel core buses is independent and they may complete in any order
- Readout of data should be allowed in any core bus order
- The data stream format uses a 6 bit core bus address for core buses with pixel hits
  - Addresses on the format 111XXX are 56 and higher and corresponds to non-physical addresses. Hence an address starting with ‘111’ flags end of data for the ROC

```
010010{data for PCB 34}000101{data for PCB 9}....110011{data for PCB 51}111
```

Address for PCB 34  Address for PCB 9  Address for PCB 51  End of Data

Next we will look at how data are formatted in the PCB
The PCB consists of 41 Pixel Core Regions of 8 × 8 pixels
  - For the formatting we consider regions of 2 rows by 8 columns – that is the pixel core regions are divided into 4 regions such that we have 164 regions in a PCB
  - We use 7 or 8 bits to encode the pixel region address
    - Addresses 0-127 use the format: 0aaaaaaa
    - Addresses 128-163 uses the format: 1bbbbbb where the actual address is 10bbbbbb
      - Since if the address starts with a 1 the second digit is always zero and can be suppressed
  - Since regions are read out sequentially one could use a relative address. For some additional complexity you can make a small gain
  - If two sequential regions have data - flag this and don’t send second address

Next we will look at how data are formatted in the 2 × 8 pixel regions
- Assume we have 4 pixels with hits (4 bits for charge)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>AAAA</th>
<th>BBBB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCC</td>
<td>DDDD</td>
</tr>
</tbody>
</table>

- **Encode in ‘Binary Tree’:**

```
  0
 / \     
11   11    
   / \   /  
  11  0  10  11
```

Split left-right: 0 indicates data only in right half (10 indicate data in left half 11 in both)

Split up-down: 11 indicate data in both halves

Split left-right

Split left-right

Different orders of splitting the region can be used

**Final format:** 0 11 11 0 10 0 11 AAAA BBBB CCC DDDD
Data Sample & Readout Assumptions

- We have studied the formatting performance in CMS using the full detector simulation (based on same detector response simulation as used in the CMS Tracker TDR)
  - Similar results are obtained in ATLAS (poster by K. Wraight and ATL-PHYS-PUB-2019-014)

- These studies are based on occupancies in \(t\bar{t}\) events with an average of 200 pileup interactions

- For readout link occupancy we assume 750 kHz trigger rate – the maximum readout for the CMS detector at the HL-LHC
  - In the regions with the highest occupancy, i.e. L1, CMS use three 1.28 Gbit/s links (‘e-links’) per ROC
  - In the outer regions of the detector up to 4 ROCs may share one readout link

- These studies do not include any overhead for the data format on the link, e.g. the Aurora 66/64 bit formatting
Data Formatting

- `<#pixels/ROC> = 21`  
  Size (22 bits/pixel) = 470  
  Format size = 280  
  Min. Size = 161

- `<#pixels/ROC> = 21`  
  Size (22 bits/pixel) = 460  
  Format size = 305  
  Min. Size = 170

- `<#pixels/ROC> = 320`  
  Size (22 bits/pixel) = 7000  
  Format size = 3700  
  Min. Size = 1950

- `<#pixels/ROC> = 415`  
  Size (22 bits/pixel) = 9150  
  Format size = 3890  
  Min. Size = 1495

- `<#pixels/ROC> = 124`  
  Size (22 bits/pixel) = 2728  
  Format size = 1983  
  Min. Size = 1418
Interpretation of Data Volume

- For the regions on the previous page we can compare achieved data size to theoretical limits
  - We can separate out the address+cluster shape vs. pixel charge performance

<table>
<thead>
<tr>
<th>Region</th>
<th>Theoretical Limit (bits)</th>
<th>Formatting Performance (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Addr.</td>
<td>Charge</td>
</tr>
<tr>
<td>L1 - Middle</td>
<td>1062</td>
<td>888</td>
</tr>
<tr>
<td>L1 - End</td>
<td>530</td>
<td>965</td>
</tr>
<tr>
<td>L4</td>
<td>99</td>
<td>62</td>
</tr>
<tr>
<td>Disk 1 - Inner</td>
<td>923</td>
<td>495</td>
</tr>
<tr>
<td>Disk 1 - Outer</td>
<td>105</td>
<td>65</td>
</tr>
</tbody>
</table>

- Formatting of addresses biggest limitation
  - Clusters split across two PCB & using full PCB address
Detailed Summary of Data Size

<table>
<thead>
<tr>
<th>Det. Region</th>
<th>Nroc</th>
<th>Avg. # Pixels</th>
<th>Raw size (bits)</th>
<th>Format size (bits)</th>
<th>Data Reduct.</th>
<th>Data Rate (Gbits/s)</th>
<th>N-elink</th>
<th>Link Occ. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBPX:L1:Inner</td>
<td>12744</td>
<td>371.5</td>
<td>8173.9</td>
<td>3793.2</td>
<td>2.2</td>
<td>2.84</td>
<td>3.00</td>
<td>74.1</td>
</tr>
<tr>
<td>TBPX:L1:Outer</td>
<td>12744</td>
<td>301.7</td>
<td>6637.8</td>
<td>3148.7</td>
<td>2.1</td>
<td>2.36</td>
<td>3.00</td>
<td>61.5</td>
</tr>
<tr>
<td>TBPX:L2</td>
<td>50976</td>
<td>90.3</td>
<td>1986.1</td>
<td>1062.0</td>
<td>1.9</td>
<td>0.80</td>
<td>1.00</td>
<td>62.2</td>
</tr>
<tr>
<td>TBPX:L3</td>
<td>84960</td>
<td>36.8</td>
<td>809.7</td>
<td>466.6</td>
<td>1.7</td>
<td>0.35</td>
<td>0.50</td>
<td>54.7</td>
</tr>
<tr>
<td>TBPX:L4</td>
<td>11894</td>
<td>21.3</td>
<td>467.7</td>
<td>278.5</td>
<td>1.7</td>
<td>0.21</td>
<td>0.25</td>
<td>65.3</td>
</tr>
<tr>
<td>TFPX:R1:Inner</td>
<td>37760</td>
<td>124.5</td>
<td>2739.1</td>
<td>1986.4</td>
<td>1.4</td>
<td>1.49</td>
<td>2.00</td>
<td>58.2</td>
</tr>
<tr>
<td>TFPX:R1:Outer</td>
<td>37760</td>
<td>66.3</td>
<td>1459.1</td>
<td>1041.0</td>
<td>1.4</td>
<td>0.78</td>
<td>1.00</td>
<td>61.0</td>
</tr>
<tr>
<td>TFPX:R2:Inner</td>
<td>60416</td>
<td>68.8</td>
<td>1513.2</td>
<td>1085.7</td>
<td>1.4</td>
<td>0.81</td>
<td>1.00</td>
<td>63.6</td>
</tr>
<tr>
<td>TFPX:R2:Outer</td>
<td>60416</td>
<td>42.6</td>
<td>937.9</td>
<td>658.6</td>
<td>1.4</td>
<td>0.49</td>
<td>1.00</td>
<td>38.6</td>
</tr>
<tr>
<td>TFPX:R3:Inner</td>
<td>90624</td>
<td>35.7</td>
<td>785.8</td>
<td>542.2</td>
<td>1.4</td>
<td>0.41</td>
<td>0.50</td>
<td>54.7</td>
</tr>
<tr>
<td>TFPX:R3:Outer</td>
<td>90624</td>
<td>26.4</td>
<td>581.8</td>
<td>391.1</td>
<td>1.5</td>
<td>0.29</td>
<td>0.50</td>
<td>54.7</td>
</tr>
<tr>
<td>TFPX:R4:Inner</td>
<td>12083</td>
<td>23.2</td>
<td>510.9</td>
<td>338.7</td>
<td>1.5</td>
<td>0.25</td>
<td>0.25</td>
<td>71.1</td>
</tr>
<tr>
<td>TFPX:R4:Outer</td>
<td>12083</td>
<td>18.8</td>
<td>413.6</td>
<td>268.0</td>
<td>1.5</td>
<td>0.20</td>
<td>0.25</td>
<td>71.1</td>
</tr>
<tr>
<td>TEPX:R1:Inner</td>
<td>37640</td>
<td>49.8</td>
<td>1095.7</td>
<td>818.4</td>
<td>1.3</td>
<td>0.61</td>
<td>0.50</td>
<td>80.4</td>
</tr>
<tr>
<td>TEPX:R1:Outer</td>
<td>37640</td>
<td>33.9</td>
<td>745.6</td>
<td>553.0</td>
<td>1.3</td>
<td>0.41</td>
<td>0.50</td>
<td>44.7</td>
</tr>
<tr>
<td>TEPX:R2:Inner</td>
<td>52688</td>
<td>26.7</td>
<td>587.6</td>
<td>428.6</td>
<td>1.4</td>
<td>0.32</td>
<td>0.50</td>
<td>44.7</td>
</tr>
<tr>
<td>TEPX:R2:Outer</td>
<td>52688</td>
<td>21.1</td>
<td>464.2</td>
<td>333.6</td>
<td>1.4</td>
<td>0.25</td>
<td>0.50</td>
<td>44.7</td>
</tr>
<tr>
<td>TEPX:R3:Inner</td>
<td>67680</td>
<td>18.6</td>
<td>409.2</td>
<td>291.6</td>
<td>1.4</td>
<td>0.22</td>
<td>0.25</td>
<td>65.4</td>
</tr>
<tr>
<td>TEPX:R3:Outer</td>
<td>67680</td>
<td>17.1</td>
<td>376.3</td>
<td>266.0</td>
<td>1.4</td>
<td>0.20</td>
<td>0.25</td>
<td>47.8</td>
</tr>
<tr>
<td>TEPX:R4:Inner</td>
<td>75200</td>
<td>14.2</td>
<td>312.7</td>
<td>218.5</td>
<td>1.4</td>
<td>0.16</td>
<td>0.25</td>
<td>47.8</td>
</tr>
<tr>
<td>TEPX:R4:Outer</td>
<td>75200</td>
<td>12.5</td>
<td>274.8</td>
<td>189.1</td>
<td>1.5</td>
<td>0.14</td>
<td>0.25</td>
<td>47.8</td>
</tr>
<tr>
<td>TEPX:R5:Inner</td>
<td>90240</td>
<td>12.4</td>
<td>272.4</td>
<td>187.8</td>
<td>1.5</td>
<td>0.14</td>
<td>0.25</td>
<td>47.8</td>
</tr>
<tr>
<td>TEPX:R5:Outer</td>
<td>90240</td>
<td>10.7</td>
<td>235.0</td>
<td>160.8</td>
<td>1.5</td>
<td>0.12</td>
<td>0.25</td>
<td>40.9</td>
</tr>
</tbody>
</table>

Data volume and link occupancy summary for CMS pixel tracker.
Formatted data fits with the planned readout links.
Next Steps

- This formatting scheme is in the process of being implemented for the RD53B chip
  - A first implementation now exists and the chip simulation results are being validated against the simulation studies presented here

- Unpacking of the data is under study
  - Algorithmically it is straightforward, but the challenge is how to do this efficiently, with low latency, on an FPGA
Summary

- This ‘binary tree’ format provides an implementation of the data formatting that efficiently formats the most dense data in the barrel layers and the more sparse regions.

- The formatting is done in one pixel core bus at the time.
  - Formatting can be done when the data in the PCB is read out.

- Provides sufficient data to meet the requirements for the CMS readout of the Layer 1 data at 750 kHz.
  - Compared to theoretical information content.
    - The independent core bus readout splits clusters and increases the number of addresses that needs to be read out.

- A first implementation now exists for the RD53B chip. We are working on validations of the hardware with respect to the simulation.
Using a different splitting order

This is an example of different splitting order where the up-down split is done first. This performs slightly worse in the barrel.
The number of pixel hit at the ends of the ladder in L1 is 30% higher than at the center. The formatted data size at the ends is 5% higher than at the ends.

Formatting uses the larger clusters at the edges efficiently.