CMS Silicon Strip Tracker FED Data Modelling in ORCA

Ivan D Reid

Brunel University

In collaboration with

Tomasso Boccali

Teddy Todorov

CERN

CMS Silicon Strip Tracker FED

The Physical Reality

Each Silicon Strip Tracker FED is contained on a single 9U VME board.
Each FED takes in 8 12-way optical-fibre ribbons.
Each optical fibre has multiplexed analogue signals from two APVs.
Each APV collects signals from 128 silicon strips.
(There will be ~18 FEDs per VME crate.)

This means potentially 96x256=24K data values per FED per selected bunch crossing.

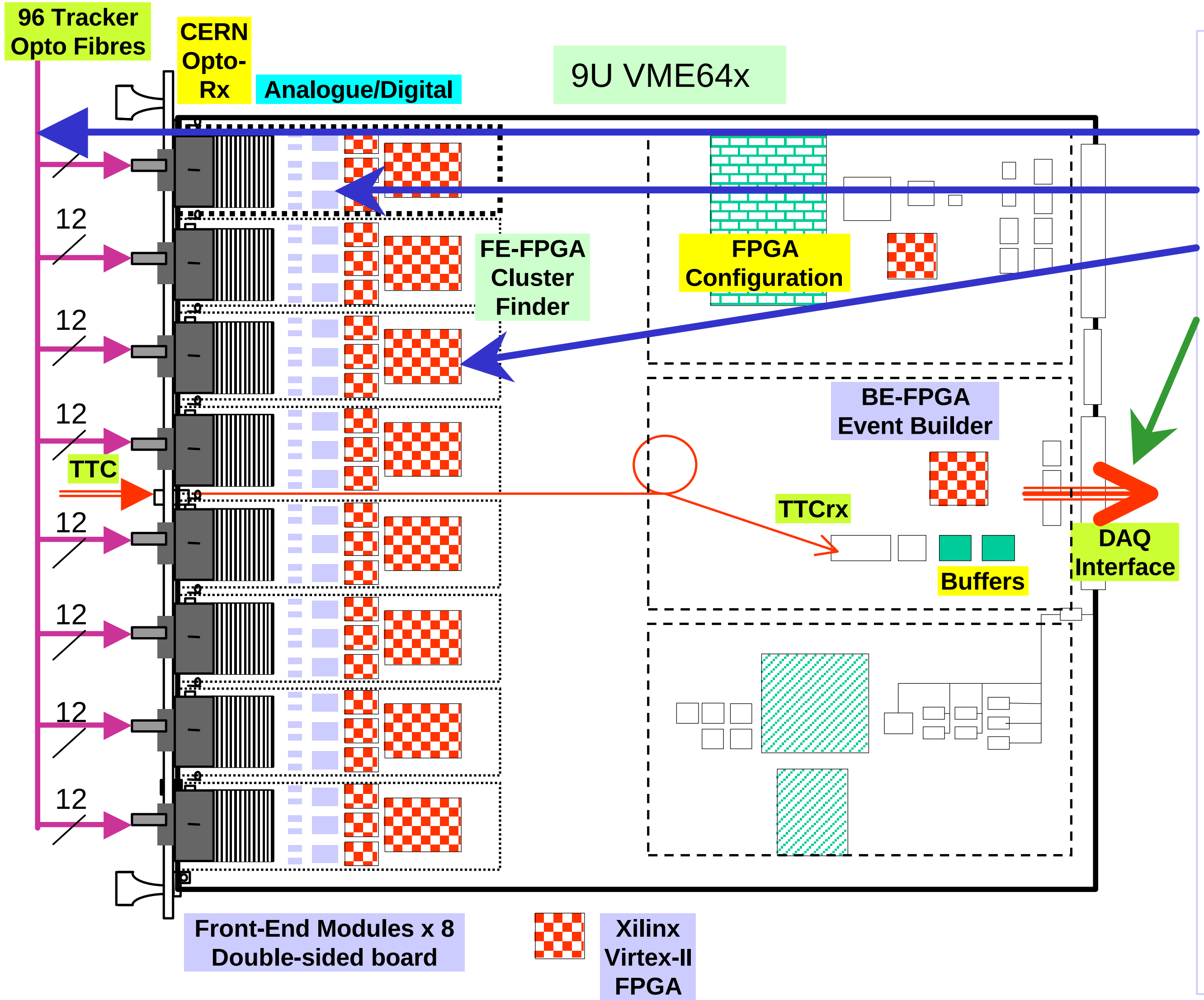
The FED:

- *Digitises each incoming data stream into 256 10-bit words;*
- *Subtracts individual pedestal values from each channel;*
- *Re-orders the data into physical order;*
- *Subtracts a common-mode base-line;*
- *Performs "zero-suppression" to reduce the amount of data (signals below a threshold are discarded);*
- *Packages the data and transmits them to the DAQ system.*

•The purpose of this exercise was to provide data-streams in the FED format, and also code for re-insertion of the data into ORCA.

CMS Silicon Strip Tracker FED

ORCA Software Model



ORCA

```
StripDigisByPairOfAPVs
SiStripFedDigitizer.cc
SiFedZeroSuppress.cc
```

TkSimEventObserver::

```
vector<unsigned char *>
*FEDDataStreams() const
{ return FEDStreams;}
//Makes the data available

size_t NumberOfFeds() const
{ if (FEDStreams != 0)
 return FEDStreams->size();
   else
 return 0;
} //Get no. of datastreams
```

Data Format

The FED can output data in four different formats

Scope Mode: Upon a trigger signal, up to 1020 10-bit samples will be made on each fibre. No re-ordering or pedestal subtraction.

<fibre1_length[7:0]><fibre1_length[11:8]><packet_code><raw_word0[7:0]><raw_word0[9:8]>…

<fibre2_length[7:0]><fibre2_length[11:8]><packet_code><raw_word0[7:0]><raw_word0[9:8]>…

Virgin Raw Data Mode: Incoming frames will have no pedestal subtraction, and not be re-ordered. (Data format as above.)

Processed Raw Data Mode: Incoming frames will have pedestals subtracted, and be re-ordered. No common-mode subtraction or zero-suppression. (Data format as above except strip data are 11 bits because of pedestal subtraction.)

Zero Suppression Mode: Incoming frames are fully processed and data truncated to 8 bits. (This is the data format we are creating.)

<fibre1_length[7:0]><fibre1_length[11:8]><packet_code> →

<median1[7:0]><median1[9:8]><median2[7:0]><median2[9:8]>→

<cluster start address><cluster length><cluster data 0><cluster data 1>… etc.

<cluster start address><cluster length><cluster data 0><cluster data 1>… etc.

<cluster start address><cluster length><cluster data 0><cluster data 1>… etc.

<fibre2_length[7:0]><fibre2_length[11:8]><packet_code>…

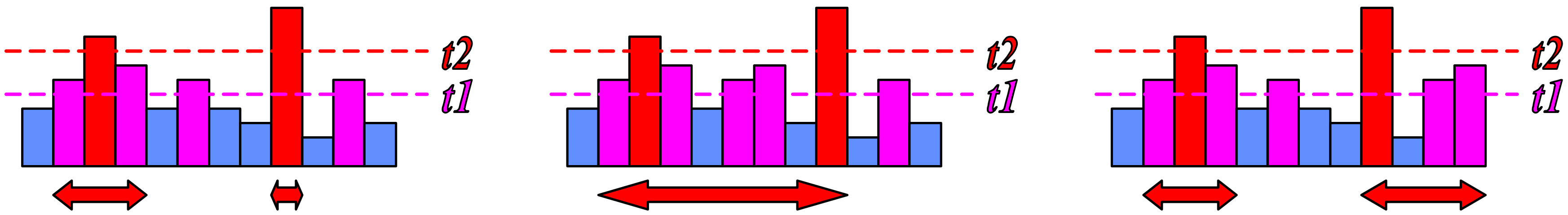
Zero Suppression

The Cluster Finding algorithm is defined as:

*All hits above **thresh1** are output, except single-channel clusters which must be above **thresh2** (where **thresh2** > **thresh1**)*

However, in order to cope with the needs of the Output FIFO Control block it is necessary to slightly modify this with the additional rule:

All clusters must be at least 2 strips away from every other cluster; any clusters violating this rule should be joined together.



NB: These clusters are different from those indicating track hits!

Data Types

Tommaso provided the following data types:

- ***typedef pair<int,int> DigiComponent;*** *// channel and value inside an APVPair*
- ***typedef vector<DigiComponent> APVPairDigis;*** *// vector of single strip signals*
- ***typedef pair<int,APVPairDigis> APVPairSignal;*** *//APVPair number and signal for it*
- ***typedef vector<APVPairSignal> FEDSignal;*** *// all the signal from the ROU*

Also provided:

- ***StripReadOutUnitAccessor::MasterTypeVector StripReadOutUnitAccessor::masterReadouts();*** *//returns a vector of FEDSignal*

So, all we have to do for each event is to iterate through the vector of ROUs, iterating through each of its APV pairs (packing them into 96-pair "physical" FED units), extracting the data for the hit strips as we go. Then the data are placed in memory as the data format specifies. As a cross-check, the data are then re-created and compared with the original values – this is a template for the data-reading subroutine (whose interface has not been specified yet).

Some Details

The Public Declarations: ***These are the only parts of the module available to other parts of ORCA:***

```
class TkSimEventObserver : Observer<G3EventProxy*> {
public:
  virtual void upDate(G3EventProxy* ev);          //Called to create the data-streams
  bool CheckFEDData(unsigned char *) const;       //Unpacker for a single FED
                                                  //(currently just verifies vs. original)
  vector<unsigned char *> *FEDDataStreams() const    //Makes the data available
                    { return FEDStreams;}
  size_t NumberOfFeds() const                     //Public access to the number of datastreams
       { if (FEDStreams != 0)
            return FEDStreams->size();
         else
            return 0;
       }
…
```

What remains to be done?

- *Definition of unpacker interface to ORCA and its implementation from the current* CheckFEDData() *template.*
- *Integration of the code into Giacomo Bruno's DAQ routines.*
- *Extension to full Virgin Raw Data format, especially in the context of Test Beam acquisition.*
- *Determination of header and trailer details and their implementation. Fine details of format: e.g. will zero-suppression clusters span the border between APV pairs (requires a cluster size of zero to represent a 256-strip cluster)?*
- *Mapping of APV pairs to individual FEDs and the concomitant reverse mapping from FEDs to detectors.*
- *Checking that ORCA's zero-suppression implementation matches the physical FED (e.g. do not omit data for single strips bridging clusters).*
- *Investigation of possible compression schemes to reduce further the amount of off-line storage needed for event data.*