Sorting of STS-XYTER2 data for microslice building for CBM experiment

MSc Marek Gumiński

Introduction

CBM experiment

CBM is a HEP experiment designed to study properties of matter at very high densities.

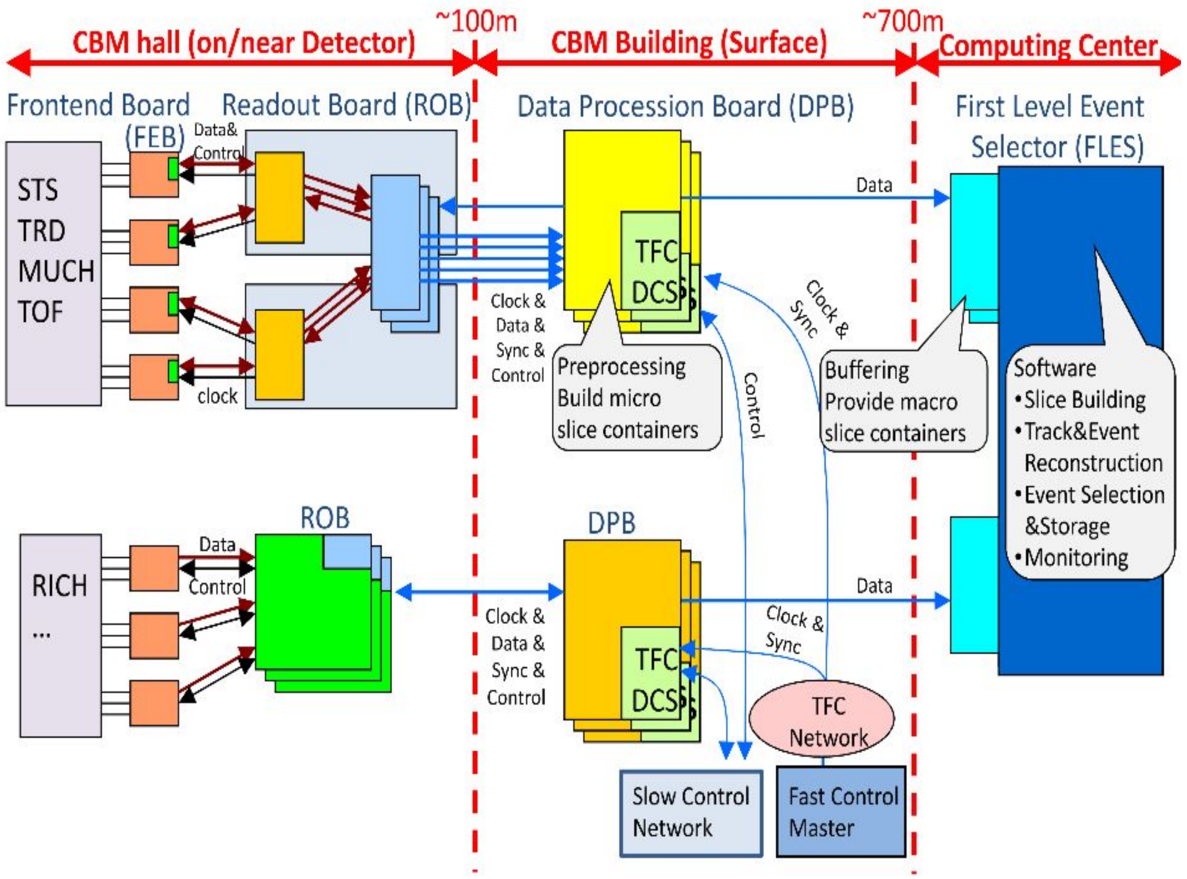
FEB are self triggered. They generate a stream of generally sorted data.

SX may send samples with a shift reaching 1024 frames.

DPB creates Micro Slices.

Data reconstruction and event selection in FLES.

DPB and FLIB will be replaced by the CRI.


~100m
~700m
CBM hall (on/near Detector)
CBM Building (Surface)
Computing Center
Frontend Board (FEB)
Readout Board (ROB)
Data Procession Board (DPB)
First Level Event Selector (FLES)
STS
TRD
MUCH
TOF
Data& Control
clock
TFC
DCS
Data
Clock & Data & Sync & Control
Clock & Sync
Control
Preprocessing Build micro slice containers
Buffering Provide macro slice containers
Software •Slice Building •Track&Event Reconstruction •Event Selection &Storage •Monitoring
RICH
...
ROB
DPB
Data
Control
TFC Network
Slow Control Network
Fast Control Master

[1]

Data aggregation architecture

Data aggregation throughput

The data aggregation module should have sufficient throughput to minimize the need for dropping any data.

It is worth to notice that the maximum throughput of the electronics may significantly differ from the throughput that may be generated by the experiment.

Throughput of the output channel is a natural limitation for the data aggregation module.

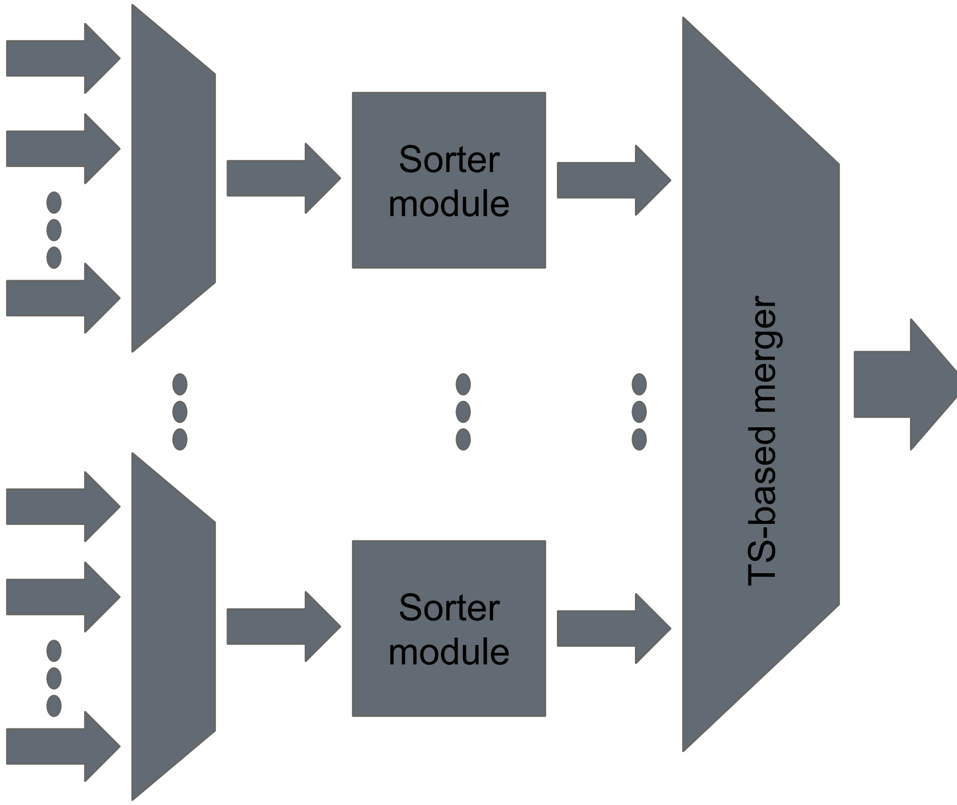
Throughput of aggregated data may exceed throughput of the sorting module, but in such case it is essential to have an overflow detection and notification method.

Aggregation and sorting architecture

We can divide the data aggregation module in reference to the sorting modules (assuming that they are the most complex part of the design):

1. aggregation of **I** input channels in order to fully utilize the sorter
2. sorting of data with a maximum delay of **D** samples
3. aggregation of **S** sorted channels in order to extend module throughput beyond single sorter capability

In some systems the I and S may be equal to 1, leaving only the sorter module.



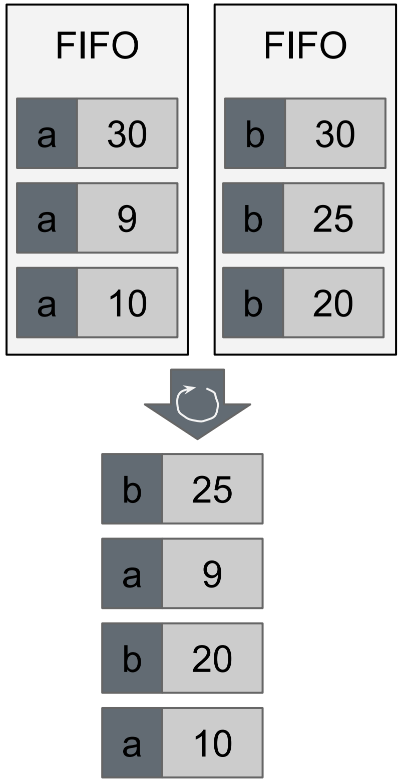
Initial aggregation

Merging unsorted data streams

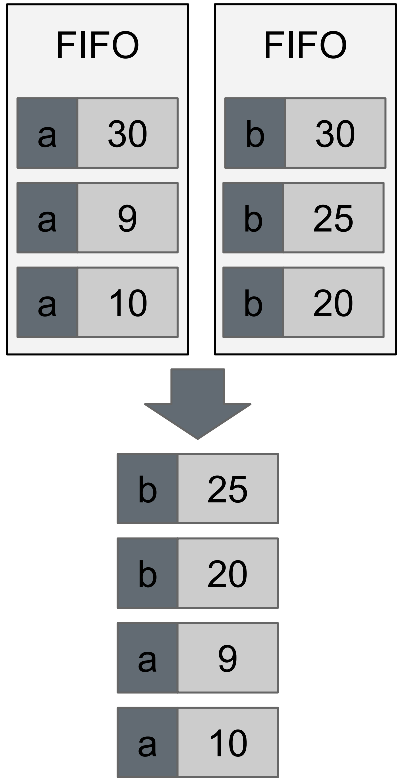
Data streams may be merged with simple round-robin algorithm, or (if the input is buffered) with priority algorithm, that reads the older sample first.

Maximum data shift of a merged stream is equal to the sum of maximum shifts of input streams.

Merging with priority algorithm may result in expected shift lower than with round robin-algorithm, but it doesn't affect the worst case.



Round robin



Priority algorithm

Sorting algorithms

Sorting algorithm for triggerless DAQ

A sample must be buffered for the number of cycles bigger or equal to the maximum data shift of the input stream.

The sorting resolution must be bigger or equal to the minimal MCS interval.

Sorter should have repeatable latency if the sorted streams are to be merged.

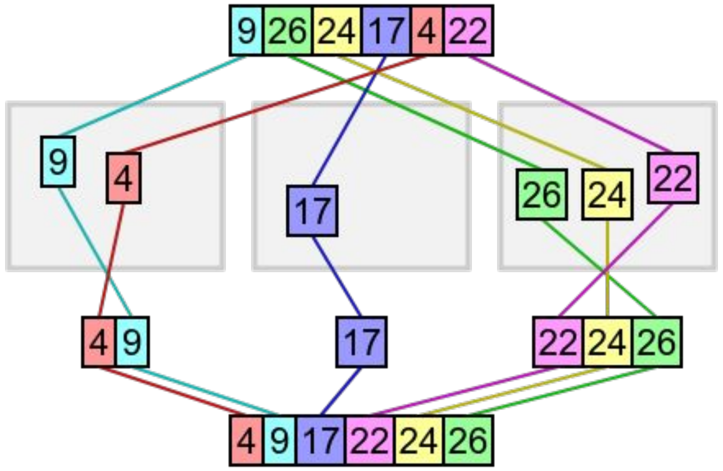
High throughput. The bigger the throughput of single sorter the more input channels may be sorted by a single sorter and the fewer sorters are needed in the aggregation module.

As memory usage is expected to be the biggest factor (at least with the detector used in the CBM) the efficient memory usage is essential.

Bucket sort

Bucket sort is a sorting algorithm that works by distributing the elements of an array into a number of buckets.

Each bucket should contain a specified range of values.



[3]

Bucket sort memory usage

Each bucket should have enough memory to buffer all hits that may be received in an MCS interval.

$$V = I * D + \frac{t_{MCS}}{\frac{1}{I*T}} = I * (D + t_{MCS} * T)$$

t_{MCS} - MCS interval
T - input channel throughput
V - bucket capacity

Such capacity would be needed in an unlikely situation that the detector saturates its buffers right after new MCS boundary, and a new sample is acquired whenever any buffer space is empty.

A number of buckets required by the sorting module depends on the relation between MCS interval and chip draining time.

Sum of the time periods stored in buckets must be bigger than the detector draining time.

$$t_{drain} = \frac{D}{T * N_{chan}} \quad n_{buckets} = \frac{t_{drain}}{t_{MCS}}$$

N_{chan} - number of links for detector

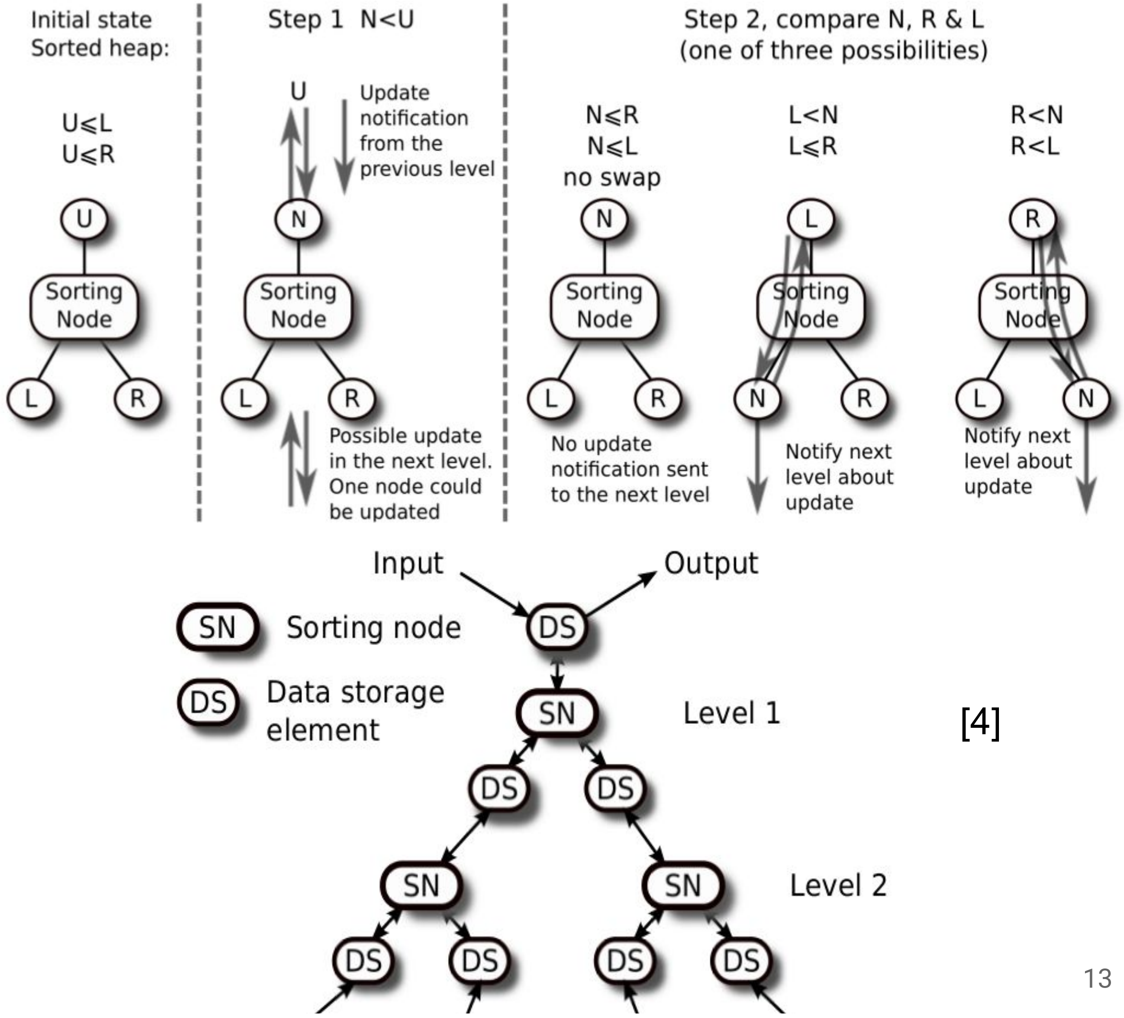
Heap sort

In each sorting cycle, we take one data record from the input data stream and compare it with the record available on the top of the heap.

If the input record is smaller or equal to the record on the heap's top, we output the input record.

If the input record is greater than the record on the heap's top, we output the record from the top of the heap,and put the input record on top of the heap.

In the second case, we need to reorder the heap.


Initial state
Sorted heap:
U⩽L
U⩽R
Step 1 N<U
U
Update notification from the previous level
Possible update in the next level. One node could be updated
Step 2, compare N, R & L
(one of three possibilities)
N⩽R
N⩽L
no swap
No update notification sent to the next level
L<N
L⩽R
Notify next level about update
R<N
R<L
Notify next level about update
Sorting Node
U
N
L
R
Input
Output
SN Sorting node
DS Data storage element
Level 1
Level 2

[4]

Heap sort implementation

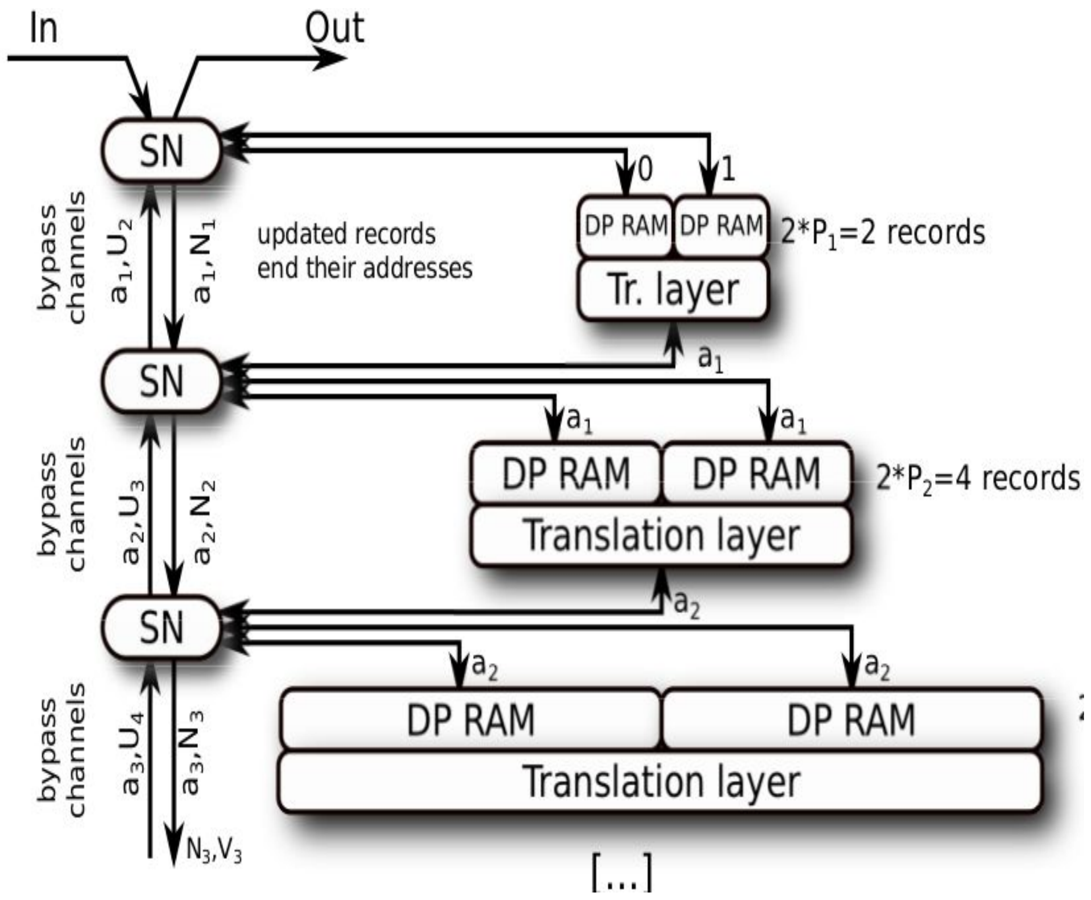
The main advantage of used heap sorter implementation is theoretically ideal memory utilization.

$$V = I * D$$

The FPGA implementation is limited by vendor-specific user blocks (e.g. Xilinx Kintex FPGA’s doesn’t have blocks smaller than 1024 36-bit words).

The maximum clock frequency of a 2048 sample sorter is around 160 MHz.

Sorter process one sample per two clock cycles.



[4]

Heap sort modifications

Block ram usage may be optimized by implementing top levels of the sorter in distributed memory.

In order to increase the maximum frequency of the sorter, the least significant bytes of the sorting key may be ignored. Such implementation would result in kind a kind of bucket sorting.

Init and flushing samples, that were present in the original implementation, were removed in order to simplify sorting nodes and increase the maximum frequency.

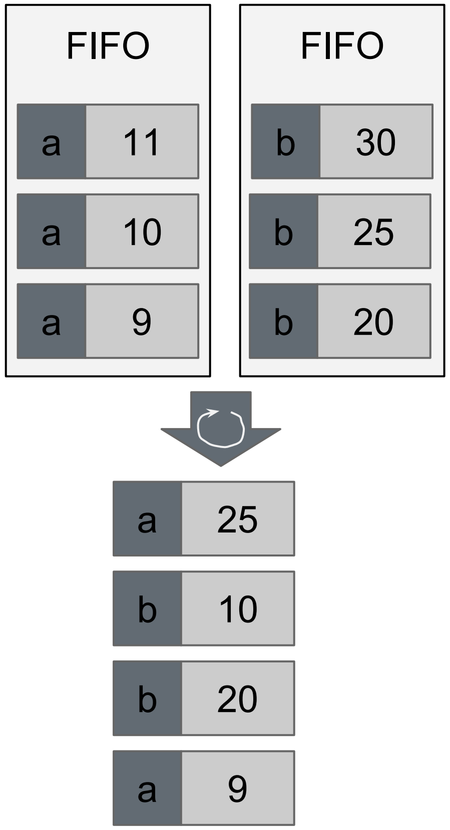
Sorted data stream aggregation

Merging sorted data streams

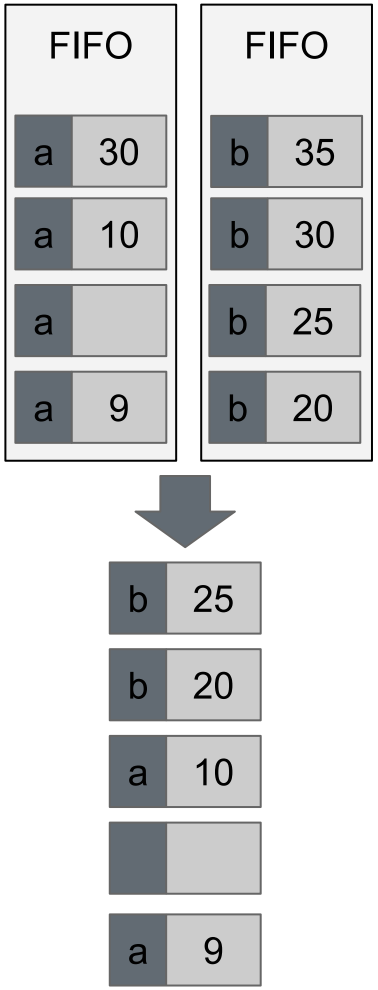
Merging of sorted data streams with round-robin algorithm may lead to the generation of a unsorted stream.

It is possible to merge the streams correctly:

- streams must be buffered
- the merger must wait for a valid sample in all the buffers



Round robin



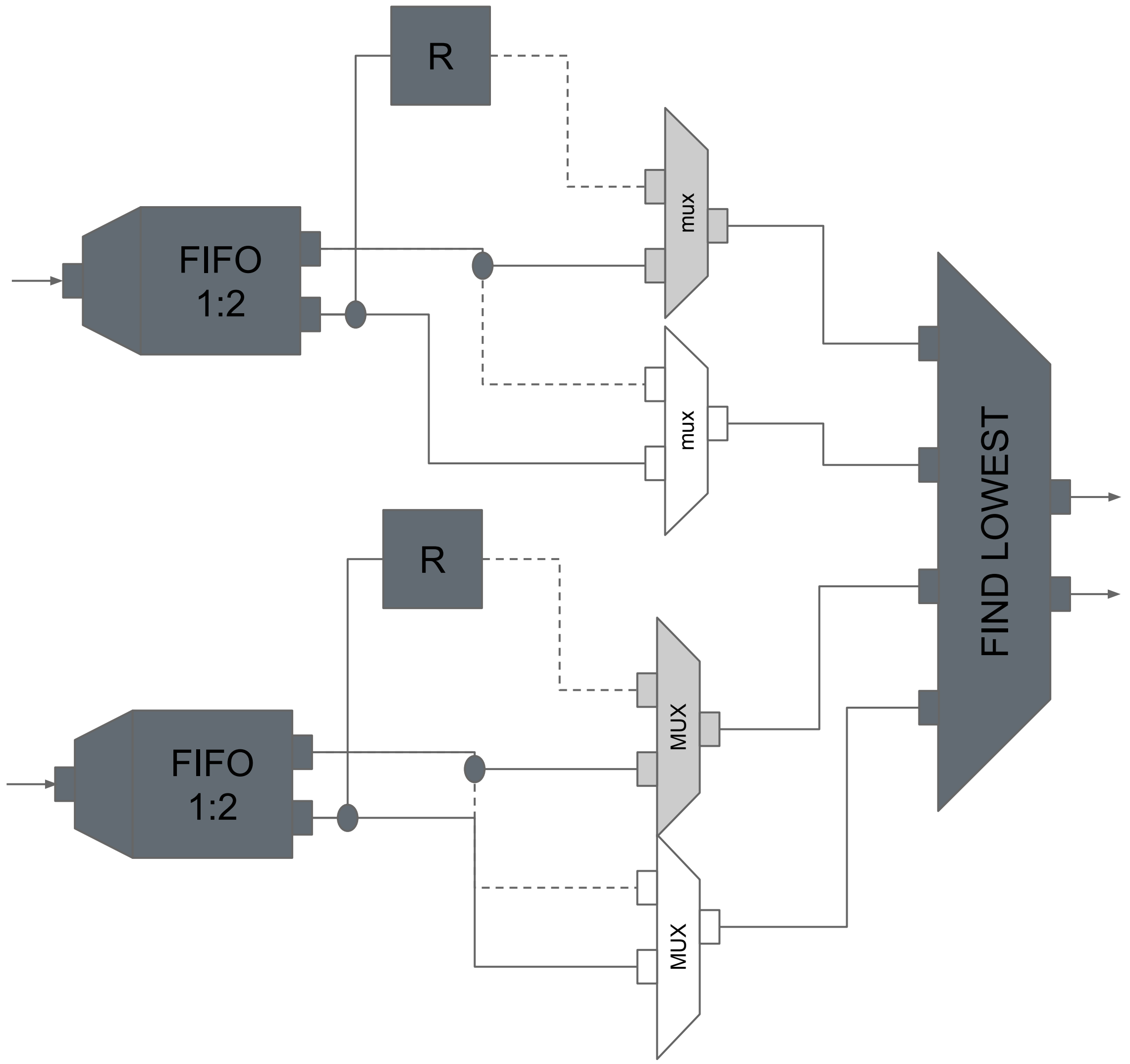
Timestamp-based

Merging throughput limitation

The throughput of classical mergers processing single sample per clock period is limited by the FPGA maximum frequency.

It is possible to implement a merger that would output multiple samples per clock cycle, thus achieving the same throughput with much lower clock frequency.

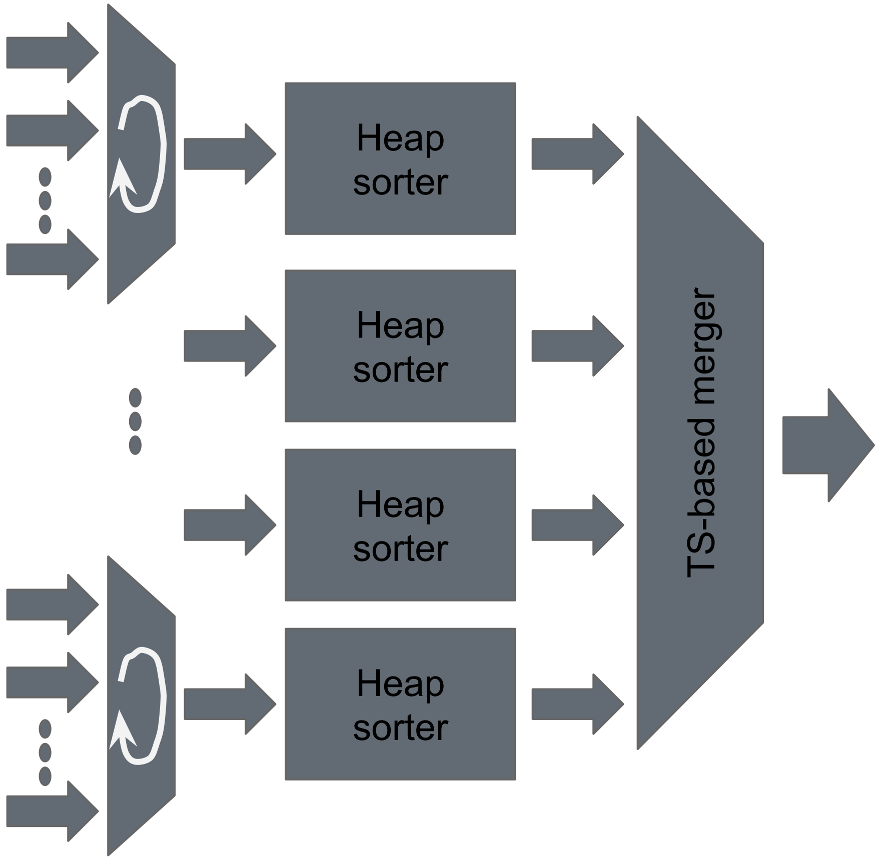
Binary merger processing 2 samples per cycle was described in [5].



CBM implementation

Architecture of CBM data processing module

- Module throughput: 320 Mhit/s - throughput of the output channel
- 4 heap sorters working in parallel
- Sorter throughput: 320 Mhit/s (4*80Mhit/s)
- Input merging: 2 SX’s (10 channels) per sorter
- Maximum data delay: 2*1024 = 2048
- Sorters capacity: 2048
- Aggregated SX’s: 8 - the number of channels transmitted via single GBTx link



Current status

Sorting module is working in a simplified version of the readout chain (without GBTx).

Sorting module was proven to work in simulation, laboratory test and in beam test at CERN.

GBTx version of the readout is currently being tested.

Future plans

Timing issues when implementing data processing modules (FPGA resource utilisation close to 80%).

Possible change to architecture with one sorter per detector. This way required clock frequency and sorter capacity would decrease.

Current sorted stream merging would double it’s memory requirements leading to the general increase of memory usage.

Multi-input sorter is currently under development.

Output channel of the planned CRI board is the PCIe bus. This change may lead to significant changes in data processing module requirements.

Thank you for your attention!

Sources

[1] - J. Lehnert et al 2017 JINST 12 C02061

[2] - K. Kasinski et al 2016 JINST 11 C11018

[3] - https://www.growingwiththeweb.com/2015/06/bucket-sort.html

[4] - W. Zabołotny, "Dual port memory based Heapsort implementation for FPGA", Proc. SPIE 8008; doi: 10.1117/12.905281;

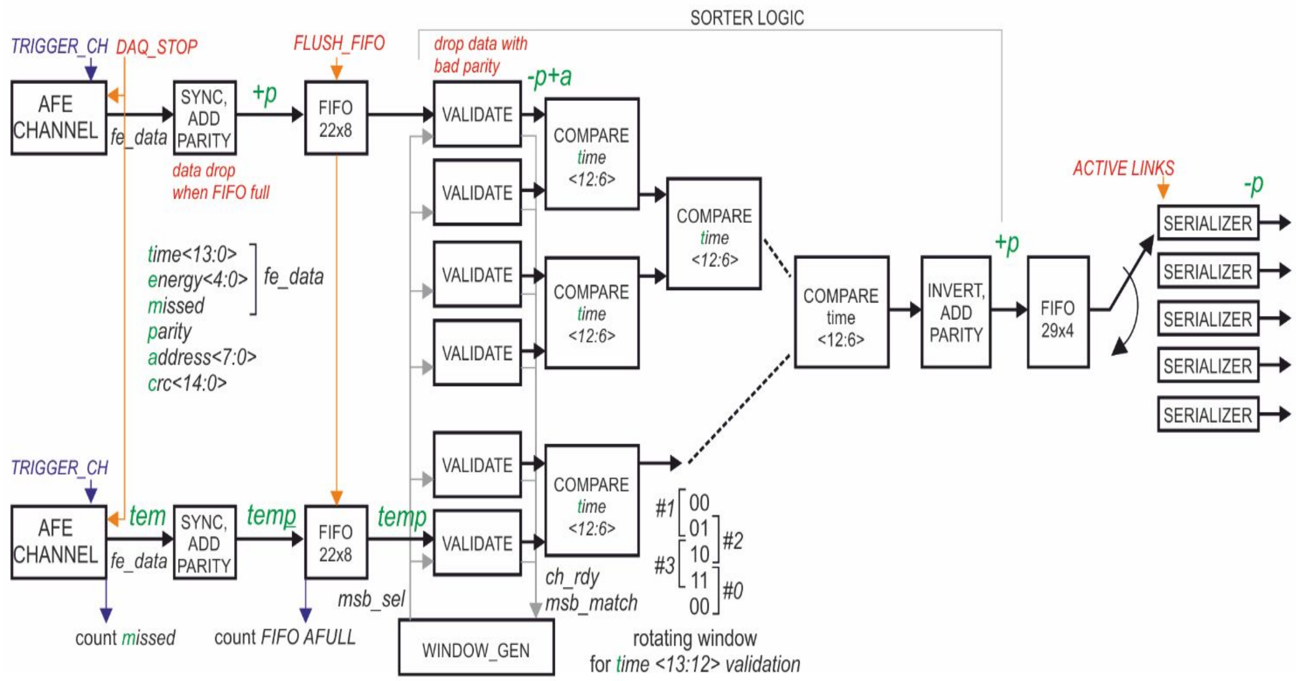
[5] - M. Gumiński, “High-speed concentration of sorted data streams for HEP experiments” - Nica Days 2017

Backup

STS-XYTER2

- Radiation hardened ASIC with 128 independent readout channels.
- Self-triggering.
- Hit amplitude is measured with Time over Threshold technique.
- Timestamping resolution of 3.125 ns
- Each channel contains 8 sample FIFO
- SX sends the data via up to five 320 Mb/s serial elinks.

- **The output stream of generally increasing TS**
- **Delay introduced by the TOT and stream merging result in possible data shift of 1024 samples (96 us)**



[2]