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

Marek Gumiński[*], **Wojciech M. Zabołotny, Adrian P. Byszuk, Krzysztof Poźniak**

Institute of Electronic Systems, Warsaw University of Technology, Poland
E-mail: marek.guminski@gmail.com, wzab@ise.pw.edu.pl, adrian.byszuk@gmail.com, pozniak@ise.pw.edu.pl

The article describes the architecture of the data sorting and aggregation module for the triggerless high energy physics experiment data acquisition.
The described module is able to revert shifts in the input data order, which are introduced by the detector readout modules. The sorted data is aggregated into a single stream and then divided into packages containing samples acquired in consecutive time intervals.
The presented article describes the usage of the heap sorting algorithm for restoring the correct data order. Data stream aggregation techniques, used in the presented module, are described. The paper contains a block diagram of the presented module and resource utilization summary generated after implementation.

Topical Workshop on Electronics for Particle Physics (TWEPP2018)
17-21 September 2018
Antwerp, Belgium

[*]Speaker.

1. Introduction

Compressed Baryonic Matter (CBM) is a triggerless High Energy Physics experiment. Such experiments do not use a common trigger signal. Instead, the detector front-end boards are self-triggered independently. The Data Acquisition Chain (DAQ) forwards all the data to the First Level Event Selector (FLES), that performs the event reconstruction and selection in software.

The presented paper describes an aggregation and sorting module (ASM), part of a Data Processing Board (DPB) firmware. The DPBs are data concentrator boards [1] located in the central part of CBM DAQ.

The DPB receives the data from the front-end boards via GBTx [2] interface. Each GBTx uplink is connected to one link of the STS-XYTER2 (SX) readout chip [3]. The data stream from SX is usually time sorted with its internal sorter. However, under high hit rate, the internal sorter may be not able to ensure proper hit order, and shifts in the data stream may occur.

The ASM is used to reorder the data according to their acquisition time and create so-called Micro Slices (MCS) transmitted to the FLES. Each MCS must contain all data acquired by the front-end boards connected to the DPB in a certain period of time.

The ASM should provide possibly high data throughput. The required number of DPB boards in the DAQ chain depends on the throughput of the ASM.

Similar modules were designed for other triggerless detectors. The thesis [4] mentions a sorting module for the Mu3e experiment. That module uses a bucket sorting algorithm using bins of a predefined size. Article [5] describes a variation of a bucket sort that may store a variable number of hits in each bin but requires the distributed memory for storage. Such implementation is only possible in systems with low data throughput and maximum shift because the distributed memory is a scarce resource in FPGA's.

2. Architecture

The input data throughput of ASM depends on the number of connected SX links. The output data throughput is limited by the throughput of the FLIM interface equal to 10 Gb/s which is equivalent to c.a., 320 millions of hits per second (Mhits/s). Because the MCS can't be split between multiple FLIM links, it is not possible to increase the output throughput of a single AMC. However, the total DPB throughput may be increased by using multiple ASMs working in parallel and connected to separate FLIM links.

To maximize the throughput, the input data are delivered to four sorting modules working in parallel. The number of input links connected to each sorter is defined based on the throughput analysis (see section 4). The sorted streams are then merged in a special manner, preserving the time order, to generate Micro Slices. A simplified block diagram of the final architecture is presented on fig. 1.

3. Sorting

In the bucket sort implementations based on block memory, the data is split into buckets of a certain capacity. When the system works at a high data rate (even due to short-term fluctuations),

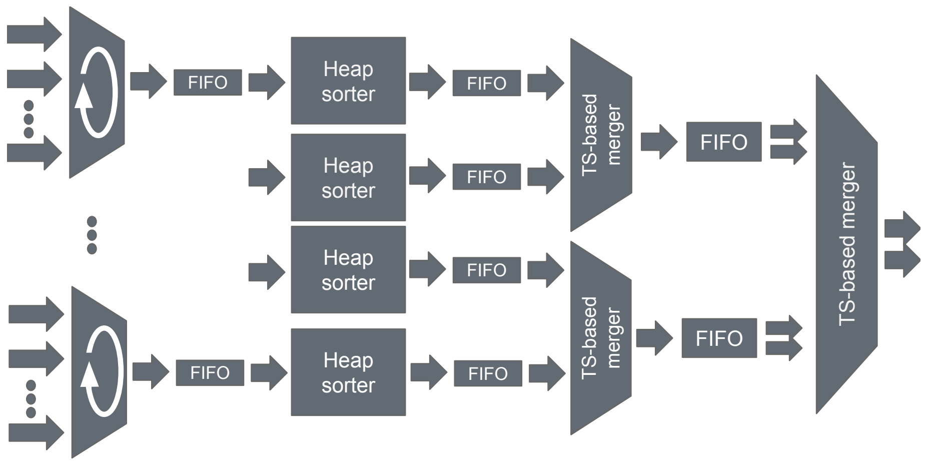

Figure 1: Block diagram of the aggregation and sorting module.

all the hits exceeding the bucket capacity must be dropped. That problem is especially important if the input stream contains multiple hits with the same timestamp. To limit the data loss in such conditions, we have decided to use a parallel implementation of the heapsort algorithm [6].

The original implementation of the heap sorter uses the FPGA BRAM to buffer the data. The sorter outputs the oldest stored sample each time a new sample arrives, but not more than one sample per two clock cycles. The maximum clock frequency of the sorter is around 160 MHz, resulting in a throughput of around 80 Mhit/s.

The maximum shift that the heap sorter is able to revert is equal to the number of samples that the sorter can store (its capacity). The capacity of sorters used in the ASM is equal to the maximum data shift, that may be introduced by the SX's.

The heap sorter in the ASM uses the whole timestamp of the hit as a sorting ensuring a perfect sorting. That is not required by the MCS specification but simplifies the event reconstruction (done in the FLES).

In the Xilinx Kintex 7 FPGA used in the DPB, the minimal memory block size is 1k of 36-bit words. Using such blocks for the top levels of the heap (that have a capacity of a few samples) would be a waste of resources. Therefore, the ASM implementation of the heap sorter uses the distributed memory for 5 topmost levels of the heap.

The memory utilization of the ASM may be limited by lowering the sorter capacity. However, that results in a necessity to drop extremely delayed samples.

Implementation of multiple relatively complex ASM blocks in the FPGA may result in timing closure problems. That may be cured by using only some most significant bits of the timestamp as a sorting key. That reduces the size of comparators and decreases the FPGA resource consumption. Less significant bits of the timestamp may be stored as part of the payload and used to restore the TS. The resulting algorithm is similar to the bucket but avoids the problem limited bucket capacity.

4. Initial merging

To fully utilize the throughput of the heap sorters, the ASM aggregates 10 input links (each with a throughput of c.a., 10 MHit/s) connected to 2 SXs. The theoretical maximum throughput of the merged stream exceeds the maximum throughput of a sorter by around 20%. Short transgressions of rate are filtered out by the FIFO's located between the initials mergers and the sorters. If the FIFO capacity is not sufficient, the excess data is dropped and the DAQ control module is informed.

Merging of unsorted data streams produces the output stream with a maximum shift equal to the sum of maximum shifts in the input streams. However, in the proposed architecture, the 10 input streams are not fully independent as they come from 2 SX chips. Therefore, merging them results in the maximum shift twice higher than the maximum shift from a single SX.

5. Merging of sorted data

The throughput of the ASM should reach the throughput of the output link. That would be difficult to achieve with a single bucket sorter and is impossible with a single heap sorter. Therefore, to match that assumption, we used four heap sorters working in parallel.

The sorted data streams must be merged with preserving the time order. For that, a dedicated component that outputs the oldest of the input samples (the one with the smallest timestamp) has been created. The inputs are buffered with FIFO's to compensate for possible delay shifts between the merged streams. The merging of the sorted streams is done in two stages (fig. 1).

First, two pairs of the data streams (each up to 80 Mhits/s) are merged with mergers that output a single (older) hit per 160 MHz clock cycle. Then, two resulting streams (up to 2 x 160 Mhits/s) are merged into a single sorted stream providing 2 samples per 160 Mhz clock cycle. Doing it with a simple merger requires running at 320 MHz clock that exceeds the capabilities of Kintex 7 FPGA used in DPB. Therefore, a modified merger, that can output two samples per 160 MHz clock cycle, was implemented [7]. The component buffers two input data streams in FIFO's with a double-width output. Then, an interface adapter selects two oldest samples from the four, available at both FIFO outputs.

6. MCS generation

The MCS time period is configurable (in clock periods) during the operation. The MCS generation module uses a counter, incremented by the MCS period value, to determine the MCS boundaries in the final sorted data stream. The MCS is closed when the first hit from the next MCS is received in the sorted data stream.

7. Time update

All the components in the ASM rely on a constant data flow: i.e. sorter outputs a sample when it receives a new one, the MCS generator closes the package when the first hit from the next one arrives. To prevent long freezes of data flow during operation of DPB at low data rate, the "dummy hits" were introduced in the design.

The dummy hit is added to the SX data stream if no real hit was received for certain defined time. Dummy hits are assigned the timestamp equal to the timestamp that should be assigned to

Table 1: ASM resource utilization.

Slice LUTs	**Slice Registers**	**Slices**	**Block RAM Tiles**
24109	11149	6741	77.5

the real hit received at the same time. The DPB is the source of the SX reference clock and reset signals, and therefore it can keep a copy of the FE chips timestamp counter. The copy of the counter is slightly offset from the real one, but that offset may be estimated and compensated for.

8. Results

The ASM is the dedicated module with parameters based on requirements of the CBM DAQ chain. The module contains 4 heap sorters working in parallel with 160 MHz clock processing up to 40 SX links in total. The sorted streams produced by the sorters are merged into the sorted stream providing 2 data samples in each 160 MHz clock period. That stream is finally split into Micro Slices containing the samples from consecutive predefined time periods. The design almost fully uses the capability of high-speed data processing offered by Kintex 7 FPGAs. The special, non-standard solution was needed in the last stage of the stream merger to fulfill the throughput requirements. The developed ASM is capable of aggregating and packaging the experiment data with the maximum throughput of the output link.

The design was successfully synthesized and verified in hardware. Table 1 shows the FPGA resource utilization of the presented design.

References

[1] W. Zabolotny, G. Kasprowicz et al., *Versatile prototyping platform for Data Processing Boards for CBM experiment*, *Journal of Instrumentation* **11** (Feb., 2016) C02031–C02031.

[2] M. B. Marin, S. Baron et al., *The GBT-FPGA core: features and challenges*, *Journal of Instrumentation* **10** (Mar., 2015) C03021.

[3] K. Kasinski, R. Kleczek, P. Otfinowski, R. Szczygiel and P. Grybos, *STS-XYTER, a high count-rate self-triggering silicon strip detector readout IC for high resolution time and energy measurements*, in *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, (Seattle, WA, USA), pp. 1–6, IEEE, Nov., 2014. DOI.

[4] A.-K. Perrevoort, "Sensitivity Studies on New Physics in the Mu3e Experiment and Development of Firmware for the Front-End of the Mu3e Pixel Detector." http://archiv.ub.uni-heidelberg.de/volltextserver/id/eprint/24585, 2018. 10.11588/heidok.00024585.

[5] J. Saini, S. Mandal, A. Chakrabarti and S. Chattopadhyay, *A real time sorting algorithm to time sort any deterministic time disordered data stream*, *Journal of Instrumentation* **12** (Dec., 2017) P12023–P12023.

[6] W. M. Zabołotny, *Dual port memory based Heapsort implementation for FPGA*, *Proc. SPIE* **8008** (June, 2011) 80080E.

[7] M. Gumiński, W. Zabolotny, A. Byszuk and K. Poźniak, *High-speed concentration of sorted data streams for hep experiments*, *Acta Physica Polonica B Proceedings Supplement* **11** (01, 2018) 689.