HIGH-SPEED CONCENTRATION OF SORTED DATA STREAMS FOR HEP EXPERIMENTS*

Marek Gumiński[a], Wojciech M. Zabołotny[a,b]
Adrian Byszuk[a,b], Krzysztof Poźniak[a,b]

[a]Institute of Electronic Systems
Faculty of Electronics and Information Technology
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warszawa, Poland
[b]Institute of Experimental Physics, Faculty of Physics
University of Warsaw
Pasteura 5, 02-093 Warszawa, Poland

(Received July 23, 2018)

Presented paper describes the data stream sorting and merging architecture, fitting triggerless HEP experiments. The presented architecture is implemented in FPGA. It is capable of merging multiple sorted data streams into a single output stream of up to 320 Mwords/s throughput.

DOI:10.5506/APhysPolBSupp.11.689

1. Introduction

The presented high throughput data stream concentration module was created as a part of Data Processing Board for the CBM experiment [1]. DPB is a part of CBM prototype readout chain [2] responsible (among other things) for aggregation of data from multiple Front End Electronics (FEE) chips called STS-XYTERS (SX) [3].

SX produces a single data sample per hit and sends it to DPB via multipurpose high-speed GBTx [4] protocol. Each SX sample is tagged with a time stamp (TS), based on the internal counter. Due to time over threshold acquisition method, high-energy hits are sent with a bigger latency than low-energy ones, resulting in the data stream that is only coarsely sorted. Each SX stream is strictly sorted in the DPB firmware (the TS of each sample is bigger or equal to the TS of previous samples). Multiple low throughput detector data streams are merged into a single stream in the DPB and sent over 10 Gbit FLES link to the event building and hit reconstruction unit.

* Presented by M. Gumiński at the II NICA Days 2017 Conference associated with the II Slow Control Warsaw 2017, Warsaw, Poland, November 6–10, 2017.

To fully utilize the output link capacity, the data aggregation component must have the throughput of 320 Msamples/s, which is unreachable for a common binary merger (BM). The maximum BM throughput is equal to its clock frequency, that is limited by FPGA hardware to *ca.* 250 MHz.

2. Stream merging

The two-part notation will be used to visualize data values organized into streams throughout this presentation: the capital letter identifies the data stream, and the digit stands for TS value. Actual data content is irrelevant from the sorters point of view. Thus, it is omitted.

Figure 1 illustrates how two streams may be merged. The output stream shown on the top is correctly merged, TS is non-descending.

The above example shows a few basic requirements regarding stream merging:

— The output stream throughput is a sum of input stream throughputs. If one sample is processed in each clock cycle, then either the clock frequency of the output stream must be higher than the sum of input frequencies or input streams must contain some "invalid" data.

— When multiple samples must be taken in sequence from a single input stream (A0, A1, and B1, B1, B1), samples from the other inputs must be buffered. A FIFO on each input stream not only enables the correct merging but may also help to cope with short periods of the increased data rate.

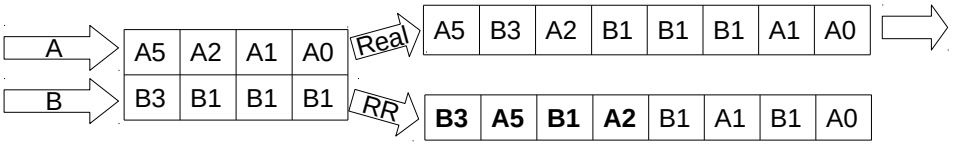

Fig. 1. Sorted stream merging. The invalid output on the top was generated by round-robin algorithm. Correct output is shown below.

The lower data stream presented in Fig. 1 illustrates that simple round-robin algorithm fails to produce the sorted data stream. A dedicated component must be used instead.

3. Binary merger

The binary merger is a standard solution for stream merging. It outputs the older from two input samples per clock cycle, so its maximum throughput is equal to its clock frequency. The merger must wait for both FIFOs to be

not empty before outputting sample. Otherwise, the output sample may be newer than the sample that will appear in the previously empty FIFO. Such situations are common, because although both input streams are generated from synchronous sources, certain TS value in one stream may arrive at the merger significantly later than the same value in the other stream.

Merging more than two streams with a single merger is possible, but unsuitable for applications optimized for maximum throughput. Such merger would have to compare multiple TS values in a single clock cycle, which generates longer logic delay path. The binary merging tree, as shown in Fig. 2 (left), should be used instead. Since the tree is made of identical binary mergers, its maximum frequency is equal to the maximum frequency of single BM. Binary merging tree disadvantage is that it requires more memory for FIFO's.

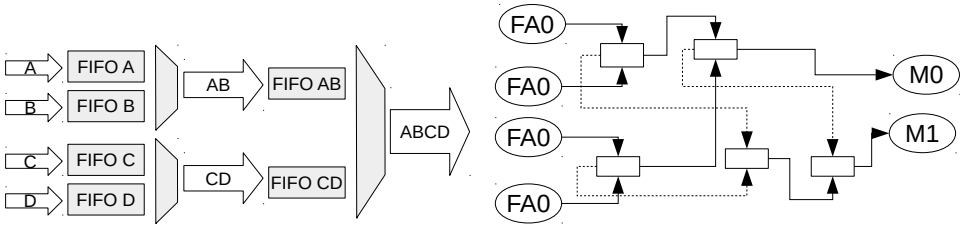

Fig. 2. Left: Binary tree created with binary mergers. Right: Selecting 2 oldest out of 4 samples. Squares represent comparators that have inputs on top and bottom. Bigger sample is output on the right-hand side, smaller on the left-hand side.

Binary merger tree was a first candidate for stream merging in DPB for CBM, but it turned out that reaching the timing closure was not possible. To simplify the task, most of the merging layers were driven by 160 MHz frequency, only the final layer used 320 MHz.

4. Multisample binary merger

4.1. Concept

Since increasing clock frequency is not sufficient to fulfill the DPB's throughput requirement, another approach had to be taken. Running multiple BMs in parallel would result in multiple sorted, but unrelated data streams, which is contrary to the initial assumptions. Outputting multiple samples in a single clock cycle is the only solution.

We have decided to output two samples per 160 MHz clock cycle, which seems to be a good compromise between clock frequency and design simplicity. Two input streams were used, because binary trees may be created with Multisample Binary Mergers (MBMs) as well as with BMs.

The MBM must select two oldest samples from its inputs in each clock cycle, so two oldest sample from each input must be readable by the merger. The 32-bit input, 64-bit output FIFOs must be used for input stream buffering. The sample on more significant bits of FIFO output is the one that was written earlier, so with sorted input stream, it is the older one. We think that it is more natural to read least significant bits first, so in this paper, FA0 will denote the older sample on FIFO A output, and FA1 will denote the newer one.

In a general case, the selection of two oldest samples require 5 comparators in 3 layers as shown in Fig. 2 (right). By utilizing the fact that input streams are sorted, we know that

$$\mathrm{FA1} \;\leq\; \mathrm{FB0} \equiv \mathrm{FA0} \leq \mathrm{FA1} \leq \mathrm{FB0} \leq \mathrm{FB1}\,, \tag{1}$$

$$\mathrm{FB1} \;\leq\; \mathrm{FA0} \equiv \mathrm{FB0} \leq \mathrm{FB1} \leq \mathrm{FA0} \leq \mathrm{FA1}\,. \tag{2}$$

Therefore, two samples from FIFO A must be output if Eq. (1) is true, two samples from FIFO B must be output if Eq. (2) is true, if none of them is true, one sample from each FIFO must be output (require additional comparison to determine which one is older). All three comparisons may be done in parallel.

4.2. Implementation

Traditional First Word Fall-Through Block RAM FIFO is not suitable as MBM input FIFO, because the read operation updates all output bits. The MBM requires the possibility of reading the older output word without discarding the newer one. Such operation is possible in distributed memory FIFO, but buffering a few hundred of samples with it (required for correct merging) would be a huge waste of FPGA resources. We have decided to use a "short" distributed memory FIFO, to provide the suitable interface, connected to BRAM FIFO output, used for actual buffering. For two samples, the MBM "short" FIFO is reduced to a single register as shown in Fig. 3.

The multiplexer on the MBM input allows it to see either Fx0 and Fx1 or Rx and Fx0. Fx0 and Fx1 are seen when Rx is empty. Fx1 is stored in Rx when two sample read is issued, while Rx is not empty. Otherwise old Fx1 (that was not seen by MBM) would be overwritten by FIFOs updated and lost. BRAM FIFO is read if less than 2 unused samples are available on output. In particular, BRAM is not read when MBM issues a single sample read with not empty Rx. In such a case, Rx is cleaned and the multiplexer is switched to Fx0 and Fx1.

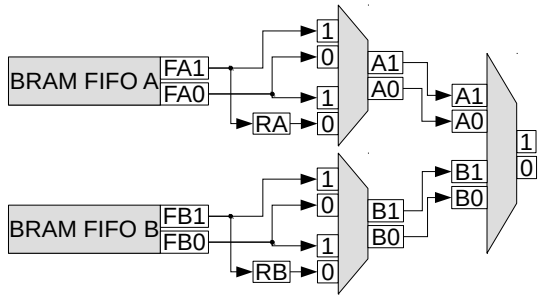

Fig. 3. Multisample Binary Merger block diagram.

5. Results and conclusions

The MBM and BM FPGA resource utilization are shown in Table I. Both components work with 32-bit data at 160 MHz clock, but MBM throughput is equal to 320 Msamples/s because its capable of processing two samples per clock cycle.

TABLE I

FPGA resource utilization.

Merger	Slice LUT	Slice register	BRAM
Binary	149	270	2
Multisample	184	306	2

CBM DPB sorting and merging component was implemented on the AFCK board and tested in a simple setup resembling final CBM readout chain.

Generation of strictly sorted microslices without creating a bottleneck inside DPB, should mitigate computing power required to create time slices in FLES, possibly leading to lowering cost of FLES hardware.

REFERENCES

[1] CBM Collaboration, *Eur. Phys. J. A* **53**, 60 (2017).

[2] J. Lehnert *et al.*, *JINST* **12**, C02061 (2017).

[3] K. Kasinski, R. Kleczek, *Microelectron. J.* **46**, 1248 (2015).

[4] The GBT project, `https://espace.cern.ch/GBT-Project`, accessed: 2018-03-01.