Development of Prototype for the Streaming Data Acquisition System of the Charmed Baryon Spectroscopy at J-PARC

T.N. Takahashi[1], H. Asano[2], T. Akaishi[3], T. Aramaki[4], M. Chang[5], W.-C. Chang[5], R. Honda[4], Y. Igarashi[6], T. Ishikawa[7], S. Kajikawa[4], Y. Ma[2], K. Nagai[5], H. Noumi[1,6], H. Sako[8], and K. Shirotori[1]

[1] *Research Center for Nuclear Physics (RCNP), Osaka University, Ibaraki, Osaka 567-0047, Japan*

[2] *RIKEN Nishina Center for Accelerator-Based Science, Wako, Saitama 351-0198, Japan*

[3] *Department of Physics, Osaka University, Toyonaka, Osaka 560-0043, Japab*

[4] *Department of Physics, Tohoku University, Sendai, Miyagi 980-8578 Japan*

[5] *Institute of Physics, Academia Sinica, Taipei 11529, Taiwan*

[6] *High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki 305-0801, Japan* [7] *Research Center of Electron Photon Science (ELPH), Tohoku University, Sendai, Miyagi 982-0826, Japan*

[8] *Japan Atomic Energy Agency (JAEA), Tokai, Ibaraki 319-1195, Japan*

The J-PARC E50 experiment will investigate the diquark correlation by a precise measurement of the charmed baryon spectroscopy. The diquark correlation is an important internal degree of freedom of hadrons. Charmed baryons are measured in a missing mass spectrum of the $p(\pi^-, D^{*-})$ reaction, where high-intensity pion beam of 30 MHz is irradiated on the liquid hydrogen target and a high interaction rate of greater than 3 MHz is expcected. In such a high multiplicity environment, a traditional hardware trigger system will not effciently work. Instead, a novel approach of a trigger-less data acquisition (DAQ) system, or streaming DAQ system, is adopted for the E50, where the entire event selection is performed at the software level with fast online filtering and tracking algorithms. In 2018, we developed a prototype of both hardware and software for the cocenpt validation of the streaming DAQ.

Hardware

In the streaming DAQ, all detector hits should have timestamp for event reconstruction. Therefore, we developed a streaming TDC module with time synchronization capability. The prototype was implemented on the hadron universal logic (HUL)[2]. The firmware consists of two blocks, the online data processing and the data transfering block. In the online data processing block, both leading and trailing edges timing of the input signal are independently measured by TDC units. The TDC data are composed of a clock counter of 130 MHz base clock and a fine counter to interpolate the clock interval. For the fine counter measurement, four 260 MHz clock signals are equally shifted by 90 degrees so as to obtain the time precision better than 1 ns. Then, the leading edges and trailing edges are paired and the time-over-threshold (TOT) of the pair is calculated for signal-noise separation and slewing correction. Each hit information includes a channel ID, timestamp, and TOT, which are encoded into 5 bytes in the present design. The data transfering block merges TDC hits into a double-buffer. The buffer is switched by a heartbeat signal, which is a carry signal of 16-bit counter driven by a 130 MHz base clock. A special data word called a heartbeat frame (HBF), which contains a counter of the heartbeat signal, is inserted every heartbeat cycle ($\sim$500 μs), while a spill-end frame is also inserted at the spill-end timing. The HBF plays an important role in the streaming TDC becuase it works not only as a data separator but also as an offset value to calculate the absolute time from the start timing of each spill. Figure 1 shows a schematic drawing of the data stcuture from the 6 TDC modules in consecutive 2 spills. The buffered data are sent to a PC with 1G Ethernet. Since data size of each hit is 5 bytes, one TDC can cope with a maximum hit rate of $\sim$20 Mcps.

In order to synchronize multiple TDC modules, we also developed a prototype of a clock distributor and its receiver on a mezzanine card for HUL. Both the sampling clocks of four 260 MHz and the base clock of 130 MHz are generated from an external clock of 130 MHz, which is distributed from the clock distributor mezzanine card.

Software

The prototype DAQ software was developed with FairMQ[3, 4], which is written in C++ and provides a highly flexible and efficient message queueing framework for online and offline data processing. A base unit of FairMQ is called a *device*, which is a process with a state machine to run a user task, message passing channels, and control/monitor interfaces for the *device*.

Figure 2 presents typical topology for the algorithm study of online data processing. Data in one TDC module are read out by one *Sampler device* with a standard TCP socket. The data size of each readout cycle is fixed at a configured value. Therefore, the time length of each readout cycle depends on its data size (i.e. number of hits) and is different from cycle by cycle. Each *Sampler device* is followed by a *Sub-time frame builder (STFB) device*. The *STFB* seeks the heartbeat frames or spill-end frames in the data blocks, which are candidates for the separator to slice the data stream. The *STFB* divides the data to make data segments called sub-time frames (STFs). The lenght of STF is configured as a number of heartbeat frames. STF data are sent to *Time frame builder (TFB) devices*. *TFBs* are arranegd as multiple *devices* for load-balancing. A

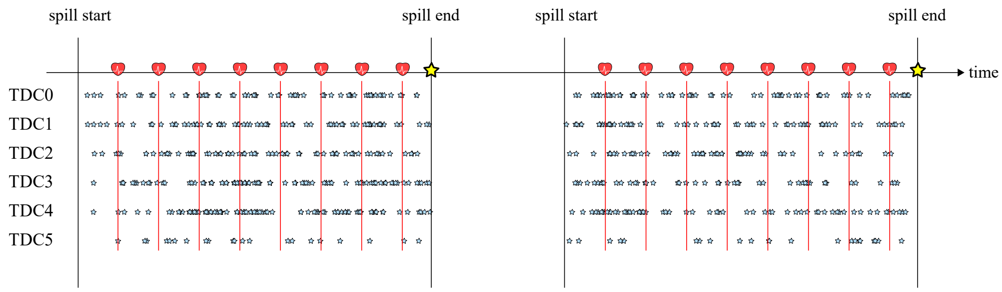

Figure 1: A schematic of time structure of streaming TDC. Hits (small light blue starts), heartbeat frames, and spill-end frames are shown.

simple static round-robin algorithm is used in the present system, where *STFB* selects the destination *TFB* in circular order without any priority and so that all STFs in the same time slice are gathered to the same *TFB*. *TFB* simply concatenates the received STFs, therefore, all hits have to be sorted in temporal order before event reconstruction. Hit sorting (or merging) and some analysis to reduce the data size is performed in the next *device* (ex. *Filter*). Finally, *FileSink* devices write data to disks.

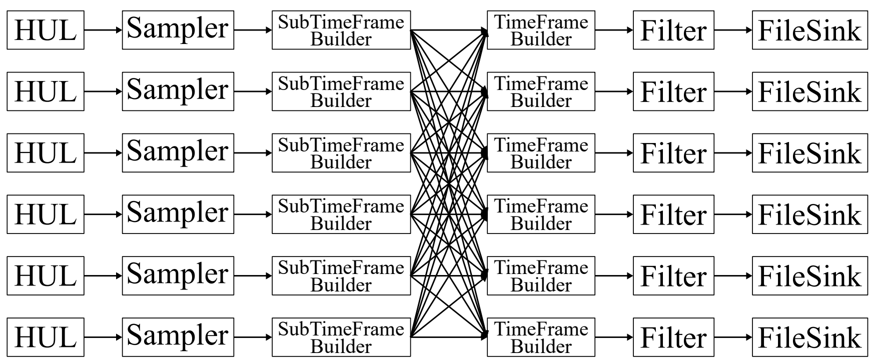

Figure 2: A topology example of DAQ processes in the prototype system.

In-beam test at ELPH

In December of 2018, we carried out a in-beam stress test for the streaming DAQ at the second experimental hall of ELPH. A photon beam of around 1 GeV from an electron synchrotron called the Booster STorage ring was converted into $e^{\pm}$ by the aluminum flange. The produced $e^{\pm}$ were used as a beam in the present test, where the average counting rate was controlled in the range from several kHz to more than 3 MHz by adujsting the intensity of circulating electrons in the ring, the speed of radiator insertion, and adding an aluminum plate on the flange. In total, 128 channels of scintillation fiber, 14 channels scintillation counter and 30 channels of drift chamber, which were about a hundred times smaller setup than that of the goal of E50, were read by the 6 TDC modules without any trigger. Six links of 1G Ethernet from TDCs were connected to a network switch (CISCO 6120xp or Fiberstore S3800-24F4S), while one 10G Ehternet port of the switch was connected to a DAQ PC with dual Intel Xeon E5-2630 v4 CPU (10 cores and 20 threads per CPU), and 256 GB RAM. All six streaming TDC modules worked at their highest throughput without obvious data loss. This is an important evidence for the transportation capablity of TCP/IP technique in near real time condition.

During the in-beam test, online processing algorithms to pack data from diffrent TDC modules together and filter out random hits were studied. It was found that the performance is strongly dependent on the data segment size: a smaller data segment to be processed the better performance can be expected. In any case, we have achieved an important goal by demonstrating the data processing capability of modern server machine and established the feasibility of streaming DAQ. Our next step is to construct a fast online track reconstruction package by fully exploring the possiblity of CPU and GPU.

References

[1] `http://www.j-parc.jp/researcher/Hadron/en/Proposal_e.html#1301`

[2] `http://openit.kek.jp/project/HUL/HUL`

[3] M. Al-Turany *et al.*, J. Phys. Conf.Ser. 513 (2014) 022001

[4] `https://github.com/FairRootGroup/FairMQ`