CBM First-level Event Selector Input Interface Demonstrator

Dirk Hutter, Jan de Cuveland and Volker Lindenstruth
Frankfurt Institute for Advanced Studies, Ruth-Moufang-Str. 1, 60438 Frankfurt, Germany
E-mail: hutter@compeng.uni-frankfurt.de

Abstract. CBM is a heavy-ion experiment at the future FAIR facility in Darmstadt, Germany. Featuring self-triggered front-end electronics and free-streaming read-out, event selection will exclusively be done by the First Level Event Selector (FLES). Designed as an HPC cluster with several hundred nodes its task is an online analysis and selection of the physics data at a total input data rate exceeding 1 TByte/s. To allow efficient event selection, the FLES performs timeslice building, which combines the data from all given input links to self-contained, potentially overlapping processing intervals and distributes them to compute nodes. Partitioning the input data streams into specialized containers allows performing this task very efficiently.

The FLES Input Interface defines the linkage between the FEE and the FLES data transport framework. A custom FPGA PCIe board, the FLES Interface Board (FLIB), is used to receive data via optical links and transfer them via DMA to the host’s memory. The current prototype of the FLIB features a Kintex-7 FPGA and provides up to eight 10 GBit/s optical links. A custom FPGA design has been developed for this board. DMA transfers and data structures are optimized for subsequent timeslice building. Index tables generated by the FPGA enable fast random access to the written data containers. In addition the DMA target buffers can directly serve as InfiniBand RDMA source buffers without copying the data. The usage of POSIX shared memory for these buffers allows data access from multiple processes. An accompanying HDL module has been developed to integrate the FLES link into the front-end FPGA designs. It implements the front-end logic interface as well as the link protocol.

Prototypes of all Input Interface components have been implemented and integrated into the FLES test framework. This allows the implementation and evaluation of the foreseen CBM read-out chain.

1. Introduction

1.1. The Compressed Baryonic Matter experiment

The Compressed Baryonic Matter (CBM) experiment is one of the major experiments at the future FAIR facility in Darmstadt, Germany. Its goal is to explore the QCD phase diagram at high baryon densities by measuring rare probes as well as bulk observables in fixed-target heavy-ion collisions [1]. CBM is designed as a multi-purpose detector composed of several sub-systems. Figure 1 shows the CBM detector in one possible configuration. The Silicon Tracking System (STS), located inside a large dipole magnet, is the main tracking device. For lower rate measurements it is accompanied by the Micro Vertex Detector (MVD) to allow high precision vertex reconstruction. Further downstream it is followed by a Ring Image Cherenkov detector (RICH) identifying electrons, a Transition Radiation Detector (TRD) for pion suppression and additional tracking, a Time of Flight wall (TOF) for particle identification, an Electromagnetic...
Figure 1. Rendering of the CBM detector setup. The beam enters from the left side and interacts with a fixed target inside the dipole magnet.

Calorimeter (ECAL) to measure photons and the Particle Spectator Detector (PSD) for event characterization. For muon measurements the RICH can be exchanged with the Muon Chamber detector (MUCH) which is shown in parking position.

In order to gain the needed precision for rare probe measurements the experiment is designed to run at event rates of up to 10 MHz. This places high demands on the detectors as well as the data acquisition systems. In addition, to filter weakly decaying particles, simple trigger signals don’t provide sufficient background suppression. A full online event reconstruction and topological analysis has to be performed to select these events [2]. Thus, a classical trigger architecture is not feasible for CBM.

1.2. CBM readout chain and online event selection
To fulfill the given requirements on event selection, CBM uses self-triggered front-end electronics (FEE) and a free-streaming readout system. Hits are registered autonomously by the FEE, time stamped and pushed to an online processor farm, the First-level Event Selector (FLES) for analysis. There is no central trigger signal. Event selection is exclusively done in the FLES.

The FLES is designed as a scaleable high-performance compute cluster. Equipped with heterogeneous many-core architectures such as GPUs and a high-throughput network, it performs the online event reconstruction and analysis [3]. With respect to the input interface the most important step in the FLES processing chain is timeslice building. In this process data from all given input links is combined to self-contained, potentially overlapping processing intervals and distributed to compute nodes in the cluster. This allows efficient event selection without the need for inter-node communication during analysis. As it is not foreseen to have link local processing on the FLES input nodes, timeslice building is performed directly on input data. Even though this process is similar to classical event building, there are two distinct differences. Due to the absence of a global trigger signal, there is no event definition prior to analysis. Thus, data has to be combined to time intervals according to its timestamp information. Second, timeslice building has to be preformed at full input data rate, as there is no previous filtering of data. Both render timeslice building a nontrivial task and affect the design of the input interface.
Figure 2. Schematic view of the CBM data flow. The FLES input interface connects the front-end read out with the FLES compute cluster.

2. FLES input interface

The FLES input interface is the linkage between the front-end readout electronics and the FLES data transport framework. The readout tree of the CBM experiment is shown in Figure 2. Streams from the detector front-ends are merged in multiple stages and sent via optical links to the FLES. Depending on the link speed available when producing the custom electronics, between 400 and 1000 links will be required. The FLES cluster will be installed in the central Green IT Cube computing center at GSI, which gives a routing distance of approximately 700 m between the Data Processing Boards (DPB) and the FLES input nodes. This enforces the use of single mode optics and link protocols that are not susceptible to latency.

To allow efficient FLES data handling without parsing single detector messages, input data streams are time-partitioned into specialized containers, named microslices. Each microslice covers the same, predefined amount of time for all sub-systems and is composed of a descriptor in a global format and a block of sub-system specific data. The descriptor provides information needed for data handling – especially the start time. The data block is specified to be self-contained, i.e., not depending on the stream history. This allows a data-agnostic implementation of the transport framework.

The central element of the input interface are custom PCIe FPGA boards, the FLES Interface Boards (FLIBs). Those terminate the optical links, perform pre-processing as needed and transfer the data to the hosts’ memories. An associated software stack publishes this data to the transport framework.

To show the feasibility of the FLES input interface and refine the design toward the final system, a demonstrator has been implemented. The key components are presented in the following sections.
3. FLIB prototype
Implementing the FLIB requires an FPGA board with a fast PCIe interface and connections for multiple high-speed optical transceivers. These features can be found in commercially available development boards. For the current FLIB prototype, the HTG-K7-PCIE development board is used. Based on a Xilinx Kintex7 K325T FPGA, its main features are a PCIe 8x gen2 interface, up to eight 10 GBit/s links via FMC and a DDR3 SO-DIMM slot. For the usage as FLIB it is equipped with FM-S14 or FM-S18 FMC modules providing four or eight SFP+ connectors for optical transceivers. Alternatively, FMCs with two QSFP connectors are available. A picture of the FLIB equipped with a FM-S14 is shown in Figure 3.

4. Host interface
Data is transferred to the host’s memory using a custom, full offload DMA engine adopted from the CRORC project [4] and fitted for CBM needs. For each input channel a dual ring buffer structure in the host’s memory as shown in Figure 4 is managed by the hardware. Data content of all received packages is continuously written to the data buffer. Once a package is finished, a descriptor entry is created and written to the descriptor buffer. It contains offset and size of the corresponding package in the data buffer and additional meta data form the package header. In case of empty packages only the descriptor entry is written. In this schema the descriptor buffer holds exactly one fixed size element for each package and can thus serve as index table to the data content. This allows full random access to all data and meta data without the need for any stream processing. The stored offset doesn’t wrap at the buffer boundaries. As a result the index table can be reused when the buffer content is copied to a different location by storing a single address offset.

Synchronization between the software consumers and the DMA engine is achieved by exchanging a set of ring buffer write and read pointers. Whenever a descriptor entry DMA write is passed to the PCIe interface, an internal hardware register holding the descriptor write offset is updated. The consumer can fetch this offset via a PCIe read request. Strict PCIe transaction ordering guarantees that all DMA transactions have reached the main memory before the offset is delivered to the consumer. To reduce complexity the offset to the data buffer is obtained from the index table. To free a buffer segment, the consumer writes a read offset update to a hardware register.

The chosen buffer structure allows very efficient timeslice building. Instead of dealing with individual, comparatively small packages, the timeslice building can rely on copying continuous chunks of data to the appropriate target node. This significantly reduces the transaction rate.

Figure 5 shows the measured DMA throughput for one to four 1 GB/s input channels over the
maximum size of generated packets. The test was performed on a system with an Intel Xeon E3-1200 CPU and a PCIe maximum payload size set to 128 bytes. For each configuration payload only and total DMA throughput including descriptors are plotted. The red line indicates the theoretical PCIe maximum throughput for the test system. The measured throughput saturates as expected for all configurations. For one to three channels the PCIe interface provides more bandwidth than needed allowing to shadow the descriptor overhead for very small packages and reaching the plateau earlier. For four channels the PCIe interface is saturated, shifting the curve to the right. The maximum throughput archived is 3346 MB/s. Deeper investigations have shown that the difference to the theoretical maximum is introduced by host system limitations and not the FLIB hardware design.

5. Front-end link
The connection between the FLIB and the front-end readout logic is implemented via a custom optical link. Instead of a dedicated front-end interface card commonly used in previous experiments, the FLES interface module (FLIM) HDL module integrates this functionality directly into to the DPB’s FPGA logic. It implements the front-end logic interface and the link protocol, and it encapsulates FPGA resources needed to implement the link. This approach allows better usage of the given FPGA resources while still providing convenient integration for sub-system designers. A block diagram of the FLES link composed of a FLIM and the accompanying receiver side in the FLIB is depicted in Figure 6. As transport layer the 64B/66B aurora protocol at a line rate of 10.3125 Gbit/s is used. It is foreseen to upgrade the line rate to 25.78125 Gbit/s for the next FPGA generation. The link protocol implements a packet-based data channel with flow control and a bidirectional control channel on top of the transport layer. Each transmitted package corresponds to one microslice. The protocol engine adds meta data and a CRC-32 Castagnoli end-to-end check sum, which is kept throughout the whole FLES data handling chain, allowing data consistency checks at every stage. The control channel enables run time configuration and diagnostics independent from the sub-system’s slow control interface. An optional internal pattern generator is available for in-system link verification.

Figure 4. Dual ring buffer structure used for DMA transfers. The data buffer receives the data content while the desc buffer hold an index table and meta data.
Figure 5. FLIB PCIe DMA throughput for one to four 1 GB/s input channels. The red line indicates the theoretical PCIe maximum for the tested system.

Figure 6. Schematic overview of the FLIM and accompanying receiver implementation.

6. Integration into the FLES framework

For integrating the input interface into the FLES framework a software stack was developed. It can be divided into a low-level device driver, a device library and a publisher application. The PDA [5] driver is used as low-level device driver. It provides generic communication to PCIe devices and registration of memory for DMA transfers. On top of the device driver an object-oriented C++ device library, the libflib, was developed. It provides all features needed to operate the FLIB hardware. Device driver and library are designed to take user-provided buffers and configure these as DMA target buffers for data transfer. This allows flexible integration into other software components.

The library provides all elements to integrate the input interface directly into the transport framework. This approach was taken in a first demonstrator version. Nevertheless it is beneficial to create a dedicated interface between timeslice building and input interface. On the other hand, especially in a high-throughput applications like the FLES framework, it is beneficial not
to introduce unnecessary overhead. A new approach satisfies both criteria. All communication with the hardware is managed by a publisher process. The interface to a consumer, e.g., the timeslice building process, is implemented using POSIX shared memory. Data is accessed by the consumer via a generic dual ring buffer interface. The design of the device driver and library enables the usage of shared memory segments as DMA buffers. Thus, the hardware can directly write to a dual ring buffer structure inside this shared memory without introducing additional overhead. The publisher process manages read pointer updates towards the DMA engine and write pointer updates towards the consumers via the same shared memory.

This approach has several advantages. The consumer is independent of the hardware and device library, which provides more flexibility during development. The shared memory can be filled from any data source, not only a PCI device, e.g., a software pattern-generator or recorded files for debugging and verification. Furthermore, one can insert extra processing steps like filters into the chain. This can be used for testing software mock-ups of anticipated functionality such as online hardware feature extraction of the detector data in the demonstrator system.

7. Status and summary
The FLES input interface receives free-streaming data from the CBM detector and feeds it into the FLES compute cluster. An FPGA-based PCIe board, the FLIB, is used to bridge the connection between custom electronics and commercial off-the-shelf PCs. As a first prototype a commercial development board is used. A custom firmware implements the optical link protocol, data handling and DMA engine. The measured DMA throughput reaches 94% of the theoretical maximum, limited only by the host system. For providing data access at application level, a publisher process handles all needed operations. Data between processes is exchanged using a POSIX shared memory. The software stack allows filling this shared memory directly from the FLIB card without additional copies.

Prototypes of all Input Interface components have been implemented and integrated into the FLES data transport framework to serve as a demonstrator. This allows the evaluation of the input interface in context of the real-life CBM readout chain. Several CBM sub-systems have integrated the FLES input interface into their prototype read-out setups and use the FLES framework to record test data. These read-outs are successfully used in lab setups and in detector beam tests. Results show very good performance of the FLES input interface demonstrator at high sustained input data rates and verify the feasibility of the concept.

References
[1] Ablyazimov T et al (CBM Collaboration) 2017 Eur. Phys. J. A 53 60
[2] Friman B et al 2011 The CBM Physics Book (Berlin, Heidelberg: Springer-Verlag) pp 631-656
[3] de Cuveland J and Lindenstruth V 2011 J. Phys.: Conf. Ser. 331 022006
[4] Engel H, Alt T, Breitner T, Gomez Ramirez A, Kollegger T, Krzewicki M, Lehrbach J, Rohr D and Kebschull U 2016 JINST 11 C01041
[5] Eschweiler D 2015 Efficient Device Drivers for Supercomputers (Frankfurt: Goethe University Frankfurt)