A high-throughput input interface for the CBM FLES

Dirk Hutter[1,2,*], *Jan* de Cuveland[1,2], and *Volker* Lindenstruth[1,2,3]

[1]Johann Wolfgang Goethe-Universität Frankfurt, Frankfurt am Main, Germany
[2]Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany
[3]GSI Helmholtzzentrum für Schwerionenforschung, Darmstadt, Germany

Abstract. The CBM First-level Event Selector (FLES) serves as the central data processing and event selection system for the upcoming CBM experiment at FAIR. Designed as a scalable high-performance computing cluster, it facilitates online analysis of unfiltered physics data at rates surpassing 1 TB/s.
As the input to the FLES, the CBM detector subsystems deliver free-streaming, self-triggered data to the common readout interface (CRI), which is a custom FPGA PCIe board installed in the FLES entry nodes. A subsystem-specific part of the FPGA design time-partitions the input streams into context-free packages. The FLES interface module (FLIM), a component of the FPGA design, acts as the interface between the subsystem-specific readout logic and the generic FLES data distribution. It transfers the packed detector data to the host's memory using a low-latency, high-throughput PCIe DMA engine. This custom design enables a shared-memory-based, true zero-copy data flow.
A fully implemented FLIM for the CRI board is currently in use within CBM test setups and the FAIR Phase-0 experiment mCBM. We present an overview of the FLES input interface architecture and provide performance evaluations under synthetic as well as real-world conditions.

1 Introduction

The Compressed Baryonic Matter (CBM) experiment, currently under construction at the Facility for Antiproton and Ion Research (FAIR) in Darmstadt, Germany, is a fixed-target, heavy-ion experiment dedicated to exploring the QCD phase diagram in the region of high baryon density [1]. A distinctive feature of the CBM experiment is that many of its probes lack simple trigger signatures, necessitating a more sophisticated approach to event selection and data acquisition. Furthermore, the rare production rate of some probes requires measurements at up to 10 MHz interaction rate to collect sufficient statistics.

CBM addresses these challenges through an innovative approach. Unlike traditional, multi-level triggered systems, CBM employs a self-triggered, free-streaming readout system where event selection is performed exclusively in a high-performance computing cluster called the First-level Event Selector (FLES). This approach requires processing the full unfiltered data stream, resulting in a design data rate exceeding 1 TB/s. Implementing such a system presents unique challenges in data acquisition and processing that demand novel solutions in both hardware and software design.

*e-mail: hutter@compeng.uni-frankfurt.de

1.1 CBM readout and event selection architecture

The CBM readout and event selection architecture can be divided into several tiers. At the detector front-ends, self-triggered readout electronics capture zero-suppressed hit data. Each data point is time-stamped and streamed without additional pre-filtering to the FLES cluster. FPGA-based PCIe extension cards in the FLES entry nodes—the common readout interface (CRI) cards—terminate the custom optical front-end links and mark the transition to commercial off-the-shelf computing infrastructure. Data from over 5000 front-end links enter the cluster distributed across multiple CRIs and entry nodes. To facilitate event reconstruction without inter-node communication, the FLES performs timeslice building [2]. In this process, matching time spans from all input channels are combined into processing intervals called timeslices and shipped to compute nodes for analysis.

1.2 Previous work

The CBM readout architecture has evolved significantly since its initial conception. Earlier versions included an additional FPGA processing layer preceding the PCIe cards in the FLES input nodes. As part of an architectural rework, the functionality of both FPGA layers was consolidated into the CRI card, streamlining the CBM data acquisition chain. Building on this prior architecture, the presented work introduces significant modifications and optimizations to accommodate updated requirements. While many foundational principles remain unchanged, key refinements enhance efficiency and performance. This paper presents a comprehensive description of the current implementation, incorporating these modifications and improvements.

2 FLES input interface

The FLES input interface marks the transition point from subsystem-specific readout components to the central data acquisition system (DAQ). This section discusses three key aspects: the microslice data model, which defines the underlying data format of the DAQ chain. The FLES interface module, which implements the interface to the subsystem-specific readout in FPGA hardware and manages host data transfers. And a zero-copy software interface, which connects the input interface to the timeslice building process.

2.1 Microslice data model

The FLES input interface must handle stateful streams of heterogeneous data in subsystem-specific formats, with the additional complexity that time intervals do not correspond directly to data size due to zero-suppression techniques. To address these challenges, the microslice data model was developed [3]. Data is partitioned into short, context-free time intervals called microslices. A microslice consists of a descriptor holding all metadata needed for data handling (e.g., a reference timestamp) in a common format, and a block of detector data from its specific time interval in a subsystem-specific format. Microslices are the smallest data unit in the DAQ chain. The microslice data content is context-free, i.e., it does not depend on the data of other microslices. This design choice allows for efficient processing and data handling, as microslices can be processed independently without requiring knowledge of the surrounding data. The time-based encapsulation enables the timeslice building to efficiently segment the data streams regardless of the original subsystem format or varying data densities.

2.2 FLES interface module

The central element of the input interface is the FLES interface module (FLIM), a hardware design module for the CRI's FPGA design. Figure 1 shows a schematic overview of the

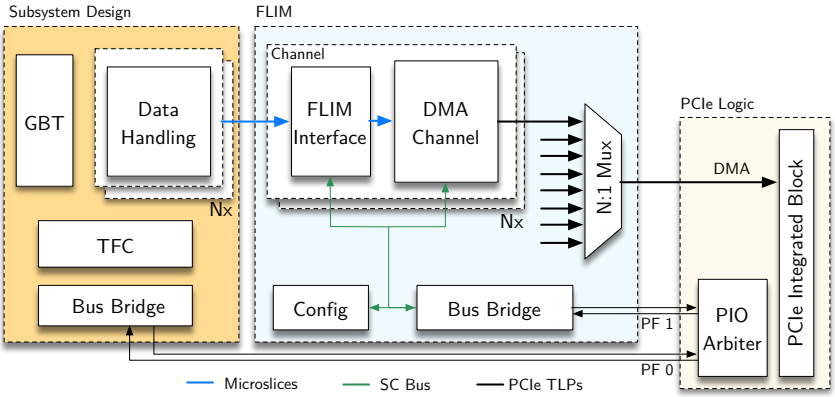


Figure 1. Overview of the CRI FPGA design. The FLES Interface Module (FLIM) in the middle receives microslices from a subsystem-specific block and manages DMA transfers to the host's main memory.

FLIM design and its integration into the overall CRI hardware design. The CRI design can be logically divided into two major parts. A subsystem-specific block handles communication with the front-end electronics and all subsystem-dependent data formatting tasks, including microslice building. The second major block is the FLES interface module (FLIM), which is common to all CRI designs. The FLIM receives microslices from the subsystem part and manages data transmission to the host's main memory. It provides multiple independent input channels, each equipped with its own DMA engine and dedicated host memory buffers. This multi-channel design reduces requirements for data concentration in hardware while offloading portions of this task to software. The targeted aggregation ratio between input links and FLIM channels is at least 6:1. The FLIM channels dynamically share the PCIe interface, optimizing bandwidth utilization while maintaining flexibility for subsystem designs. A shared slow control bus and PCIe bus bridge facilitate configuration, control and monitoring of the interface and DMA engines.

A full-featured FLIM implementation is available for the current CRI prototype. The prototype utilizes the BNL-712 board [4], which features a Xilinx Kintex UltraScale XCKU115 FPGA. The board provides dual PCIe x8 Generation-3 interfaces, managed through a PCIe switch towards the host, and supports 48 optical inputs for detector connectivity. To utilize both PCIe interfaces, the main data path is instantiated twice in the FPGA design. From the software perspective, a single CRI card is handled as two independent PCIe devices.

2.2.1 DMA engine and buffer management

The DMA engine employs a dual ring buffer memory scheme enabling low-overhead, highly efficient data handling. The scheme consists of two primary buffers per channel: a data buffer that maintains continuous microslice data content, and a descriptor buffer containing fixed-sized microslice descriptors (see Figure 2). The descriptor buffer serves as an index table

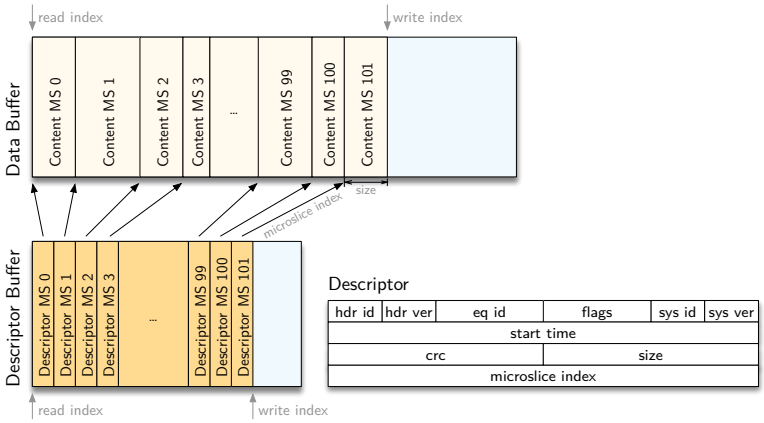


Figure 2. Dual ring buffer memory scheme used by the DMA engine. The data buffer holds the variable-size microslice (MS) data content. The descriptor buffer holds fixed-size descriptors (depicted on the right) and serves as an index table to the data buffer.

to the microslice content, enabling efficient data access for software. Additionally, it holds all available metadata for the microslices, allowing low-overhead, cache-efficient metadata scanning.

This design provides several significant advantages in data handling. The system enables efficient block-wise data access while minimizing memory fragmentation. At the same time, the buffer management is simple enough to be fully implemented in hardware without requiring extensive FPGA resources. This full-offload design reduces dependency on CPU resources. The hardware implementation supports scatter/gather DMA operations, eliminating the need for continuous buffer mappings. Synchronization overhead between hardware and software is minimized through a simple but effective read and write index exchange mechanism. Rather than using direct memory addresses, the system employs monotonically increasing buffer indices that continually increment regardless of buffer wraparound. This approach enables downstream processing stages to reuse the descriptor index table by simply applying an offset adjustment, independently of the buffer structure, while maintaining efficient translation from indices to actual ring buffer memory addresses.

2.2.2 Integration into the CRI design

The FLIM and the subsystem logic share the same physical PCIe interface between FPGA and host, but have vastly different communication requirements (see Figure 1). The subsystem logic only utilizes programmed I/O (PIO) data transfers, while the FLIM implements a high-throughput DMA data path with PIO for controls. The PIO interface of the FPGA's integrated PCIe core is shared between the two modules via a simple arbiter that routes request transaction layer packets (TLPs) to the appropriate destination. The bus mastering TLP interface is exclusively used by the FLIM.

To minimize software dependencies between the FLIM and the subsystem logic, they use two separate physical functions (PFs) of the PCIe interface. Each PCIe physical function is enumerated by the host's operating system and can be bound to an individual device driver. This design choice fully segregates the software stacks of FLES and subsystem controls, providing flexibility in the choice and evolution of device drivers. Deploying separate

device drivers also reduces interference between subsystem controls and time-critical DMA operations.

2.3 Zero-copy software interface

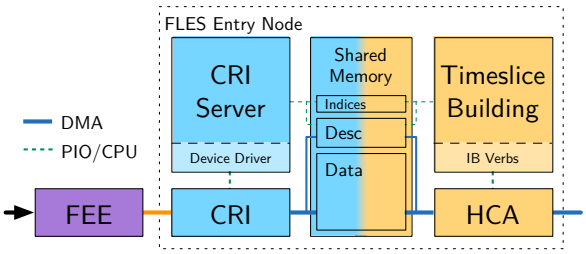


Figure 3. Zero-copy data flow between the FLES input interface and timeslice building. Both steps share the same DMA buffer in shared memory. Only synchronization information is exchanged via CPU reads and writes. Actual data movement is performed via DMA.

The software stack has been carefully engineered to achieve maximum efficiency with minimal overhead. At its core, the system implements a zero-copy interface to timeslice building, eliminating unnecessary data copying operations that could impact performance.

A key feature of the software integration is the ability to perform DMA operations directly to a POSIX shared memory in user space. This capability allows the system to handle multi-gigabyte buffers without the traditional limitations associated with kernel memory management. Furthermore, these DMA buffers can be shared with other PCIe devices, such as InfiniBand host channel adapters (HCA), facilitating efficient data movement throughout the system. The resulting zero-copy data flow is depicted in Figure 3. A data publishing agent, labeled CRI server, serves as a central coordinator for all hardware communication. This server establishes a synchronization interface in the same shared memory used by the card as DMA buffer, transferring read and write indices to or from hardware on consumer request. The consumer application (i.e., timeslice building) can efficiently create a gather list by accessing only the indices and entries in the descriptor buffer. Actual data movement can rely on DMA transfers by the HCA. The design emphasizes high-throughput data transfers while ensuring minimal overhead and memory bandwidth usage.

3 Performance evaluation

The system design's performance and feasibility have been evaluated through several synthetic benchmarks and in real-world production environments of physics experiments.

3.1 Full system throughput benchmark

The full system throughput—from the FLIM's subsystem interface via the PCIe interface to software consumers—has been evaluated in extensive measurements. For the measurement, pattern generators within the FPGA generate time-partitioned streams of microslices which are fed to the FLIM channels. In the tested setup, each pattern generator provides a maximum throughput of 1.28 GB/s. To saturate the PCIe interface, eight pattern generators and FLIM channels were used in parallel. The consumer discarded the data without any processing.

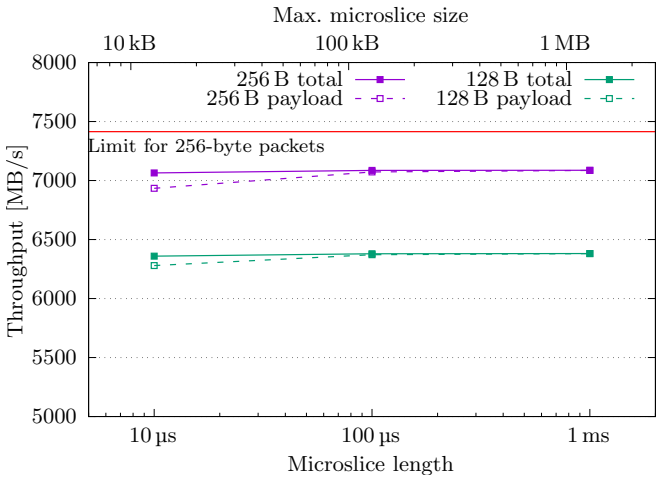


Figure 4. Full system data throughput for different microslice lengths and DMA packet sizes.

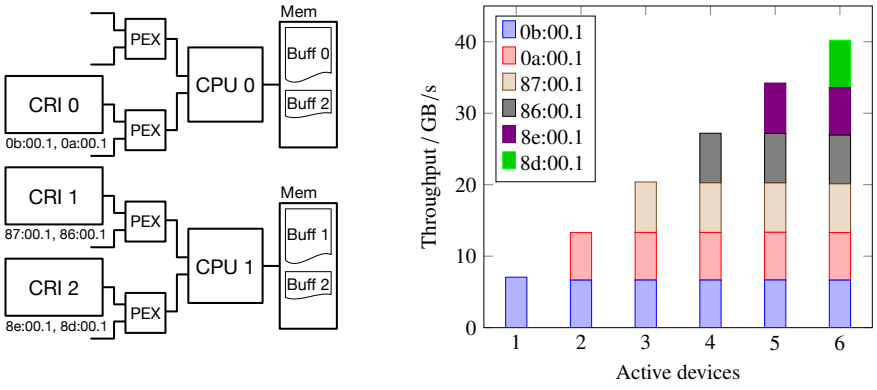
Figure 4 shows the measured system throughput versus microslice length for two different DMA packet sizes. The dotted lines show the throughput of microslice data payload without any protocol overhead. The solid lines show the total data throughput over the PCIe interface (excluding PCIe TLP headers). The red line shows the calculated theoretical throughput limit for 256-byte DMA packets. The time-based partitioning of the pattern generator implies that the data size of a microslice depends on the actual data throughput. As an orientation, the upper x-axis shows the maximum possible data size of a microslice. For targeted microslice sizes above 100 µs, the throughput for 256-byte packets reaches 95 % of the theoretical limit, representing near-optimal utilization of the available bandwidth. With a higher packet size of 512 B, this can be further improved to 98 %. Further investigations have shown that the limitation is a lack of PCIe credits passed from the host system, which cannot be influenced by the user. For very small microslices, the overhead from transmitting descriptors becomes non-negligible, and the payload throughput drops slightly below the total throughput. However, the system maintains high overall efficiency even with small microslice sizes, demonstrating the robustness of the design across varying operational conditions. Bandwidth distribution among the 8 DMA channels is fair in all cases. The observed CPU load was negligible.

3.2 Multi-device scaling

The FLES entry stage architecture has been designed to accommodate multiple CRI cards per entry node, with a targeted configuration of at least three CRI cards and one InfiniBand HCA for network connectivity per node. Therefore, the performance implications of operating multiple CRI cards simultaneously within a single node have been evaluated.

Performance evaluations were conducted using a test system equipped with dual Intel Xeon E5-2650 v4 CPUs and 128 GB of 2400 MHz DDR4 main memory. The test configuration is depicted in Figure 5a. The system supports up to eight x16 Generation-3 PCIe extension cards, with pairs of slots sharing PCIe switches (PEX) on the mainboard. Three CRI cards were installed, each connected to a separate PCIe switch, with one CRI connected to NUMA node 0 and two to NUMA node 1. Due to the dual PCIe interface, each card registers two independent PCIe endpoint devices with the system. To emulate realistic conditions, DMA buffers were distributed equally across both NUMA domains. The first CRI

in each domain exclusively utilized memory on its local NUMA node, while the second card in domain 1 employed buffers distributed across both NUMA nodes. Testing was conducted with a microslice size of 100 µs, a DMA packet size of 256 B, and 6 microslice streams per endpoint, representing worst-case operational parameters.



(a) (b)

Figure 5. Multi-device scaling evaluation. Figure (a) shows the server setup and buffer distribution. Figure (b) shows measured throughput versus the number of active devices.

The results are shown in Figure 5b. Measurements revealed excellent scaling characteristics across multiple devices. Individual endpoints achieve throughput rates of up to 7.06 GB/s, representing 95 % of the theoretical maximum at the given packet size. When operating both endpoints of a single CRI card simultaneously, throughput per endpoint shows a slight decrease to approximately 6.65 GB/s. With an increasing number of endpoints, the system demonstrated near-perfect scaling, achieving a total throughput of 40.2 GB/s. The bandwidth distribution across devices shows remarkable fairness, ensuring that all data streams receive appropriate resources. This balanced performance is crucial for maintaining consistent data processing across all detector subsystems and avoiding potential bottlenecks in the DAQ chain.

The results confirm the feasibility of operating multiple CRI cards within a single host NUMA environment without significant performance degradation.

3.3 Real-world deployments

The FLES input interface, together with the FLES software stack for data distribution, has been successfully deployed and validated in numerous detector test setups as well as in the FAIR Phase-0 experiment mCBM [5]. Using the system in production setups provides concrete evidence of its capabilities in real-world conditions.

The most demanding FLES setup is part of the mCBM experiment. The current deployment operates 12 CRIs distributed across six FLES entry nodes, serving seven distinct detector systems with a total of 58 simultaneous FLIM channels. This demonstrates the system's ability to handle complex, multi-detector configurations. Under operational conditions with beam, the system has achieved total peak data rates exceeding 5 GB/s without presenting bottlenecks in the readout. This real-world verification provides confidence in the system's ability to meet the future demands of the full CBM experiment. Operation within the production environment of mCBM requires sophisticated control and monitoring solutions. The

FLES system provides numerous metrics to determine operating conditions and system health of both the serviced subsystems and the FLES data path itself. The FLIM hardware design includes performance counters at key points of the data path to collect monitoring metrics, such as microslice input frequency and data rate per FLIM channel, or data throughput via the PCIe interface. A monitoring service on the entry nodes periodically collects measurements from all CRIs and pushes them to an InfluxDB time-series database. Additionally, application-level monitoring collects metrics like buffer utilization and network data rates, which are collected in the same database. Visualization of the monitoring data is based on the Grafana web application. This central monitoring infrastructure has proven essential for production operation, giving users a complete picture of operating conditions. In turn, experience from this real-world environment has been used to continuously improve the system toward the final CBM setup.

4 Conclusion

The FLES input interface serves as the bridge between subsystem-specific readout electronics and CBM's common data acquisition. The microslice data model has proven to be an effective solution for managing heterogeneous, time-based data streams, providing both efficiency and flexibility in data handling.

The crucial element of the input interface is the FLIM FPGA design with a full-offload DMA engine enabling a zero-copy data path. The implementation achieves near-theoretical maximum PCIe bandwidth while maintaining low CPU overhead and demonstrating excellent scaling properties across multiple devices. The implementation of separate PCIe physical functions for FLIM and subsystem design provides a robust foundation for future developments. The successful integration of multiple CRI cards within single entry nodes demonstrates the architecture's scalability. The system's ability to maintain high performance across multiple devices and NUMA domains confirms its suitability for the demanding requirements of the CBM experiment.

The system's reliability has been thoroughly validated through continuous operation in various physics and development setups, particularly in the mCBM experiment. As the CBM experiment progresses toward full operation, the current implementation provides a solid foundation for future developments. Ongoing work focuses on supporting higher data rates and upcoming hardware platforms.

This work is supported by BMBF (05P21RFFC1).

References

[1] T. Ablyazimov et al., Challenges in QCD matter physics –The scientific programme of the Compressed Baryonic Matter experiment at FAIR. Eur. Phys. J. A **53**, 60 (2017). doi:10.1140/epja/i2017-12248-y

[2] J. de Cuveland et al., *Technical Design Report for the CBM Online Systems – Part I, DAQ and FLES Entry Stage* (GSI, Darmstadt, 2023). doi:10.15120/GSI-2023-00739

[3] D. Hutter, *An Input Interface for the CBM First-level Event Selector* (Goethe University, Frankfurt, 2020) pp 35-39. urn:nbn:de:hebis:30:3-591599

[4] Kai Chen et al., A Generic High Bandwidth Data Acquisition Card for Physics Experiments. IEEE Trans. Instrum. Meas. **69.7** 4569-4577 (2020). doi:10.1109/TIM.2019.2947972

[5] CBM Collaboration, *mCBM@SIS18 - A CBM full system test-setup for high-rate nucleus-nucleus collisions at GSI / FAIR* (GSI, Darmstadt, 2017). doi:10.15120/GSI-2019-00977