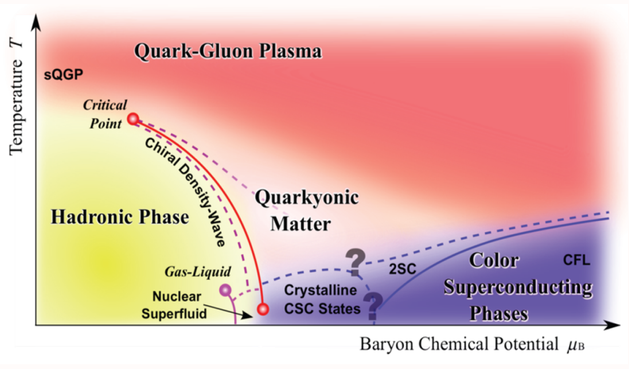
Developing a Cluster-Finding Algorithm with Vitis HLS for the CBM-TRD

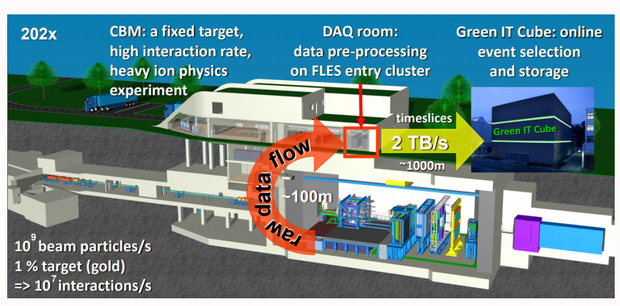
David Schledt, Christoph Blume and Udo Kebschull
Goethe-Universität Frankfurt am Main

The Compressed Baryonic Matter Experiment



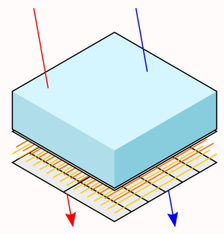
The CBM experiment aims to explore the QCD phase diagram at high net-baryon densities and moderate temperatures, with a focus on rare particles as probes.

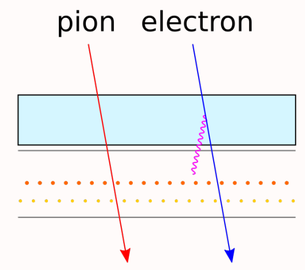
These will be produced with unprecedented statistics. With an interaction rate of up to 10 MHz, raw data rates of up to **2 TB/s** are generated. These data must undergo **4D online event selection** to limit the recorded data volume.



As this is a computationally challenging task, the data should be preprocessed in the FPGA read-out layer as much as possible.

The **CBM-TRD** consists of 4 layers of read-out chambers. Each layer will be built of different-sized modules with the read-out channel density increasing towards the center. Each layer has 82,432 read-out channels, for a total of 329,728 read-out channels for the entire TRD station.





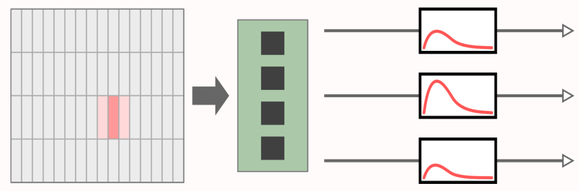
The TRD is **one of the biggest data producers** in the experiment. Therefore, it is an obvious candidate for data preprocessing in the read-out FPGA.

TRD Readout ASIC

The **SPADIC** is the read-out ASIC for the CBM-TRD. It provides an oscilloscope-like sampling of the channel input charge with an integrated trigger logic and a sampling rate of 16 MHz. When a detector hit occurs, the signal is encoded into the message stream with **up to 32 ADC samples** per message.

SPADIC
Self triggered Pulse Amplification and Digitization asIC

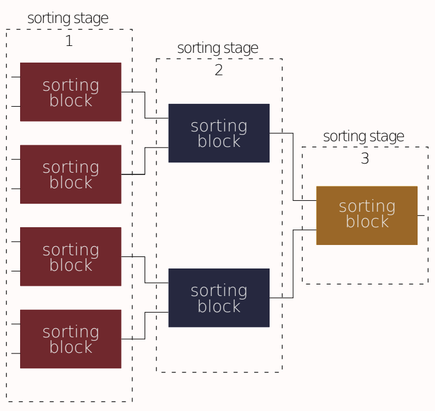
Each hit message is **marked with a timestamp, a channel ID and a trigger type**. There are 32 read-out channels per ASIC, which are internally split into two 16 channel groups. Within one group,



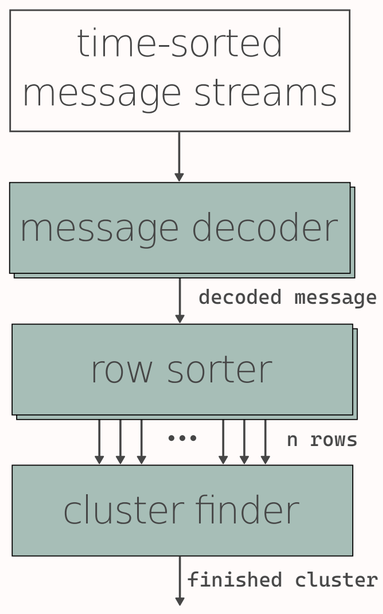
Schematic depiction of a detector trigger, where the center pad fulfilled the SPADIC trigger logic and the neighboring pads are read out as forced neighbor triggers.

the data is serialized according to the timestamp. Additionally, the SPADIC features a **forced neighbor trigger logic**, allowing to read out adjacent pads without lowering the thresholds. Channels that fulfill the trigger logic are marked as type 1 and neighbor triggers are marked as type 2.

Data Preparation

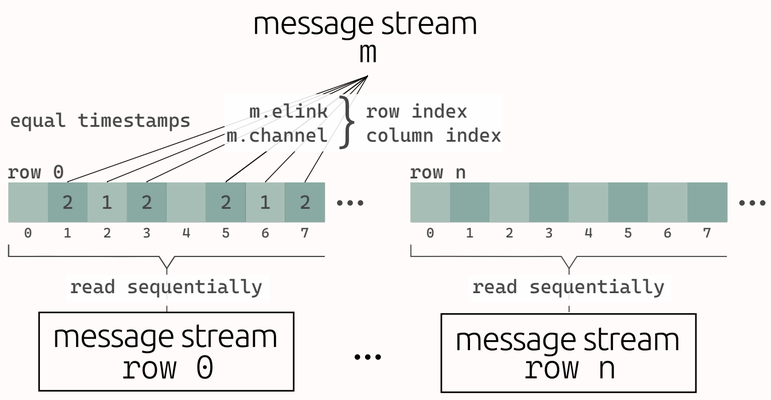


Inside the **FPGA**, the data needs to be prepared for further processing. The entire processing chain presented here was written with **Xilinx Vitis HLS**, which enables the generation of hardware with C++. This allows for much faster design iterations as one has access to the **higher-level abstractions of C++ and functional validation in software**.
The frontend links are connected to the FPGA layer via the GBTX-ASIC from CERN. The raw message streams from three GBT-links are serialized into a single time-sorted message stream in the FPGA. In the next step, raw

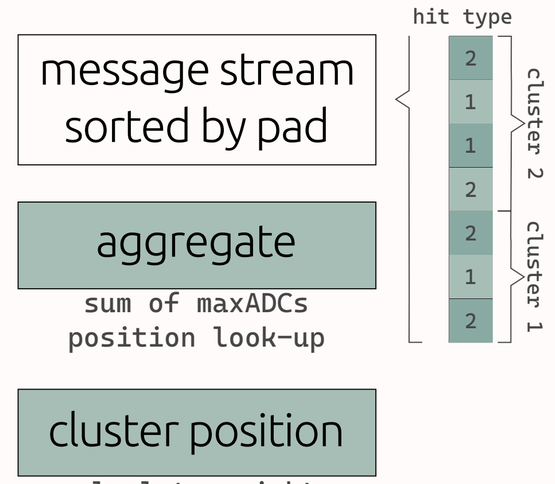


message frames are decoded in the *message decoder*, where the message frames are combined into a single trigger message. In the decoding step, the data undergoes a **baseline correction, the maximum ADC value is extracted, and a more precise signal time is reconstructed**. Additionally, pile-up is detected and removed from the message stream for a potential pile-up reconstruction.

Cluster Finder

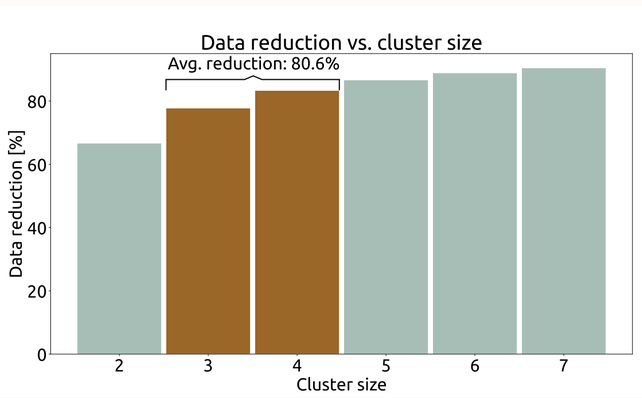


The next step is to **sort the data stream according to the pad configuration**, as this will significantly simplify the **cluster finding algorithm** both in complexity and speed.
With the cluster finding module's input sorted, the actual aggregation of trigger messages into a cluster is straightforward. As the SPADIC marks the trigger type of the messages, a cluster is (almost) always surrounded by two forced neighbor triggers. During the *aggregate* step, the maximum ADC values of the cluster messages are summed up, and the pad positions are extracted from a look-up table. After the aggregation is finished, the **cluster position** can be calculated. This is done via a **center of gravity** calculation in the current implementation. Still, it is planned to take the pad response function into account to further increase the spatial resolution of the extraction.



Results

All implemented modules could be **fully pipelined** and run at a frequency of 200 MHz, although a higher frequency is possible with the addition of more pipeline stages.
However, the usage of HLS always comes with a penalty in resource consumption, but the **increased development speed** makes it worthwhile.
With an average cluster size of 3.5 and four message frames per trigger, the data size is reduced from 42 bytes per trigger to just 8 bytes; **reducing the data load of the TRD by 80%**.



GOETHE UNIVERSITÄT FRANKFURT AM MAIN