Software design of the ATLAS Muon Cathode Strip Chamber ROD

R. Murillo, M. Huffer, R. Claus, R. Herbst, A. Lankford, M. Schernau, J. Panetta, L. Sapozhnikov, I. Eschrich and J. Deng.
raul.murillo.garcia@cern.ch

Abstract. The ATLAS Cathode Strip Chamber system consists of two end-caps with 16 chambers each. The CSC Readout Drivers (RODs) are purpose-built boards encapsulating 13 DSPs and around 40 FPGAs. The principal responsibility of each ROD is for the extraction of data from two chambers at a maximum trigger rate of 75 KHz. In addition, each ROD is in charge of the setup, control and monitoring of the on-detector electronics. This paper introduces the design of the CSC ROD software. The main features of this design include an event flow schema that decentralizes the different dataflow streams, which can thus operate asynchronously at its own natural rate; an event building mechanism that associates data transferred by the asynchronous streams belonging to the same event; and a sparcification algorithm that discards uninteresting events and thus reduces the data occupancy volume. The time constraints imposed by the trigger rate have made paramount the use of optimization techniques such as the curiously recurrent template pattern and the programming of critical code in assembly language. The behaviour of the CSC RODs has been characterized in order to validate its performance.

1. System architecture

The ATLAS Cathode Strip Chamber (CSC) system is designed to measure high momentum muons in the forward regions. The CSC system consists of two end-caps, each containing 16 chambers. Each CSC chamber has four layers, providing in total 768 precision coordinate channels in the direction of magnetic curvature and 192 transverse coordinate channels in the less critical non-bending direction.

Due to severe radiation levels in the CSC environment, a minimum of CSC electronics, known as the Front-End-Electronics (FEE), are located on the detector. The FEE are partitioned into units of Amplifier-Shaper Module (ASM-II) boards. One board is designed to handle 192 channels. Thus, each chamber employs five ASM-II boards, four for the precision layers and one for the transverse layers. Each channel is connected to a Preamplifier and Shaper (P/S), which makes a bipolar pulse with 140 ns shaping time. The shaped pulses are sampled every 50 ns and stored in a Switched Capacitor Array. The analogue signal is digitized to a 12-bit value. The data from the ASM-II (192 channels) for a single time sample is called a time-slice [1, 2].

The off-detector electronics consist of optically linked CSC Transition Modules (CTM) and Readout Drivers (RODs). Each ROD has a corresponding CTM, connected together through a custom-built backplane. The CTM provides three major functions: the logic to monitor, control and receive data from the FEE of its corresponding chambers; the logic and buffering to respond appropriately to trigger requests; and a single fiber-optic transmitter, referred to as the Read-Out Link (ROL), used to send event data to the ATLAS Trigger and Data Acquisition (TDAQ) system. The responsibilities of each ROD are twofold: setting up, controlling and monitoring the on-detector electronics and the CTM; and extracting data from the chambers and sending the resulting event to the ROL [3].
2. CSC ROD architecture
The CSC ROD is a purpose-built 9U VME board encapsulating thirteen 300 MHz Texas Instruments TMS320C6203 DSPs and around 40 Xilinx Spartan II FPGAs (Figure 1). In the CSC jargon, the DSPs are referred to as Generic Processing Units (GPU), which are physical plug-in daughter boards (Figure 2) containing the DSP and the so called XB and EMIF FPGAs. Twelve of these GPUs are known as Data Processing Units (DPU), which are grouped in ten Sparcification Processing Units (SPU) and two Rejection Processing Units (RPU). The thirteenth GPU is the Host Processing Unit (HPU), responsible to initialize, configure and monitor the ROD as well as to provide a communication interface with the single board computer used as the Readout Crate Controller (RCC).

The DSP-based architecture was born of the belief that software is a more manageable vehicle for solving complex problems than hardware. The DSP obviated the need for costly FPGAs for data processing, large FIFOs or SRAM chips for data buffering, and other hardware for histogramming and monitoring of detector performance [4].

Each ROD is designed to service the data from two chambers. The board has thus two identical halves known as side A and side B. Each side includes five SPUs and one RPU, dedicated to perform the feature extraction on the raw data from one chamber [4]. In total, there are sixteen boards split over two ATLAS purpose-built crates. Each crate, in addition to housing its eight ROD-CTM pairs, contains a Timing Interface Module (TIM), a Local Trigger Processor (LTP) and a RCC. The RCC functions as the crate’s VME bus master and executes ATLAS specified run control software, used to orchestrate and monitor the behaviour of the RODs operating as one component of the ATLAS TDAQ system.

3. Event flow model
The ROD responds to triggers issued either by the ATLAS Central Trigger Processor (CTP) system or by an external signal that emulates the CTP. Each trigger, known as a Level 1 Accept (L1A), initiates the digitalization of the time-slices and their transmission via high-speed fiber-optic G-Links to the off-detector electronics. The ROD must be able to process chamber data at a maximum L1A rate of 75 KHZ [4].

The data corresponding to one precision layer or four transverse layers - 192 channels - is relayed to a single SPU, where different degrees of feature extraction are performed based on a predefined configuration. The ROD must also capture a description of the L1A trigger, which is not only
necessary to satisfy TDAQ requirements, but it is also critical to ensure that all the independent data streams remain phase coherent with respect to one another. This information is referred to as the Trigger Information Summary (TIS) and is generated and buffered in the CTM.

Streams (depicted as arrow lines in Figure 3) refer to paths carrying event data into the SPUs, the RPU, and the TM (for event building). Streams are independent in the sense that they are not synchronized between one another. Thus SPU_0 might take longer than SPU_1 to process its share of event data corresponding to a given trigger. SPU_1 does not wait for SPU_0 to complete but instead it continues processing the contribution for the next trigger if any. This makes for a better utilization of the available bandwidth.

The five SPU contributions and the associated TIS are rendezvoused in the RPU. This phase is called Event Building and produces the event data for a chamber, which is an aggregate of cluster information and trigger description. The process of composing chamber events occurs in parallel on sides A and B. Originally, the RPU also performed feature extraction based on a neutron rejection algorithm. However, it was not deployed because it proved ineffective and took too long to run.

Finally, the two chamber contributions for a trigger are merged in the CTM to form a suitably formatted CSC event, which is transmitted down-stream to the ATLAS TDAQ system via the Read-Out Link. Figure 3 depicts this event flow model.

Figure 3. Event flow schema in the CSC ROD.

4. DPU software design
The DPU software structure is composed of three layers: the frame layer, which interacts with the hardware to ensure the event data are received and transmitted correctly; the builder layer, which assembles frames into segments (SPUs) or fragments (RPUs); and the feature extraction layer, which filters and reformats the event data.

Although not being a hard real-time system, the CSC ROD software must ensure that each event is processed within a time window in order to handle the 75 KHz constraint. To this end, assembly has been used to implement some functionality, although most of the code has been written in C++. Also, generic programming has been extensively used, not only to implement efficient linked lists, but also
and more importantly, to solve the performance degradation observed when using virtual methods\(^1\). The Curiously Recurring Template Pattern (CRTP) [6] provides an idiom to implement polymorphism at compile time and thus minimizes the overhead incurred by dynamic polymorphism.

4.1. Frame layer
The main significant achievement – and challenge – of the DPU design has been to make each data stream operate asynchronously at its own natural rate unconstrained by the behaviour of any other streams and with no external coordination required. Events are moved along a given stream at different rates, at different stages with each stage containing buffering, allowing for events to be pipelined as they progress. Whenever the buffering in a stage is momentarily insufficient, the stage asserts back-pressure to its predecessor. When the back-pressure reaches the stream’s head, trigger throttling is asserted. This involves asserting a busy signal to the LTP, which forwards it to the CTP to temporally stop issuing triggers. When buffer occupancy decreases below a certain threshold, the back-pressure condition is de-asserted and triggers are resumed by the CTP.

Data is transmitted in and out of the DSPs through two external I/O buses: the EMIF bus and the XB bus. The XB FPGA and the EMIF FPGA are located on the GPU daughter board next to the DSP (Figure 2) and contain the logic to control these channels and transmit data from and to the DPUs by governing the DMA engines in the DSPs. The XB channel is used to bring the event data into the SPU and the SPU contributions into the RPU. The EMIF channel is used to transmit the event contribution from the SPU to its corresponding RPU, while for the RPU this channel is used to send its merged events to the TM. The 5 SPU EMIF FPGAs are physically connected to the RPU XB FPGA, where an arbiter guarantees that data is transmitted in an orderly fashion. The throughput on this bridge is 35 MByte/sec, which results in a total ROD throughput of 70 MByte/sec., making the system sensitive to data occupancy.

The frame layer reserves 448 KB - out of the 512 KB of internal DSP memory - to allocate a number of fixed-size inbound (to the DSP) and outbound (from the DSP) frames. Frames are transferred by the DMA controller, which is externally governed by the XB and EMIF FPGA. For inbound frames, the DMA controller reads a frame descriptor from its on-board free-list FIFO and programs the DSP’s DMA engine to transfer the data into the corresponding buffer in the DSP’s internal memory. The memory address of the inbound frame is encapsulated in a frame descriptor and sent to the DSP via its serial interface. When the event data has been processed, the DSP writes the buffer’s frame descriptor back to the free-list FIFO on the XB FPGA via the serial interface so that it can be reused. Each inbound frame descriptor arrives in the DSP with an outbound frame descriptor, so similarly for outbound frames, the DPU writes a buffer’s frame descriptor to the EMIF FPGA’s work-list FIFO to program the DMA engine to transmit the frame downstream. When the DMA completes, the firmware puts the descriptor back on the free-list FIFO for reuse. Three words are transmitted via the serial link for each inbound frame, whilst two words are sent for outbound frames. For each event, the RPU receives five frames from the SPUs and one corresponding to the TIS. Given that three words describe an inbound frame and that the serial interface operates at 55 MHz, Equation 1 yields the theoretical maximum event rate.

\[
55 \text{ MHz} / (3 \text{ words} \times 32 \text{ bits/word} \times 6 \text{ frames}) = 95.5 \text{ KHz} \quad (1)
\]

\(^1\) Disassembling the code around a virtual function call showed that there were a lot of instructions and memory accesses that most likely were the cause of the overhead. Since this study was made however, new compiler versions have been released, which might have improved the implementation of run-time polymorphism.
4.2. Building layer
The building layer in the SPUs processes a configurable number of time-slices. These are sequentially received and assemble to build a segment, which is then handed off to the feature extraction layer. Since the duration of recording a pulse or hit spans several beam crossings, a new L1A may be declared while time-slices corresponding to a previous L1A are still being read out. The CTM handles this by flagging time-slices that belong to multiple L1A as shared. The building layer must account for shared time-slices and reuse them appropriately.

When a segment leaves the SPU, it becomes an input frame for an RPU. The aggregation of the five SPU segments and the TIS associated with the same trigger is known as event building. Since the SPU streams are independent, the contributions arrive asynchronously. Consequently, the RPU must maintain phase coherence and merge contributions from the same trigger. The algorithm to do so is based on a unique 24-bit sequence number associated to each trigger. This number has an immutable offset with respect to the Event ID contained in the TIS. Thus, contributions which share the same offset correspond to the same event in time.

4.3. Feature extraction layer
Feature extraction is encapsulated in so called handlers, which run in the context of the frame layer. Depending on the nature of the run, a different pair of SPU-RPU handlers is loaded. Currently, there are three types: physics handlers used during data-taking runs; pass-thru handlers, used in calibration and pedestal runs; and wave-form handlers, used to study with high resolution the bipolar pulse.

The CSC occupancy is quite low, that is, only a few channels contain meaningful information, whilst the rest are noise and leftovers from other beam crossings. In order to eliminate channels that are not hits and thus reduce data occupancy in the RODs, a sparcification algorithm has been implemented in the SPU physics handler. This involves two steps: a threshold elimination and a wrong time rejection. The former discards channels whose value is less than a predefined threshold. The latter provides an acceptance window of 75 ns. and requires that the waveform peak in the middle of the readout window. This is only fulfilled if the second or third sample is larger than the first and fourth sample. Since these handlers are configured for data taking runs, they must be able to process up to 75,000 events per second. To achieve these large sections of code were written in assembly to fully utilize the resources available in the DSP. The pass-through handler simply forwards all the chamber data. The waveform handler also performs sparcification but unlike the physics handler it can process up to 30 time-slices, which allows the bipolar waveform to be study with a better granularity. These handlers are only meant to be used in calibration or pedestal runs with low trigger rates.

5. HPU software design
The HPU is dedicated to a number of housekeeping responsibilities, which involve setting up, configuring and monitoring the different entities that form the ROD. These include loading the DPU code and configuration parameters retrieved from the RCC; flashing the firmware to the FPGAs; setting up the clock generation; supervising the transition module; and monitoring the ROD itself. An internal Finite State Machine (FSM) orchestrates the correct initialization, setup, configuration, starting and stopping of the different subsystems at the appropriate transition state. The ROD monitoring is encapsulated in a set of tasks, which are periodically invoked by a best-effort scheduler engine based on a priority level and period.

Bidirectional communication between the HPU and the RCC is performed via shared memory in a dual-port RAM. One side of this RAM is accessible to the HPU, whilst the other side appears as a slave device on the VME bus that interconnects the RCC and all the RODs in the same crate. The VME interface includes two types of asynchronous transfer channels. One channel encompasses a set of C++ structures located at well known memory locations. These structures encapsulate among others the configuration parameters, the ROD status, the monitoring status and the debugging commands, etc. The second communication channel groups three buffers used to transfer variable size data, such as the
DPU code, the threshold values and histograms. This channel is also used by the HPU to rely fatal conditions which could result in the run being halted.

The C++ classes that form the HPU software can be grouped in logical layers that share a common functionality. The Hardware Abstraction Layer encapsulates physical components, such as the Clock Generation, the Transition Module, the VME, etc. The DPU communication layer handles the unidirectional communication between the HPU and the DPUs. The RCC communication layer handles the bidirectional communication between the HPU and the RCC via VME. The Utilities layer includes a set of classes with functionality that can be used by other layers: Vector, Map, Print, etc. The FSM layer implements the ROD’s finite state machine, whereas the Scheduler layer implements the ROD’s scheduler engine and task interface. When possible, functionality is provided through a minimal interface that share a common syntax: service(), configure(), etc. This design and classification has yielded a non-monolithic orthogonal code that facilitates its maintainability and evolution.

5.1. HPU to DPU communication
The HPU is able to perform arbitrary access into the entire memory space of any of the DPUs. Data is physically transferred on the XB bus. Since this bus is also part of the event flow stream, an arbiter in the XB FPGA mediates requests from the different sources based on those requests and a relative priority. The HPU to DPU unidirectional communication is achieved through shared memory. All DPUs put aside a memory section of 12 KB starting at a well defined location. This area stores a general C++ structure containing further sub-structures with configuration data, monitoring parameters, statistics measurements and histogram objects. During booting, the HPU writes the configuration parameters needed by the DPU to initialize itself. Monitoring parameters, statistics and histograms, essential for debugging and performance evaluation, are periodically updated by the DPU and read by the HPU on user demand.

6. Performance
Each CSC ROD software release is validated by performing a set of regression tests, which include a L1A trigger rate scan with the CSC sub-detector and TDAQ infrastructure in place. The trigger rate is increased from 40 KHz to 100 KHz in steps of 10 KHz. For each rate, the CSC dead-time is recorded. The dead-time is defined as the fraction or percentage of total time where events cannot be recorded due to backpressure (busy assertion) exercised by one or more CSC RODs on the CTP. This procedure is repeated for various data occupancy volumes, which are artificially achieved by varying the threshold values via the so-called Sigma factor. It is important to notice that the conditions during these tests differ from those encountered during data taking runs: the triggers are random instead of physics-based and only the CSC sub-detector is included in the run. Nevertheless, the results of these tests provide a baseline to evaluate the behavior and performance of new releases.

Under this controlled environment, the software version to be used throughout 2012 was tested. The results are shown in Figure 4. At the end of the previous year, the average event size per ROD was recorded to be around 250 bytes. In 2012 this value might double as the LHC luminosity is expected to increase likewise. The Sigma factors that best encompass this range are 2.6 (320 bytes) and 2.1 (730 bytes). For the conventional physics handler, which sends the 4 time-slices retrieved from the FEE, the dead-time for a L1 trigger rate of 75 KHz varies from 2.2% to 8.1%. The CSC ROD performance is thus very sensitive to the data occupancy. Thus the CSC dead-time might reach unacceptable values as the luminosity increases. To handle this requirement, a new handler has been developed to retrieve 4 time-slices but only send the two centre ones. This reduces the data occupancy and thus improves performance, as seen in Figure 4 where the dead-time ranges 1.1% - 3.2 %.

To contrast these results, high-rate scans were performed during real physics runs (with stable beam) in August 2012. The RODs were configured with the two time-slice handler. Figure 5 shows how for run 208485 the CSC dead-time reaches 1% already at ~63 kHz. The average event size was approximately ~320 bytes per ROD, which does not differ much with the test results. The CSC is thus
working at its limit, the reason being the outstanding performance of the LHC, which has surpassed the estimated luminosity, thus increasing the pile-up and the data volume. To improve this scenario, the thresholds have been increased, thus reducing the data occupancy and increasing by 6 kHz the trigger rate achieved at 1% dead-time, as depicted in Figure 5 for runs 209550 onwards. There is still margin to selectively increase the thresholds on specific layers and chambers without losing too much efficiency. This strategy will be employed to compensate for an increase in luminosity until the end of 2012, when p-p runs will end.

Figure 4. CSC dead-time as a function of the trigger rate and data occupancy obtained in a controlled environment.

Figure 5. CSC dead-time as a function of the trigger rate obtained during physics runs.
7. Conclusion
Since the CSC chambers are located close to the collision point, the amount of data that needs to be analyzed and transferred is rather large. A design that has made the data streams independent and the use of optimization techniques and assembly have been paramount to obtain the maximum throughput from the available hardware. The CSC subsystem has been operating as expected since the first proton-proton collision was recorded in November 2009. However, the outstanding performance of the LHC, which has given luminosity values $> 7 \times 10^{33}$ cm$^{-2}$ s$^{-1}$ and pile-up $> \mu = 30$, has had an impact on the CSC. This coupled with the new requirements to handle higher energy levels: 13-14 TeV, higher luminosity: $1 \times 10^{34}$ cm$^{-2}$ s$^{-1}$ and higher trigger rates: 100 kHz [7] after the first long shutdown (end of 2014) has resulted in the decision to replace the current RODs with new ones.

Acknowledgements
This work was supported by the U.S. Department of Energy DE-FG03-91ER40679 and DE-AC02-76-SFO0515 and the National Science Foundation PHY-0612811. The authors would also like to thank Venetios Polychronakos for the numerous valuable discussions and the ATLAS TDAQ group for their continuous support.

References
[1] J. Dailing et al., “Performance and radiation tolerance of the ATLAS CSC on-chamber electronics”, in proc. 6th Workshop on Electronics for LHC Experiments, 2000.
[2] “Atlas muon spectrometer technical design report”, CERN / LHC / 97 – 22.
[3] I. G. Eschrich, “Readout Electronics of the ATLAS Muon CSC”, TWEEP 2008, Naxos, Greece.
[4] N. Drego et al., “Off-Detector Electronics for a High-Rate CSC Detector”, in proc. 6th Workshop on Electronics for LHC Experiments, 2000.
[5] G. Aad et al. [ATLAS Collaboration], “The ATLAS Experiment at the CERN Large Hadron Collider”, J. Inst. 3 S08003 (2008).
[6] http://en.wikipedia.org/wiki/Curiously_recurring_template_pattern
[7] ATLAS collaboration, “Letter of Intent Phase-I Upgrade”, http://cdsweb.cern.ch/record/1402470/files/LHCC-I-020.pdf?version=8