The CMS Phase-2 Trigger Upgrade

Interim Report to the LHCC

CMS Collaboration
Research and Development

1 Introduction

In this section, we describe the R&D programme currently underway within the L1 trigger community. In many cases, this R&D is shared with other CMS Phase-2 projects. Final hardware designs will need to be ready for production after 2021/2022. Prototype hardware for most systems must therefore be available between 2018 and 2020 with key technologies demonstrated by 2020.

The key technologies and challenges that we will seek to address in the R&D period can be summarised as: (1) Experience in designing hardware using Ultrascale class field programmable gate arrays (FPGAs) and survey of pluggable optics capable of running at links speeds between 3 Gb/s and 25 Gb/s; (2) Signal integrity testing of signals at full link speeds on modern printed circuit board (PCB) materials and backplanes, including capabilities/limitations of FPGAs and optics; (3) Understand how to minimise risk to FPGAs and to design systems with flexibility in mind via the use of interposer technology; (4) The latest FPGAs & optics consume significantly more power and it will be essential that the power delivery and thermal management are carefully addressed and validated in the initial design stage.

2 Advanced Processor Demonstrators

An extensive program of CMS HL-LHC L1 trigger hardware R&D along with accompanying firmware and software development is underway within the L1 Trigger Project, leading to a full capability demonstrators planned for 2019: (1) Advanced Telecommunications Computing Architecture (ATCA) form factor including Rear Transition Module (RTM); (2) Multi-Gigabit Transceiver (MGT) Links designed beyond 10 Gb/s line rates (i.e. 16 Gb/s and 25 Gb/s); (3) Efficient cooling of next generation FPGAs; (4) Next generation Intelligent Platform Management Interface (IPMI) and embedded Linux solutions; (5) Advanced random access memory (RAM) and FPGA interconnections. The program will identify design blocks suitable for re-use across platforms, either as reference designs or mezzanine boards and provide a next generation platform for ongoing software and firmware R&D work for the HL-LHC. The program will produce a general ATCA technology demonstrator (APd) with emphasis on trigger applications.

One R&D avenue targets the Xilinx C2104 package, along with 100 optical links running up to 25G based on Samtec Firefly modules and approximately 24 links to a Rear Transition Module (RTM) for enhanced versatility. Embedded Linux, an IMPI controller and Deep Memory will be included on mezzanines. The development of an initial APd includes an IMPI controller for ATCA blades with a Xilinx ZYNQ 7020 that can run general linux, supports 5 MMC’s, has 16 ADC inputs for monitoring and 1000BASE-T ethernet. One of the initial APd’s (APd1) will also includes an mezzanine providing a ZYNQ-based embedded Linux endpoint for ATCA blades. It will feature a Xilinx ZYNQ 7000 035/045 device with 8 MGT links 2 USB 2.0 ports 512 MB of DDR RAM (1066), on-board boot sources(QSPI and MicroSD Flash), 10 GbE capable ethernet and dedicated JTAG Master and Slave ports. The APd1 will be prototyped with a Controller Development Board, providing a host development system in the ATCA crate environment. It will contain the infrastructure of the APd1, including, power and the IPMI and embedded linux connectivity. The APd1 will also feature a low profile heat-sink with embedded fans being developed to provide a high-velocity air-source right at the FPGA.
Another R&D path targets a Peripheral Component Interconnect Express (PCIe) based card, known as MPUltra see Fig. ??, and a Serenity Extended Advanced Technology eXtended (ATX) motherboard (in design), which will explore links speeds up to 28 Gb/s, interposer technology, an alternative to ATCA, and a PCIe interface for slow control. The ATCA service card (in manufacture), which provides all the base infrastructure for an ATCA card and allows the evaluation of running 16 Gb/s to 25 Gb/s serial links over the backplane if required by CMS. The Service card and Serenity card are dual purpose in the sense that both are also designed to allow power / cooling tests to be conducted for the different form factors. The aim would be to bring all the strands of work together to produce a first full hardware prototype in 2018. Until then, if demonstrator systems for physics applications (e.g. endcap calorimeter or track-finder demonstrators) need to switch from a µTCA to an ATCA form factor, the ATCA services demonstrator could be designed to include the capability of hosting the MPUltra (or any similar card) via a PCIe connector. A board like this would be available in early 2018.

Figure 1: (Left) Embedded Linux Mezzanine (ELM) providing a ZYNQ-based embedded Linux endpoint for ATCA blades. (Right) The MPUltra is a PCIe card currently under test that was primarily built to explore new PCB technology, while also evaluating new Xilinx Ultrascale parts and Avago Optics.

Additional areas that we will study include modularity within the context of higher link speeds, where signal integrity needs to be validated, especially when using optical connectors. Applications with the Belle II Detector showed that some of the power modules on the carrier board and an Advanced Mezzanine Card (AMC) processor module could block airflow; part of the R&D effort will be devoted to card designs that enable efficient heat dissipation. Also, in the Belle II applications, some issues between channels inside the FPGA were observed, and the potential of crosstalk inside the FPGA is under investigation. Demonstrator boards will study modularity using an AMC as a processor, an ATCA form factor with routine and control as a carrier module, and RTM as auxiliary input-output channels. Input-output bandwidth will be increased with prototypes aiming a 26 Gb/s link speeds. Higher possibilities will depends on the test results. An Ultrascale FPGA will be used for the first prototype in 2017 and later an Ultrascale+ FPGA in 2018. Some tests from the Belle II application will be performed to better understand the issues observed in that context, with potential implications for CMS.

3 Configuration Infrastructure

The leading candidate for processing boards for the CMS Phase-2 Trigger is the ATCA standard. Central to this standard are the presence on each board of a low level controller called an Intel-
ligent Platform Management Controller (IPMC), and the use of Ethernet for high level control functions. As part of the Phase-2 Trigger R&D, two custom boards are being designed based on Xilinx ZYNQ System-on-Chip (SoC) devices. The first is an IPMC in a 244-pin mini Dual In-line Memory Module (MiniDIMM) form factor, and the second is a Linux-capable Transmission Control Protocol (TCP) endpoint in a custom form factor called the Embedded Linux Mezzanine (ELM). Together these boards support the card through startup and configuration into operation, in the benchtop lab, test beam, or production environment at LHC Point 5.

The ZYNQ-based IPMC is responsible for power-up and environmental monitoring of the ATCA card, including voltage and temperature. Running a Real Time Operating System (RTOS), the ARM Cortex-A series-based CPU is powerful enough to support TCP connections for network-based I/O, firmware upgrades, and a Joint Test Action Group (JTAG) controller that can assist with main board debug. Fast ADC channels integrated with the ZYNQ programmable FPGA logic can quickly detect fault conditions on the board and support rapid intervention to prevent damage in the event of over-temperature conditions or power supply faults, including a waveform capture capability around the time of the fault. Improvements over existing commercial IPMC solutions include faster response times to faults, wider input/output options, and additional monitoring features.

The ELM is responsible for the high-level control functions of the card, specifically those associated with operating the one or more FPGAs on the main board. Use of a ZYNQ device with its programmable FPGA logic gives the ELM significant flexibility to directly support a wide variety of main board auxiliary devices, such as optical modules, clock synthesizers and console ports. By controlling these devices from the ELM, this function is off-loaded from the main board FPGA(s), allowing that firmware image to be fully focused on the specific processing application. For those main board applications needing a very high-bandwidth (> 1 Gb/s) control connection, the ELM is capable of supporting a 10GbE interface. The flexibility of the ELM pinout and ZYNQ programmable logic allows the ELM to interface to main board FPGAs, either through fast parallel I/O or through MGT serial links. If local storage is needed (e.g. large LUT storage for fast configuration), a SSD can be added to the ATCA board and connected to the ELM.

4 Links

We have tested FireFly high-speed optical links for use in an Advanced Processor. Both 14 Gb/s and 28 Gb/s versions are available, although only the 14 Gb/s version has been tested at the time of this writing. Figure 2 shows the 14 Gb/s version with 4 TX and 4 RX channels and a short fiber connection. Also shown is a 10 cm electrical cable from FireFly compatible with either bandwidth options that was also tested. The optical links were tested using a Xilinx evaluation board, XUSP3S, with a Kintex Ultrascale FPGA on it (see Fig. 2). The QSFP+ optical link modules were replaced with custom connections to the FireFly links. In addition, the evaluation card was outfitted with 16 GB of DDR4 SODIMM memory. In the link tests we also evaluated the performance of Molex Impel connectors (see Fig. 2) for use in routing signals to/from a Rear Transition Module as well as for mezzanine cards. The connectors are rated up to 40 Gbps and are mounted on 100G PCB material, where a 15 cm differential line was also added. We conducted the link tests using PRBS-31 data patterns. In the first set of tests, we routed the signals through 1 m of OM4 fiber between the Firefly transmitter and receiver and through the Impel connectors that were placed either before the transmitter or after the receiver. The eye diagram with maximum pre-emphasis and output level applied is shown in Fig. 3 (left) when the connectors were placed after the FireFly receiver (worst case). In another
Figure 2: (Left) FireFly optical link packages and fibers with 4 TX and 4 RX channels at 14 Gbps (top), and a 10 cm FireFly electrical cable (bottom). (Middle) Xilinx XUSP3S evaluation board with FireFly optical links (left) and a 16 GB DDR4 SODIMM package (right). (Right) Molex Impel connectors evaluated for connections to/from a Rear Transition Module.

test, we replaced the Molex connectors with a 10 cm cable attached to the optical link, either before the transmitter or after the receiver. That eye diagram is shown in Fig. 3 (right), again for the worst case when the cable is placed after the receiver. The conclusion from the FireFly link tests is that using FireFly links at 14 Gbps with either a 10 cm cable or a Rear Transition Module connection is definitely possible.

Figure 3: Eye diagram of the FireFly optical link transmission at 14 Gbps and passing through a Molex Impel connector plus 15 cm PCB differential line (left) and through a 10 cm FireFly cable (right).

5 Memory

Using the same Xilinx evaluation card used for the previously discussed Firefly optical link tests, we examined the performance of DDR4 memory for use as a large look-up table in track-fitting applications. Using a 16 GB module, the maximum latency was measured to be 68 clocks at 4.284 ns spacing, or approximately 12 BX. However, it can vary between 6 to 12 BX. The addressing was completely random in this test as would be the case for triggering. An address management scheme would need to be developed to avoid hitting the same memory bank too soon, but this is not a problem. For example such a scheme was developed for using RLDRAM in the current CMS endcap muon trigger. Our conclusion is that DDR4 memory is suitable for use in the trigger, and allows us to reach very large memory bank sizes.

6 Firmware

The role of firmware in the success of the Phase 1 Trigger Upgrade is substantial, and can scarcely be overstated. Firmware is employed to receive and align detector data, by the algorithms that process it, and in the formatting and delivery of data to the DAQ system. While the diversity in the trigger is large in terms of the types of source data, how it is formatted, the links
6.1 Management and Build Systems

speeds and processing required, there are also similarities across the various components. Regardless of the associated detector, links carrying data into the trigger must be aligned, and the integrity of that data validated. Errors must be reported, and data for readout must be passed through a series of buffers on its way to the DAQ system. Control computers must supervise the firmware. Offline data verification requires agreement between operational firmware and emulators.

For Phase 2, a number of R&D topics on firmware have emerged. Some of them are evolutionary, building and improving on the work done for Phase 1. Others are revolutionary, reflecting the emergence of new technologies. In pursuing this R&D, the challenge is to preserve the balance between the virtues of flexibility for specific needs and the advantages of standardization for recurring tasks.

6.1 Management and Build Systems

Original text provided by Dave N. Modified text provided by Tom G. Current text awaiting input from Dave N. List of concrete R&D topics adapted by Rick C. from original text by Dave N.

Firmware partitioning is a topic of great interest. Finding efficient methods to separate firmware for overhead tasks like board initialization or generic FPGA control from application algorithms has obvious advantages. One method is the formal division of firmware for a FPGA device into infrastructure and algorithm sections. Another demonstrated successful method is to segregate these functions along physical device lines, such that there is a control device and a processing (algorithm) device. Looking forward into Phase 2, one promising emerging strategy is to make use of a modular design approach based on firmware libraries, integrated right into the vendor-supplied synthesis tools like Vivado. Work on all of these methods will continue, giving trigger applications a variety of proven partitioning strategies based on the specific needs of the application.

The idea of a modular, library-based approach to firmware feeds directly into a firmware R&D topic on the development of re-usable modules. Many general functions already have well-established solutions, either as vendor-supplied or CMS-written HDL cores. Nevertheless, there are several areas that could benefit from a more common approach. For example, low-level link firmware, interfacing to and aligning data on serial links, and protocol packers/unpackers for 64b66b transport could benefit from standardization. Indeed, large-scale use of FPGAs in other Phase 2 subsystems will occur, and there may be significant opportunities to collaborate across the CMS detector on many firmware libraries that are also relevant to the Level-1 Trigger. Successful identification of a set of “best practices” for recurring functions will benefit the Trigger in Phase 2.

Another area of significant interest is the development of a consistent approach for migrating algorithms between the simulation test-benches, software emulators and implemented FPGA firmware. High Level Synthesis is one promising new technique to evaluate, but whatever tools and techniques are developed will be of central importance in the development and deployment of the Phase 2 CMS Trigger System.

Along these lines of reasoning, we plan to carry out an R&D programme over the coming two years into a coherent strategy for firmware development, with the following goals: (1) Investigate a common firmware build system, as a front-end to FPGA vendor and third-party Computer Aided Design (CAD) tools (including high level synthesis tools where appropriate); (2) Develop standards for control of firmware blocks and interfaces between them; (3) Develop a verification / emulation strategy that allows a consistent approach at all stages of system
testing, from basic simulation testbenches, to system test with real data; (4) Investigate the requirements and strategy for physical partitioning of FPGA devices to allow consistent and reproducible firmware performance when implemented. In each of these areas, we will examine best practice in both the field and in industry, and will seek to cooperate with other CMS projects undertaking firmware development tasks of similar scale.

6.2 High Level Synthesis

In the past few years improvements in performance and capabilities of FPGA synthesis and implementation tools have been observed. At the same time, due to the increased particle density seen in the detector caused by the higher collision rate of the HL-LHC, we also face a growing complexity of trigger algorithms at Level 1. This motivates the utilization of tools and methodologies that operate at higher abstraction levels. High-Level Synthesis (HLS) provides a direct path to generate production quality register transfer level (RTL) firmware implementations from algorithm specifications written in high-level languages (HLLs); particularly in C/C++ in our use case. HLS enables accelerated design time while keeping full control over the choice of optimal algorithm architecture exploration; it provides mechanisms to investigate the proper level of algorithm parallelism while taking into account implementation constraints. HLS automates an otherwise tedious manual process, eliminating the source of many design errors and accelerating a long and iterative part of the development cycle, while significantly reducing the verification burden. It also offers effective code reuse by working at a higher abstraction level.

HLS tools and methodologies have matured considerably over recent years, yet they do come with some challenges. Firstly, HLS still considered a disruptive technology. Use of HLS implies a change in the methodologies, in the firmware design processes, and to some extent, in the skillset required. Also, not all C/C++ coding styles are equal in terms of Quality of Results (QoR), and there is the potential for producing poor quality RTL if the C/C++ code is not well suited for HLS. Good coding style not only requires an understanding of the underlying hardware architecture of a given algorithm and deep familiarity with target technology, but also an understanding of how HLS works.

The firmware for the CMS Endcap Muon Track Finder for Phase 1 already successfully uses HLS in production online. Other recent case studies have demonstrated the above-mentioned benefits on a realistic algorithm implementation (CMS Phase-1 calorimeter trigger). By employing HLS tools we were able to implement production-ready RTL code that rivals the performance of the existing trigger implementation, both in terms of smaller resource utilization and algorithm latency reduction, and importantly, at a fraction of the time for design implementation and verification cycle. R&D efforts on the subject of HLS will focus on further evaluation of the tools and methodologies, with the aim to determine how working at a higher abstraction level and accelerated automation benefits Phase-2 trigger algorithm development.

6.2.1 Case Study: simple electromagnetic cluster finder algorithm synthesized using HLS tool

We show below the performance of cluster finder algorithm for calorimeter trigger card, which we synthesized using HLS Vivado tool. We consider \((17\eta \times 4\phi)\) electromagnetic towers equivalent to \(17 \times 4 \times 5 \times 5\) crystals as input to the trigger card and study the algorithm performance for two target devices: Xilinx Virtex-7 690T, used in Phase1 CTP7 cards, and Xilinx Virtex UltraScale+ VU9P, used for illustration purpose for Phase2, the exact parts are still under consideration.

Clustering algorithm is performed in following steps:
• Step 1. Identify for every $5 \times 5$ crystal (trigger tower), the peak crystal position in eta-phi. The energy-weighted position algorithm is used for finding the peak position as shown in Fig. 4.

Figure 4: (Left) $5 \times 5$ crystal is shown with the seed crystal (brown color). In order to identify the seed, we calculate the energy sum in phi strips and eta strips (Middle) and use that to estimate the energy weighted peak position (Right).

• Step 2. Calculate the $3\times3$ cluster energy around the peak crystal as shown in Fig. 5

Figure 5: Shown is the $3\times3$ cluster around the peak energy position in one tower.

• For each tower processed in the card, $17\eta \times 4\phi$ towers, we provide the output as peak position in eta and phi, $5 \times 5$ tower energy sum and $3 \times 3$ cluster energy around the peak position.

• Step 3. Clusters that have their peak positions at the tower edges are stitched with the neighbour towers, we merge the energy of smallest cluster to largest cluster and then assign the smallest cluster energy to be 0 GeV to avoid double counting as shown in Fig. 6

For the Xilinx Virtex-7 690T, target clock used is 240MHz (4.16 ns), for the Xilinx Virtex UltraScale+ VU9P - 320MHz (3.12 ns). Performance and utilization estimate is shown in Table 1 and Table 2 respectively.

<table>
<thead>
<tr>
<th>Device</th>
<th>Latency min</th>
<th>Latency max</th>
<th>Interval min</th>
<th>Interval max</th>
<th>Pipeline Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex-7 690T</td>
<td>53</td>
<td>53</td>
<td>6</td>
<td>6</td>
<td>Function</td>
</tr>
<tr>
<td>Virtex UltraScale+ VU9P</td>
<td>72</td>
<td>72</td>
<td>8</td>
<td>8</td>
<td>Function</td>
</tr>
</tbody>
</table>

For both the target devices, we obtain Absolute latency of 8.8 BX.

7 System Level Considerations

Text to be provided by Alessandro T. and Dave N.
Figure 6: Shown is the 2x2 towers in $\eta - \phi$. We merge the clusters where the seed is at the tower boundary as shown by red color.

Table 2: Utilization estimates.

<table>
<thead>
<tr>
<th>Name</th>
<th>Virtex-7 690T</th>
<th>Virtex UltraScale+ VU9P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BRAM18K</td>
<td>DSP48E</td>
</tr>
<tr>
<td>DSP</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Expression</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FIFO</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Instance</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Memory</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Available</td>
<td>2940</td>
<td>3600</td>
</tr>
<tr>
<td>Utilization(%)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>