CentOS Linux for the ATLAS MUCTPI Upgrade

R. Spiwoks, A. Armbruster, P. Czodrowski, N. Ellis, P. Farthouat, S. Haas, A. Kulinska, A. Marzin, P. Papageorgiou, T. Pauly, S. Perrella, M. Saimpert, P. Vichoudis, and T. Wengler

Abstract—A new Muon-to-Central-Trigger Processor Interface (MUCTPI) was built as part of the upgrade of the ATLAS Level-1 trigger system for the next Run of the Large Hadron Collider at CERN. The MUCTPI has 208 high-speed optical serial links for receiving muon candidates from the muon trigger detectors. Three high-end field-programmable gate arrays (FPGAs) are used for real-time processing of the muon candidates, for sending trigger information to other parts of the trigger system, and for sending summary information to the data acquisition and monitoring system. A System-on-Chip (SoC) is used for the control, configuration, and monitoring of the hardware and the operation of the MUCTPI. The SoC consists of an FPGA part and a processor system (PS). The FPGA part provides communication with the processing FPGAs, while the PS runs software for communication with the run-control system of the ATLAS experiment. In this article, we will describe our experience with running CentOS Linux on the SoC. Cross-compilation together with the existing framework for building the ATLAS trigger and data acquisition (TDAQ) software is being used in order to allow the deployment of the TDAQ software directly on the SoC.

Index Terms—ATLAS, CentOS, Linux, Muon-to-Central-Trigger Processor Interface (MUCTPI), System-on-Chip (SoC).

I. INTRODUCTION

ATLAS is a general-purpose experiment at the Large Hadron Collider (LHC) at CERN [1]. It observes proton–proton collisions at a center-of-mass energy of almost 14 TeV, with a bunch crossing (BC) every 25 ns (40 MHz) and up to 80 pile-up collisions expected for the next run (Run 3), starting in 2022. This results in more than $10^9$ interactions per second and requires the use of a trigger system in order to select the events most interesting for physics studies within constraints on the maximum rate that can be recorded.

The ATLAS trigger and data acquisition (TDAQ) system [2] consists of a two-level trigger system. The Level-1 trigger is based on custom electronics and firmware, and reduces the event rate to 100 kHz. The high-level trigger is based on commercial-off-the-shelf computers and network, and processing software, and reduces the event rate to around 1.5 kHz (peak) which corresponds to a data rate of around 1.5 GByte/s. The Level-1 trigger uses reduced-granularity information from the calorimeters and information from dedicated muon trigger detectors, the resistive plate chambers (RPCs) in the barrel region ($|\eta| < 1.05$) and the thin-gap chambers (TGCs) in the endcap and forward region ($1.05 < |\eta| < 2.5$), where $\eta$ is pseudo-rapidity, see Fig. 1.

The Muon-to-Central-Trigger-Processor Interface (MUCTPI) receives muon candidate information from 208 muon Sector-Logic modules, and combines the information. It avoids double counting of single muons, which are detected by more than one muon sector due to the geometrical overlap of the chambers and the trajectory of the muons in the magnetic field. This is called overlap handling. A typical overlap of a number of barrel and endcap sectors is shown in Fig. 2.

Up to 16 highest transverse-momentum candidates after taking into account the overlap handling are sent to the topological trigger processor (L1Topo), while muon multiplicity information is sent to the central trigger processor (CTP). The CTP combines all trigger information and takes the final Level-1 Accept or Reject decision.

II. MUCTPI UPGRADE

For Run 3 of the LHC starting in 2022, higher rates of muons are expected. In order to enhance the trigger selectivity, more muon candidates per sector and more information per candidate will be sent, requiring higher bandwidth between the muon sector logics and the MUCTPI. Improved overlap handling allowing for possible overlap between octants that was previously not supported will be provided. This requires more processing power. In addition, muon candidates with the full positional granularity will be sent to the topological trigger processor, and muon-only topological processing could possibly be carried out in the MUCTPI. Furthermore, the MUCTPI

Fig. 1. Schematic of the ATLAS TDAQ systems [3].
should be future proof, allowing for further improvements for Run 4 starting in 2027 [4].

The previous Versa Module Europa (VME)-based system [5] with bulky electrical cables and 18 modules will be replaced by an Advanced Telecommunications Computing Architecture (ATCA)-based system with optical cables and a single module using state-of-the-art field-programmable gate arrays (FPGAs). Two Muon Sector Processors (MSPs) based on Xilinx Virtex Ultrascale + FPGAs [6] receive the muon trigger information for the two halves of the complete system. The MSPs implement the overlap handling and calculate multiplicities for each half of the ATLAS detector. This information is sent to the Trigger Readout Processor (TRP) based on a Xilinx Kintex Ultrascale + FPGA [6], which combines the partial results and sends overall multiplicities and other flags to the CTP. The two MSPs further send lists of muon candidates with full-precision information to the Topological Trigger Processor. The architecture of the upgraded MUCTPI is shown in Fig. 3, and a photograph of the current prototype is shown in Fig. 4.

### III. System-on-Chip

In the previous VME-based system, a Single-Board Computer (SBC) in the VME crate controlled the MUCTPI’s modules. The SBC was based on Intel x86_64 processors and was running Scientific Linux CERN (SLC) [7]. The user application software and its development were fully integrated into the ATLAS TDAQ system. In the new ATCA-based MUCTPI, a System-on-Chip (SoC) implemented on the MUCTPI itself is used for control, configuration, and monitoring of the hardware and the operation of the MUCTPI, see Fig. 5.

The SoC consists of a processor system (PS), which is like a CPU, and a programmable logic (PL), which is like an FPGA. The PS of the SoC is based on multicore ARM processors. It has memory and peripherals, e.g., Gigabit Ethernet, I2C, serial peripheral interface (SPI), and general-purpose input/output (GPIO). The Gigabit Ethernet is used for communication with the ATLAS run-control system, while the other interfaces are used for the control of the hardware of the MUCTPI. The PS of the SoC runs software, which could be a “bare-metal” application or a full-blown operating system like Linux. The PS represents the software side of the SoC and integrates it into a local area network.

The PL of the SoC has logic cells, memory block, and input/output (IO) links. It uses the IO links in order to implement interfaces to the other processing FPGAs of the MUCTPI for configuration of the FPGAs and for reading and writing from registers and memories implemented in those FPGAs. The PL represents the firmware side of the SoC and integrates it with the hardware of the MUCTPI.
Since the MUCTPI uses Xilinx FPGAs, it was natural to use a Xilinx SoC, and a choice was made for the Xilinx Zynq Ultrascale + multiprocession system-on-chip (MPSoC) ZU3EG [6], which has a quad-core ARM Cortex A53 processor with a 64-bit architecture (aarch64), runs at 1.2 GHz and has 4 GByte of memory.

The Xilinx/Vivado and Xilinx/software development kit (SDK) tools [6] provide several of the files required for booting the SoC, including the bitstream file for the PL, the first-stage boot loader for initializing the hardware and for loading the secondary program loader, and the device tree file for use with the Linux kernel. For the operating system, there is a choice. Xilinx provides PetaLinux [8] or alternatively Xilinx meta-layers to be used with the Yocto/OpenEmbedded framework [9], which can be used to build the Linux kernel, the Linux root file system, and the U-Boot for loading the kernel, the device tree, and the root file system.

In the ATLAS experiment, the MUCTPI will be running within the ATLAS Technical Control Network (ATCN), which contains several thousands of network nodes of different types, belonging to many different detector groups [10]. It will be running over long periods of several years; a run of physics data taking typically lasts for three years [2], [4]. Strict computer security rules apply and only certified operating systems are allowed on the ATCN [11]. Systems with noncertified operating systems are isolated behind a gateway host.

Another very important aspect is the maintenance and long-term support of the operating system and the user application software, not only for security patches but also for continuous improvements. In addition, system administration support is needed for the systems used for software development and the systems used with the hardware prototypes. This requires considerable effort, in particular, for sometimes rather small groups, for which the main expertise is on developing and maintaining the hardware and firmware of the system, and not the software. Therefore, an operating system has to be chosen, for which central support is provided by CERN’s Information Technology (IT) department or the experiment’s system administration, or both.

Finally, the control software for the MUCTPI will have to be integrated into the ATLAS TDAQ system [12] for it to be operated at the experiment. One possible approach to achieve this is to use a remote-procedure-call like protocol over the network; another is to implement the TDAQ run control applications directly on the SoC [13]. We have chosen the latter, which has the advantage of removing the intermediate protocol layer and of reusing existing software, and which can be achieved with a similar effort as for the previous VME-based system, as will be shown in the following.

IV. USE OF CentOS LINUX

The operating system is key to the operation of the SoC and its integration into the ATCN and the ATLAS TDAQ software [12]. CERN and consequently the ATLAS experiment chose the CentOS Linux distribution [7], [14]. It is a security-certified operating system at CERN and allowed on the ATCN. Support for it is available from the CERN IT department. The ATLAS TDAQ software uses CentOS. It is fully adequate for use in an online environment. The same operating system was also used for the previous VME-based system of the MUCTPI and a lot of experience was gained with it; all other groups in the ATLAS experiment, that continue to use VME-based system, continue to use CentOS as the operating system.

CentOS is equally available for the aarch64 architecture. We opted for using CentOS on the SoC of the MUCTPI, not for its potential technical superiority over other distributions, but rather for the advantage of using the same distribution as the one for which support is already provided by CERN and ATLAS, and from which software can be reused.

Furthermore, that choice is also recommended by the SoC interest group at CERN [15]. The interest group held a workshop on SoC issues in 2019 [16] and continues with a series of meetings [17] following up on issues of common interest, including support for CentOS/aarch64.

Unfortunately, the current CERN and ATLAS policies still do not allow the MUCTPI to be directly connected to the ATCN, but the advantages of using a common operating system with the rest of the experiment are considered more important. It is also a way of preparing for the future: once the administrative issues will be solved, the workflow of building the TDAQ and user application software will be the same. The MUCTPI will also be used in Run 4, such that the experience gained in Run 3, running Centos on the SoC with a gateway PC will be very useful for Run 4. Furthermore, the workflow will be very similar, even if a different Linux distribution will be chosen for Run 4.

Using CentOS Linux on the MUCTPI SoC is possible because the MUCTPI does not have any specific real-time requirements and the SoC does not place any resource limitations on the application, as other embedded systems might do. In fact, we are going to show that it can be used in exactly the same way as the previous system with a VME-based SBC.

 Practically, for the MUCTPI, the CentOS/aarch64 operating system is installed on a host PC using the Dandified YUM (Yellowdog Updates, Modified) (YUM) cross installer [18]. This makes use of the quick emulator (QEMU) [19] and the Linux kernel support for miscellaneous binary formats [20]. The full recipe can be found on the web pages of the SoC interest group [15].

The cross-installed CentOS/aarch64 root file system is used together with a Linux kernel using the drivers and settings required by Xilinx, and the U-Boot loader built with the Yocto/OpenEmbedded framework—it could equally be built using Xilinx PetaLinux. The root file system is mounted from a host PC using network file system (NFS) for simplicity, but could also be copied to an secure digital (SD) card.

Booting the SoC with the files described above provides a full operating system in the same way as with the previous VME-based SBCs. The firmware/software developers can use ssh for login, bash, python, tcl, expect, etc., for scripting. A Xilinx Virtual Cable (XVC) server [6] can run on the SoC and allows firmware developers to debug the processing FPGAs from the SoC. The system provides NFS and network time
protocol (NTP) for mounting other file systems from the host PC. Other Linux tools for networking like ip and ifconfig or a dynamic host configuration protocol (DHCP) client are also available, as well as the iptables tool for network security. Many other packages can be added to the installation easily. As a result, the SoC behaves exactly like any other CERN Linux host in the ATLAS TDAQ system.

V. User Application Software

In order to add user application software accessing the MUCTPI’s specific hardware features, and in order to integrate the MUCTPI into the ATLAS TDAQ run-control system, cross-compilation is used. The gcc/aarch64 compiler from the CentOS repository is not necessarily up to date, but developer.arm.com or building from the source from gcc.gnu.org can be used to get an up-to-date cross-compiler.

With CentOS as the system root, the software development flow from the previous MUCTPI generation can be reused:

1) All registers and their bit fields, as well as memories and FIFOs of the processing FPGAs of the MUCTPI are described in an extensible markup language (XML) file.

2) We developed a dedicated software tool, which takes the XML file and generates VHIC (very high-speed integrated circuit) hardware description language (VHDL) code for the firmware of the FPGAs, as well as C++ code for the software for the SoC, effectively providing firmware/software codevelopment [21].

3) The registers, memories, and FIFOs are mapped into the PS part of the SoC using advanced extensible interface (AXI), and are accessed as user input–output (/dev/uio) device files.

4) The C++ code opens the /dev/uio device files and executes single read/write functions. In addition, we have also developed a kernel module, which allows using the direct memory access DMA) engine of the SoC PS for reading/writing larger blocks of data efficiently.

The low-level software developed in this way provides access to the complete functionality of the hardware of the MUCTPI and its processing FPGAs. The automatic generation of software provides an air position indicator (API), as well as menu programs for easy use by a firmware/software developer.

In addition, Python wrappers for that software are automatically generated using Simplified Wrapper and Interface Generator (SWIG) [22]. They allow firmware developers to write complex test scripts for interactive debugging.

The cross-compilation also works within the framework of CMake [23], which is the build tool chosen by the ATLAS TDAQ and the MUCTPI teams. The ATLAS TDAQ software can be cross-compiled in the same way as the low-level software. This allows us to reuse existing software and to develop run control and monitoring applications, which are accessing the MUCTPI’s functionality using the low-level software described above, and which are running directly on the SoC.

An overview of the software architecture is shown in Fig. 6. In addition, the Worldwide LHC Computing Grid (WLCG) [24] provides software builds for the aarch64 architecture, which allows us to use, e.g., ROOT [25] as a library for histograms, graphs, and other objects. In this way, the run-control application can produce histograms to be sent to the ATLAS TDAQ run-control system.

A simple run control application and an example of a monitoring task providing a histogram were already developed, and presented before [26]. While those tasks previously were built using a SDK prepared by the Yocto/OpenEmbedded framework, they are now built using the CentOS root file system and the cross compilation described in this article. The run control and monitoring tasks are now still rather simple, but they will be extended with functionality of the firmware of the MUCTPI’s processing FPGAs, which is currently still under development.

As mentioned above, due to the ATLAS policy, the MUCTPI is currently still not allowed to run directly on the ATCN. Therefore, a TDAQ gateway application running on the gateway PC was developed [27]. That application works fully at the user level and relays all ATLAS TDAQ commands, configuration, and monitoring data between the ATLAS TDAQ services on the ATCN and the run control application and monitoring on the MUCTPI. The gateway application will only be required until the MUCTPI will be allowed directly on the ATCN. It is transparent for the running of the run control and monitoring task.

The requirements on the processing resources of the SoC are moderate: there are no real-time requirements, and the processing requires only a relatively moderate level, as was already the case for the previous VME-based system using SBCs. The SoC chosen also compares very much to the latest generation of the previous SBC, which had a quad-core Intel Atom processor, which ran at 1.9 GHz, and which had 4 GByte of memory.

During the configuration, the SoC reads all data necessary from the host PC and writes them to the processing FPGAs. This includes the actual binary configuration files of the FPGAs themselves, as well as configuration data for the register and memories. The processing and monitoring of trigger data are executed in the processing FPGAs of the MUCTPI. The SoC reads out integrating counters, per-bunch counters, and some selected trigger data for event monitoring. The data are transferred to other ATLAS TDAQ services at
relatively low rates, fully compatible with the Gigabit Ethernet interface. A second Gigabit Ethernet interface is available in case the monitoring requirements increase.

Finally, we have developed a single setup script to set up the host PC for development, i.e., for cross-compilation, or the SoC of the MUCTPI for deployment. In this way, the developer or user does not notice the “cross” environment and the flow of software development works exactly as before when using the VME-based SBC.

As a result, the MUCTPI low-level software, including menu and test programs, runs directly on the MUCTPI’s SoC. In addition, the low-level software is used by ATLAS TDAQ run-control and monitoring applications to integrate the MUCTPI into the ATLAS TDAQ run-control system.

VI. Conclusion

A new MUCTPI was built in the ATCA form factor and using an SoC. The SoC is used for configuration, control, and monitoring of the hardware and the operation of the MUCTPI. The SoC runs the CentOS Linux operating system. CentOS is used together with cross-compilation for the SoC. User application software can be developed with a development flow that is the same as before, when using a VME-based system and an SBC. It reuses existing software for the ATLAS TDAQ run control application in order to integrate the MUCTPI into the ATLAS TDAQ run control system.

It has also to be noted that there are many projects for the High-Luminosity LHC (HL-LHC) starting in 2027, in particular in the ATLAS and Compact Muon Solenoid (CMS) experiments, which are planning to use SoCs. There are projections of several thousands of SoCs to be installed in the experiments. Using a common Linux operating system will provide them with the network security and the system administration support necessary, and will allow them to reuse existing workflows and software.

ACKNOWLEDGMENT

The authors would like to thank M. Wittgen, Stanford Linear Accelerator Centre (SLAC), USA, for the original idea of cross installing the CentOS root file system, and S. Kolos, University of California Irvine, USA, for providing the ATLAS trigger and data acquisition (DAQ) gateway application.

REFERENCES

[1] G. Aad, “The ATLAS experiment at the CERN large hadron collider,” J. Instrum., vol. 3, Aug. 2008, Art. no. S08003.
[2] G. Aad et al., “Technical design report for the phase-I upgrade of the ATLAS TDAQ system,” ATLAS Collaboration, Washington, DC, USA, Tech. Rep. CERN/LHCC/2013-018, 2013, Accessed: Mar. 4, 2021. [Online]. Available: https://cds.cern.ch/record/1602235.
[3] ATLAS Collaboration. Approved Plots TDAQ. Accessed: Mar. 4, 2021. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ
[4] The ATLAS Collaboration, “Technical design report for the phase-II upgrade of the ATLAS TDAQ system,” Washington, DC, USA, Tech. Rep. CERN/LHCC/2017-020, 2017. Accessed: Mar. 3, 2021. [Online]. Available: https://cds.cern.ch/record/2285584
[5] M. V. S. Oliveira et al., “The ATLAS level-1 muon to central trigger processor interface for run 2 of the LHC,” J. Instrum., vol. 10, Feb. 2015, Art. no. C02027.
[6] Xilinx. Accessed: Jul. 7, 2021. [Online]. Available: http://www.xilinx.com
[7] Linux at CERN. Accessed: Jul. 7, 2021. [Online]. Available: https://linux.web.cern.ch
[8] Xilinx PetaLinux. Accessed: Jul. 7, 2021. [Online]. Available: http://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html
[9] The Yocto Project. Accessed: Jul. 7, 2021. [Online]. Available: http://www.yoctoproject.org
[10] S. Ballestero et al., “ATLAS TDAQ system administration: Master of puppets,” J. Phys., Conf. Ser., vol. 898, no. 8, 2017, Art. no. 082002.
[11] S. Luersd, “Computer security at CERN,” USENIX, Aug. 2014, vol. 39, no. 4.
[12] ATLAS Collaboration, “The ATLAS data acquisition and high level trigger system,” J. Instrum., vol. 11, no. 6, Jun. 2016, Art. no. P06008.
[13] R. Spiwoks et al., “Run control communication for the upgrade of the ATLAS muon-to-central-trigger-processor interface (MUCTPI),” in Proc. ICALEPSCS, Barcelona, Spain, 2017, p. 4. Accessed: Mar. 4, 2021. [Online]. Available: http://accelconf.web.cern.ch/icalepscs2017/papers/tupha071.pdf
[14] The CentOS Project. Accessed: Jul. 7, 2021. [Online]. Available: http://www.centos.org
[15] SoC Interest Group System-On-Chip@Cern.ch. Accessed: Jul. 7, 2021. [Online]. Available: http://wiki.cern.ch/wiki/bin/view/SystemOnChip/WebHome
[16] CERN. (2019). SoC Workshop. [Online]. Available: http://indico.cern.ch/event/799275
[17] SoC Interest Group Meetings. Accessed: Jul. 7, 2021. [Online]. Available: https://indico.cern.ch/event/799275
[18] Software Package Manager DNF. Accessed: Jul. 7, 2021. [Online]. Available: https://fedoraproject.org/wiki/DNF
[19] The QEMU Fast Processor Emulator. Accessed: Jul. 7, 2021. [Online]. Available: http://www.qemu.org
[20] Linux Kernel Support for Miscellaneous Binary Formats. Accessed: Jul. 7, 2021. [Online]. Available: http://www.kernel.org/doc/html/latest/admin-guide/binfmt-misc.html
[21] The ATLAS L1CT Hardware Compiler. Accessed: Jul. 7, 2021. [Online]. Available: http://gitlab.cern.ch/atlas-tdaq-software/L1CTHardwareCompiler
[22] Simplified Wrapper and Interface Generator. Accessed: Jul. 7, 2021. [Online]. Available: http://swig.org
[23] The CMake Build Framework. Accessed: Jul. 7, 2021. [Online]. Available: http://cmake.org
[24] Worldwide LHC Computing Grid. Accessed: Jul. 7, 2021. [Online]. Available: http://wlcg.web.cern.ch
[25] ROOT Data Analysis Framework. Accessed: Jul. 7, 2021. [Online]. Available: http://root.cern
[26] R. Spiwoks et al., “Run control software for the upgrade for the ATLAS muon to central trigger processor interface,” in Proc. CHEP, Sofia, Bulgaria, 2018, p. 5. Accessed: Mar. 4, 2021. [Online]. Available: https://www.epj-conferences.org/articles/epjconf/abs/2019/19/epjconf_chep2018_01034.epjconf_chep2018_01034
[27] S. Kolos. IPCGatewayProxy. Accessed: Jul. 7, 2021. [Online]. Available: https://gitlab.cern.ch/atlas-tdaq-software/IPCGatewayProxy