# AthXRT: Centralized FPGA Management for Accelerated Algorithms in Athena

*Quentin* Berthet[1,*], on behalf of the ATLAS Collaboration

[1]DPNC, University of Geneva, CH-1205 Geneva 4, Switzerland

**Abstract.** The integration of FPGAs as heterogeneous hardware accelerators in high-performance computing environments, such as the Athena framework used in the ATLAS experiment, presents significant opportunities for algorithm acceleration. However, it also introduces challenges in device configuration management. AthXRT is a centralized service designed to simplify and streamline FPGA configuration file management within the Athena framework. Supporting both native XRT and OpenCL APIs, this service offers a flexible, future-proof solution for FPGA management, enabling the seamless integration of accelerated algorithms into the Athena environment.

ATL-DAQ-PROC-2025-002
30 January 2025

## 1 Introduction

The increasing demand for computational power in high-energy physics (HEP) has driven the exploration of heterogeneous computing solutions, including the use of FPGAs as accelerators. FPGAs, with their massively parallel processing capabilities and energy-efficient design, are being investigated as potential solutions to accelerate data-intensive tasks in the Event Filter (EF) of the ATLAS experiment.

The Athena framework [1], the software infrastructure used in the ATLAS experiment [2], integrates a variety of algorithms and computational techniques for both online trigger and offline physics object reconstruction. In the context of the Phase II upgrade of the LHC and with the increase in computing power resulting from higher luminosities during Run 4 of the LHC, FPGAs are being considered as accelerators to offload computations to dedicated hardware. Currently, tracking, muons and calorimeter sub-systems of the EF are actively exploring FPGA-based solutions to accommodate the increased workload. Additionally, other online and offline algorithms within Athena may also benefit from hardware acceleration in the future.

Despite significant efforts to simplify their usage, FPGA development and operation remain demanding and differ fundamentally from traditional CPU and GPU software workflows. One key difference is that FPGAs require a configuration file (commonly referred to as a bitstream) to define the interconnection of internal logic elements, thereby materializing the hardware architecture needed to execute a specific algorithm. Figure 1 illustrates the process of converting a Register Transfer Level (RTL) hardware description—or a C/C++ implementation using High-Level Synthesis (HLS)—into a list of logic gates, known as a netlist, and

*e-mail: quentin.berthet@cern.ch

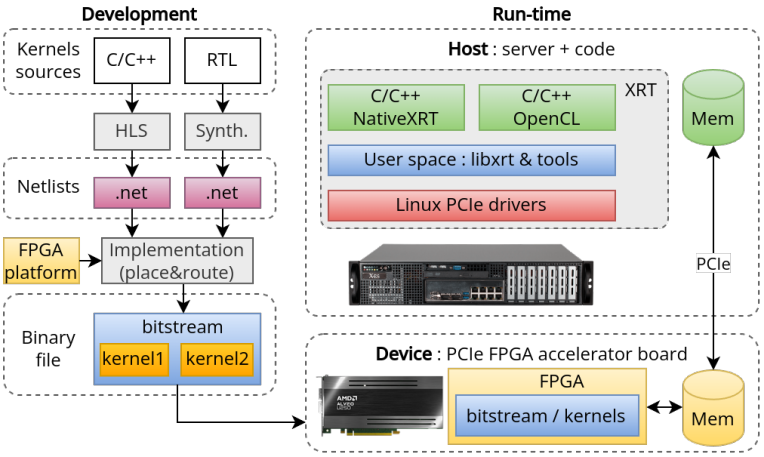



**Figure 1.** FPGA development flow and communication with software accelerators.

subsequently into a device-specific bitstream. This bitstream can contain multiple accelerated algorithms, often referred to as kernels. Loading this bitstream takes a non-negligible amount of time, ranging from tens of milliseconds to several seconds. As a result, dynamic switching of bitstreams (or algorithms) is not desirable in latency-constrained systems such as the ATLAS trigger. Furthermore, the loaded bitstream becomes a shared state, making the management of FPGA configurations across Athena algorithms challenging, particularly when handling multiple FPGA devices or algorithms simultaneously. This challenge is compounded by the lack of centralized bitstream management within the Athena framework, placing the responsibility on algorithm developers to use low-level XRT or OpenCL APIs to perform these actions themselves. AthXRT [5] addresses these challenges by providing a centralized service that manages AMD FPGA accelerators configuration files during system initialization, allowing multiple algorithms to utilize FPGAs without runtime overhead.

## 2 AthXRT presentation

As currently integrated into Athena, AthXRT is designed to avoid boilerplate code and make the FPGA loading process more robust and error-resistant in the least intrusive and constraining way possible, simplifying the work of algorithm developers. As illustrated in figure 2, it serves as an intermediate layer between Athena user algorithms and the Xilinx Runtime (XRT) library [3] provided by AMD (formerly Xilinx) for programming and communicating with the Alveo line of PCIe-attached FPGA accelerators. AthXRT, implemented as an Athena service, primarily manages the configuration and loading of bitstreams (referred to as XCLBIN files in AMD nomenclature) onto the accelerators. By centralizing this process, AthXRT reduces the complexity of managing multiple FPGA devices and provides a unified configuration interface seamlessly integrated into existing Athena practices. This centralization facilitates the early detection and resolution of bitstream conflicts that may arise when multiple algorithms require incompatible XCLBIN files and compete for FPGA resources.

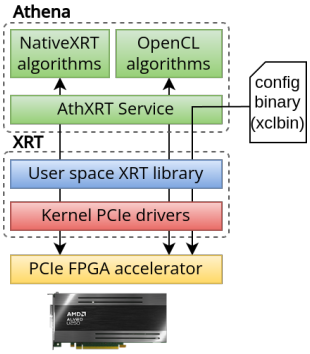



**Figure 2.** Position of AthXRT in the software stack.

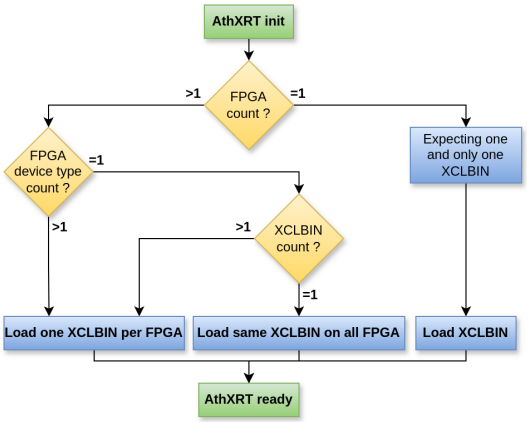



**Figure 3.** AthXRT FPGA configuration file loading logic.

### 2.1 Device enumeration and configuration

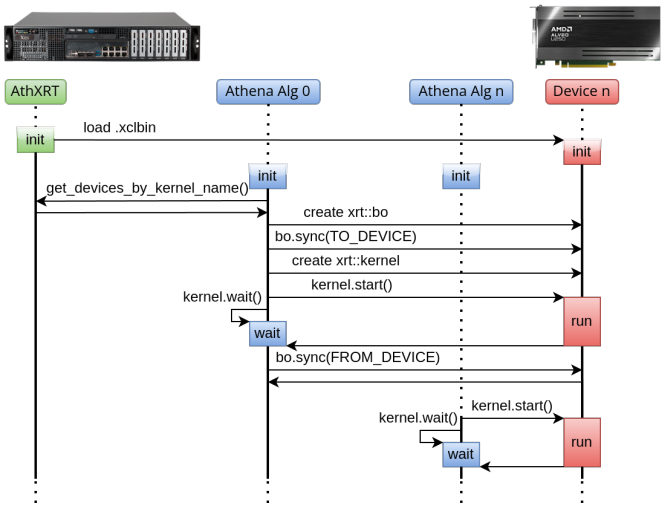
AthXRT abstracts both device enumeration and the configuration loading interface of the XRT library. Using an Athena Python configuration mechanism, user algorithms can specify one or more XCLBIN files to be loaded in preparation for a run. During the initialization of AthXRT services, the list of requested XCLBIN files is collected and matched against the list of accelerator devices identified during enumeration. A well-defined logic, illustrated in Figure 3, determines how the service assigns devices to XCLBIN files, supporting the following configurations:

- Single FPGA, single XCLBIN file.
- Multiple similar FPGAs, all loaded with the same XCLBIN file.
- Multiple similar FPGAs, each loaded with a unique XCLBIN file.
- Multiple different FPGAs, each loaded with a unique XCLBIN file.

In the event of an error during device configuration, such as when the number of XCLBIN files exceeds the available devices, the service terminates the run with a clear error message. Upon successfully loading all XCLBIN files, the acceleration kernels in the system are inspected and stored alongside their respective device handles, allowing for future retrieval of the device handle associated with a specific kernel. Once all devices are configured, the service completes its initialization.

### 2.2 User algorithms and API support

During initialization, user algorithms can query the AthXRT service to retrieve a list of device handles associated with a specific kernel. While no strict naming convention is enforced for kernels, it is expected that their names are unique by function and version. Kernels with the same name, even on different devices, should be functionally interchangeable. User algorithms are free to select one or multiple kernel instances from this list and use them as needed. A simplified example of the initialization and usage of the service by two user algorithms is illustrated in Figure 2.2. It is important to note that the underlying XRT library manages




concurrent access to the same kernel, ensuring that multiple threads—or even multiple algorithms—can access the same kernel without issues. Kernel executions are obviously still exclusive and are serialized over time. In the context of multi-threaded Athena (AthenaMT [4]), this provides a significant advantage: by allowing multiple threads of the same algorithm to share the same kernel, data transfers to and from the device can be overlapped with kernel executions. This efficient resource sharing not only hides transfer latency but also increases overall computational throughput.

AthXRT supports both native XRT and OpenCL APIs provided by the underlying library, giving developers the flexibility to choose the API best suited to their use case. Internally, the service uses the OpenCL API to program the boards and retrieve the lower-level XRT API handles afterward. Once the service is initialized and the FPGA devices are programmed, user algorithms can interact with the devices directly and transparently. They can create buffers and interact with specific kernels without being constrained by API choice or additional run-time abstractions. This approach eliminates the need for dynamic configuration loading, which is time-intensive and prone to introducing delays during critical execution stages. More importantly, it prevents multiple successive programming operations on devices—a potential issue if FPGA loading were handled by user algorithm initialization code, where two algorithms might compete for the same device. Furthermore, AthXRT is designed to be as unobtrusive as possible for user code. Once the devices are programmed, the service imposes no constraints on API usage or run-time interactions, allowing developers to work within their preferred paradigms.

### 2.3 Example code and testing

AthXRT is integrated into the main Athena branch and is accompanied by example code provided in the AthExXRT package [6]. This package includes vector addition and vector multiplication example algorithms that utilize the AthXRT service with both XRT and OpenCL APIs. These examples demonstrate how user algorithms can implement communication with both AthXRT and the accelerator device, as well as provide kernel code that can be compiled into effective hardware kernels within an XCLBIN. The examples have been extensively used to test configurations involving multiple devices, multiple kernels, and multiple APIs operating simultaneously. Tests were conducted at the University of Geneva on a pair of Alveo

VCK5000 accelerators, as well as on an ATLAS testbed equipped with an Alveo U250 and an Alveo U50.

## 3 Future Work

While the current implementation of AthXRT focuses on managing the configuration of AMD FPGAs, future work may extend its support to additional hardware platforms and APIs. This would enable broader compatibility with diverse accelerator ecosystems, including other FPGA vendors and potentially non-FPGA hardware accelerators, fostering greater flexibility and scalability for heterogeneous computing environments.

There is also significant potential for AthXRT to evolve into a comprehensive FPGA configuration management system. Such a system could streamline the entire lifecycle of XCLBIN files, including their building, storage, cataloging, and versioning. By centralizing these processes, AthXRT could significantly reduce the overhead associated with managing FPGA-accelerated workflows, enabling developers to focus more on algorithm design and optimization. Additionally, incorporating features such as automated dependency tracking, compatibility validation, and rollback mechanisms would further enhance the reliability and maintainability of FPGA-accelerated systems. These advancements would not only simplify the development and maintenance of FPGA-accelerated algorithms but also establish AthXRT as a robust tool for managing heterogeneous computing resources in high-performance environments like Athena.

## 4 Conclusion

AthXRT provides a centralized and efficient solution for managing FPGA configurations within the Athena framework. By offloading the configuration of FPGAs to a centralized service during initialization, AthXRT eliminates runtime overhead and simplifies the use of FPGAs in high-performance applications. With its support for both native XRT and OpenCL APIs, AthXRT offers flexibility and scalability, making it a valuable tool for heterogeneous computing in the ATLAS experiment. The integration of AthXRT into the Athena framework, along with its extensibility for future use cases, positions it as a key component for enabling efficient use of hardware accelerators in scientific computing.

## References


[1] ATLAS Collaboration. (2024). Athena: Software framework for ATLAS experiment. Zenodo. https://doi.org/10.5281/zenodo.2641996

[2] ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider." JINST 3 (2008) S08003. https://doi.org/10.1088/1748-0221/3/08/S08003

[3] AMD. (2024). Xilinx Runtime (XRT). GitHub repository. https://github.com/Xilinx/XRT

[4] Charles Leggett et al. (2017). AthenaMT: upgrading the ATLAS software framework for the many-core world with multi-threading. Journal of Physics: Conference Series. https://dx.doi.org/10.1088/1742-6596/898/4/042009

[5] ATLAS Collaboration. AthXRT code source in ATHENA Gitlab repository. https://gitlab.cern.ch/atlas/athena/-/tree/main/Control/AthXRT/AthXRTServices

[6] ATLAS Collaboration. AthXRT examples in ATHENA Gitlab repository. https://gitlab.cern.ch/atlas/athena/-/tree/main/Control/AthenaExamples/AthExXRT