From software to hardware: An easy guide to accelerate algorithms for the HL-LHC upgrades of the CMS Level-1 trigger system

Pelayo Leguina[a,*] **and Santiago Folgueras**[a]

[a] *ICTEA,*
University of Oviedo, C. San Francisco, Oviedo, Spain

E-mail: leguinapelayo@uniovi.es, folguerassantiago@uniovi.es

At the LHC, the vast amount of data from the experiments demands both sophisticated algorithms and substantial computational power for efficient processing. Hardware acceleration is an essential advancement for HEP data processing, focusing specifically on the application of High-Level Synthesis (HLS) to bridge the gap between complex software algorithms and their hardware implementation. We will explore how HLS facilitates the direct implementation of software algorithms into hardware platforms such as FPGAs to enable real-time data analysis. We will use the case study of a track-finding algorithm for muon reconstruction with the CMS experiment, demonstrating the role of HLS in translating algorithms into high-speed, low-latency hardware solutions that improve the accuracy and speed of particle detection. Key techniques in HLS, including parallel processing, pipelining, and memory optimization, will be discussed, illustrating how they contribute to the efficient acceleration of algorithms.

42nd International Conference on High Energy Physics (ICHEP)
17-24 July 2024
Prague, Czech Republic

[*]Speaker

1. Introduction

The CMS experiment [1, 2] has been a pivotal system for high-energy physics research. As part of ongoing developments, the CMS Level-1 Trigger System is undergoing a significant upgrade [3], which demands highly efficient algorithms to process vast data streams in real-time. Vitis High-Level Synthesis (HLS) [4] enables the use of high-level programming languages, such as C++, to design algorithms that are later synthesized into FPGA hardware. This process simplifies the development of efficient hardware, allowing faster iterations than a traditional RTL-based design.

2. Designing an Algorithm with HLS

The design process is summarized in Fig. 1, showing the transition from software design to hardware implementation, in three steps:

- **Software Design**: This initial stage involves gathering ideas about the algorithm and developing a software model to verify its functionality through simulation.
- **HLS Refactoring**: In this block, the software code is refactored into HLS-synthesizable code. Design iterations are applied, testing different architectures and optimization techniques like pipelining and memory reordering to ensure efficient hardware acceleration.
- **Hardware Implementation**: The final step involves synthesizing the HLS modules and implementing them on FPGA hardware using Vivado. This ensures that the design is optimized for timing, resource usage, and overall system performance.

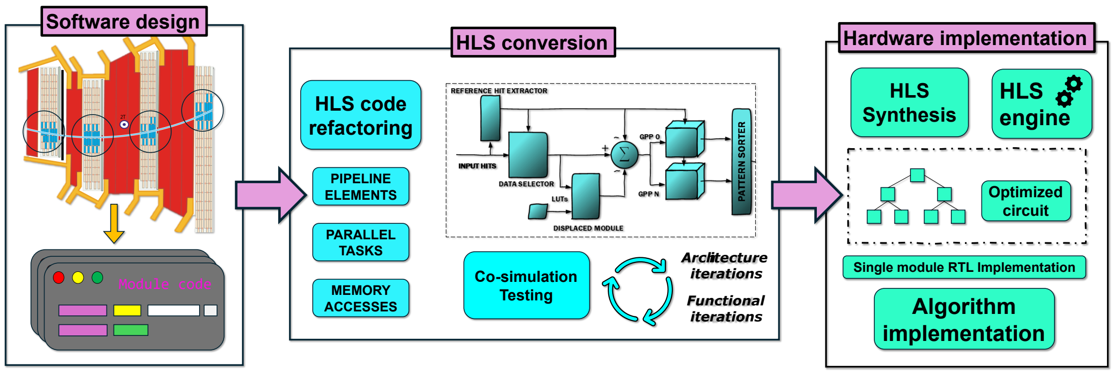

Figure 1: Main steps in the algorithm design: software design, HLS refactoring, and hardware implementation.

3. HLS Refactoring Techniques

After designing the software modules, they must be refactored for synthesis using HLS [5]. The primary goal is to transform the code into a hardware-friendly version that optimizes resource utilization while maintaining high performance.

Two refactoring techniques are typically employed: **Loop Pipelining**, which minimizes initiation intervals (II) and enables parallel execution of operations; and **Memory Optimization**, where HLS provides options for reordering and combining memory for more efficient FPGA resource use.

4. Hardware Implementation with Vivado

After refactoring the algorithm using HLS, it is synthesized into RTL code. Vivado is then used to implement the design on an FPGA, optimizing speed, resource usage, and clock timing.

Vivado supports:

- **Clock Optimization**: Ensuring data is processed in real-time by adjusting clock cycles.
- **Resource Management**: Optimizing FPGA resources such as LUTs and BRAMs during synthesis.

Using Vivado, the design is implemented to meet the high-speed processing requirements of the CMS Level-1 trigger system.

5. Integration Design Pipeline

After the hardware modules are synthesized, they are integrated into the overall FPGA framework. This involves verifying timing, performing system-level tests, and loading the modules onto the FPGA board.

6. Conclusion

High-Level Synthesis significantly simplifies the development of hardware for the HL-LHC CMS Level-1 trigger system by reducing development time and optimizing FPGA resources. Through techniques such as loop pipelining, memory optimization, and parallel processing, HLS enables the acceleration of algorithms for real-time data processing. The integration of these techniques into the CMS Level-1 Trigger System ensures a fast production cycle for the required algorithm implementations, contributing to the success of the HL-LHC upgrades.

References

[1] CMS Collaboration, “The CMS experiment at the CERN LHC,” JINST 3 (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.

[2] CMS Collaboration, “Development of the CMS detector for the CERN LHC Run 3,” JINST 19 (2024) P05064, doi:10.1088/1748-0221/19/05/P05064.

[3] CMS Collaboration, “The Phase-2 Upgrade of the CMS Level-1 Trigger,” CERN, Geneva, CERN-LHCC-2020-004, CMS-TDR-021, 2020.

[4] Xilinx, "Vivado Design Suite User Guide: High-Level Synthesis," 2021.

[5] P. Coussy and A. Morawiec, "High-Level Synthesis: From Algorithm to Digital Circuit," *Springer*, 2008.