Journal of Instrumentation

SystemC framework for architecture modelling of electronic systems in future particle detectors

Published by IOP Publishing for Sissa Medialab

Received: *November 3, 2023*
Accepted: *December 17, 2023*
Published: *January 19, 2024*

Topical Workshop on Electronics for Particle Physics
Geremeas, Sardinia, Italy
1–6 October 2023

SystemC framework for architecture modelling of electronic systems in future particle detectors

Francesco Enrico Brambilla, Davide Ceresa,* Jashandeep Dhaliwal, Stefano Esposito, Kostas Kloukinas, Xavier Llopart Cudie and Adithya Pulli

CERN,
Esplanade des Particules 1, 1211 Geneva, Switzerland

E-mail: davide.ceresa@cern.ch

Abstract: The prototyping cost in advanced technology nodes and the complexity of future detectors require the adoption of a system design approach common in the industry: design space exploration through high-level architectural studies to achieve clear and optimized specifications. This contribution proposes a configurable SystemC framework to simulate the readout chain from the front-end chips to the detector back-end. The model is transaction accurate, includes an event generator, interfaces with real physics events, and provides metrics such as readout efficiency, latency, and average queue occupancy. This contribution details the structure of the framework and describes a case study based on the LHCb VeLo upgrade II.

Keywords: Simulation methods and programs; Software architectures (event data models, frameworks and databases); Data acquisition concepts; Digital electronic circuits

*Corresponding author.

https://doi.org/10.1088/1748-0221/19/01/C01039

2024 JINST 19 C01039

Contents

1 Introduction

The CERN EP department has launched a strategic R&D programme [1] on technologies for future experiments. In this context, the IC technology work package, also called WP5, is developing a simulation framework, called PixESL, for the architectural modelling of future particle detectors. This work focuses on modelling pixel-based detectors, from front-end to back-end, at a high level of abstraction to perform architectural studies. The objective is to study the readout network efficiency and provide the metrics to compare different solutions to satisfy functional and non-functional requirements, such as data throughput, data quality, downlink material cost, and other parameters detector-wise and at the ASIC level. At the same time, it can provide a reference model for system design, which includes several ASICs and some components of the back-end system.

The proposed framework adopts ESL, or Electronic System-Level design, a methodology for modelling and simulating complex electronic systems. It focuses on designing at a higher level of abstraction than traditional hardware description languages (HDLs) like VHDL or Verilog. ESL design is particularly useful for designing System-on-Chip (SoC), complex integrated circuits, and embedded systems. The key aspects of the PixESL framework are:

- Abstraction: the design starts at a higher level of abstraction, capturing the system's functionality and behavior rather than low-level hardware details. This allows designers to work with functional blocks, interfaces, and data flow before diving into specific hardware components.
- System-Level Perspective: the design takes a holistic view of the entire system, including hardware and software components. It enables the design and verification of the entire system, not just individual hardware modules.
- Modelling: designers create high-level models and abstractions of the system's behavior. The models are based on SystemC, a system-level description language.
- Simulation and Analysis: the readout chips' models are used for simulation and analysis to verify system functionality, performance, and other characteristics. Designers can explore different design options and configurations at this stage.
- Performance Estimation: the framework allows for early system performance estimation, such as power consumption, execution time, and resource utilization. This helps in making design decisions that meet performance goals.

2024 JINST 19 C01039

- Design Space Exploration: designers can explore different architectural choices, configurations, and trade-offs to optimize the system's performance and cost. This exploration is critical for achieving design objectives efficiently.

- Validation: the design helps validate the system's requirements early, ensuring that the final product will meet its intended purpose. Identifying and addressing issues before they become costly can save time and resources.

In summary, the PixESL framework proposes a system-level approach to the electronic system design of particle detectors that focuses on abstraction, modelling, simulation, and the concurrent development of hardware and software. It facilitates a top-down design approach and helps designers create efficient, well-optimized electronic systems by providing a high-level perspective and enabling early validation of design choices. This paper presents a preliminary implementation of the framework and a proof-of-concept study based on the LHCb VeLo upgrade II.

2 Framework description

The framework is composed by three main modules, a SystemC modelling environment, an event generator, and a Python analysis toolset. They are used together to simulate a given architecture and evaluate its performance.

SystemC modelling environment. SystemC [2] is a C++ library that provides a discrete-event simulation kernel along with C++ classes to simulate concurrent processes. It provides a way to model the behavior of a system at various abstraction levels and allows encapsulations of components as modules, which can be reused in different parts of the framework. This modularity promotes code reusability and reduces development time. SystemC's abstraction allows for a functional description of modules rather than a description of the connections between registers and logic gates, to study the system without the burden of a hardware description language at the Register Transfer Level (RTL) abstraction.

The readout architecture of pixel detectors is a regular network of nodes whose goal is to concentrate digitized data from a large number of sources, the pixels, into a limited amount of output channels. This readout network can be modeled as a tree-like hierarchy of nodes, based on the order followed by packets going from leaves, the pixels, to the root, the system output. The nodes belonging to the same hierarchical level called *layer*, are identical and include:

- *communication* methods to exchange packets with nearby nodes;

- *arbitration* and *routing* functions to decide from where to receive data and where to send it;

- *memory elements* to act as temporary storage.

A readout architecture model is created by describing a network of these nodes that defines the connections between elements in the same layer and between parent and children layers. Currently, this model is implemented in SystemC with TLM-2.0 [3] non-blocking transport and follows a pull-based communication semantics with clock-based timing. Each layer's function or element describes processes sensitive to a clock signal representing the node's functions. Whenever a node has

2024 JINST 19 C01039

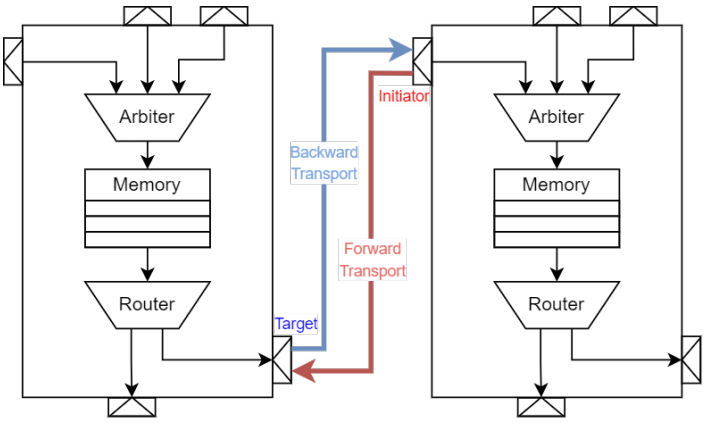

Figure 1. Block diagram of two nodes including the associated transport functions.

storage available, it issues pull requests through a *forward transport* call to the other connected nodes on a clock event. Upon receiving a request, each node waits for a simulation delta cycle to accumulate other possible requests. Then, the routing function selects the request with the higher priority and sends data to that node, ending the transaction with a *backward transport* call.

To model this system, a description of a generic layer with *communication, memory, arbitration,* and *routing* functions has been developed and then specialized based on the specific layers. For example, the generic layer is designed to communicate with other modules using a pull-based protocol, but in the case of the pixel-superpixel links, a special protocol was added because of the asynchronous nature of pixels in the model.

Input stimuli and event generator. To simulate the detector model, input stimuli can be provided externally as a list of pixel hits, or can be generated internally by a configurable event generator that provides several classes of cluster types and spatial distributions.

The input stimuli are comma-separated value files containing a list of pixel hits, each carrying at least the pixel coordinate, the Time-of-Arrival (ToA), and the Time-over-Threshold (ToT) information. Otherwise, a built-in event generator is bundled with the model: it provides different event types, enabling the user to configure the rate and patterns of possible particle hits on the detector. These types include single-pixel, diamond-shaped, or straight clusters, which can be uniformly distributed over the detector or follow Gaussian or exponential spatial distributions.

Python-based results and metrics analyzer. The framework includes a Python3-based results and metrics analyzer to help process and interpret the output data generated by visualizing the results, assessing the performance, automatizing the process of collecting and analyzing metrics, and adding alerting and reporting features.

Python, among the open-source languages, is chosen because of its ease of learning and use, its rich ecosystem, and community support. The results and metrics analyzer should also be accessible to framework users who might not have any expertise in SystemC. Since the model aims to evaluate readout architectures and configurations, data and statistics are logged in each SystemC node and then processed and analyzed by the Python analyzer.

These results can be scalar values, such as the readout efficiency or the average packet latency, which allow a quick comparison of different systems or plots and maps, such as memory occupancy or data loss, to spot architecture limitations such as bandwidth bottlenecks.

2024 JINST 19 C01039

3 LHCb VeLo upgrade II study

The first case study chosen as a proof-of-concept of the framework is the LHCb VeLo upgrade II [4]. The procedure followed is:

1. Model the current LHCb VeLo pixel Read-Out Chip (ROC), namely Velopix [5];
2. Simulate the ROC with physics event from the LHCb VeLo upgrade II;
3. Find limitations and bottlenecks of the current architecture;
4. Propose an optimized architecture for the upgrade.

The Velopix exploits a data-driven readout architecture based on four layers: *pixels*, *super-pixels (SP)*, *regions* and *end-of-column (EoC) nodes*, as shown in figure 2. All layers present arbiters, First-In-First-Out (FIFO) memories, and routers. The readout architecture uses a back-pressure mechanism that limits data losses only to the *super-pixels* layer. This layer includes two memory elements: the internal FIFO for the packets incoming from the pixels and the external FIFO for the packets incoming from the other super-pixels. Table 1 shows the architectures and the results of a parametric study that started from VeloPix to reach a new proposal in order to increase the performance.

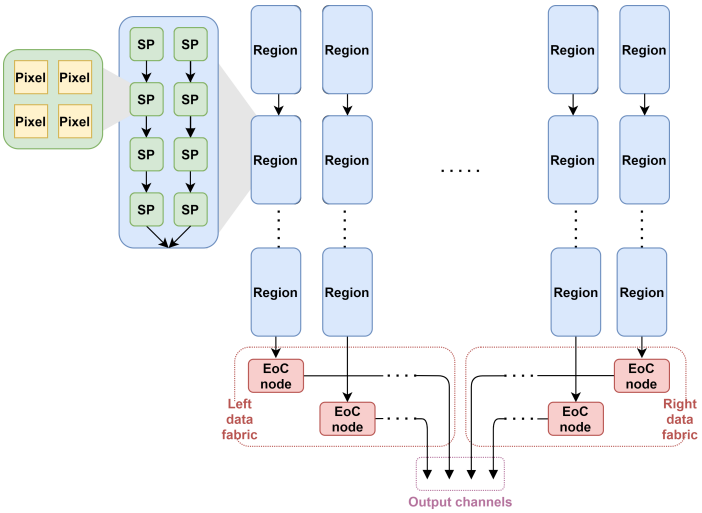

Figure 2. Schematic of LHCb VeLo architecture.

Table 1. Configurations and results of VeloPix and proposed architectures.

		VeloPix	Proposed
Layers size	Pixel	256x256	
	Super-Pixel	128×128	
	Region	64×16	128×8
	EoC node	64	128
	Output ch.	8	16
FIFOs depth	SP FIFO int.	4	
	SP FIFO ext.	2	
	Region FIFO	4	
	EoC node FIFO	2	
Clock	Matrix clock	×1	
	EoC node clock	×8	
Results	Readout eff.	86%	100%
	Avg. latency	114 cy.	17 cy.

*The clock is referred to the Bunch-Crossing rate.

An external semiconductor detector Monte Carlo simulation framework [6] provides the input stimuli for one thousand bunch collisions. Figure 3 shows the injected pixel hits map characterized by an extremely high input rate on the top side of the pixel matrix. The main metrics analyzed are readout efficiency, expressed as the percentage of packets correctly read out of the chip, and latency, expressed as the number of cycles from the injection of a pixel hit to the output of its corresponding data packet.

The VeloPix architecture shows an average latency of 114 cycles and a readout efficiency of 86% caused by several lost packets on the top left part of the pixel array, as shown in figure 4. Using the cycle-based memory occupancy per node from the Python analyzer, it is clear that the *regions* are not

2024 JINST 19 C01039

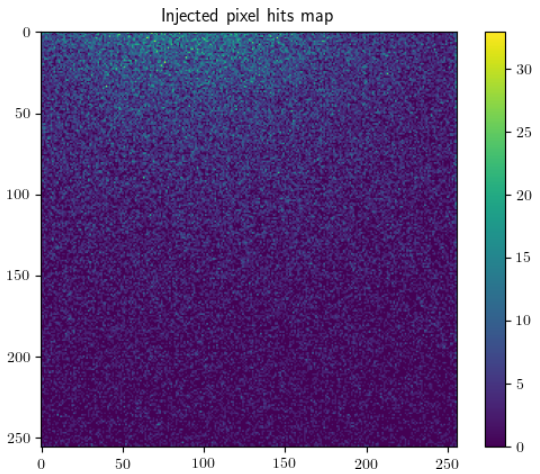

Figure 3. Map of the injected pixel hits.

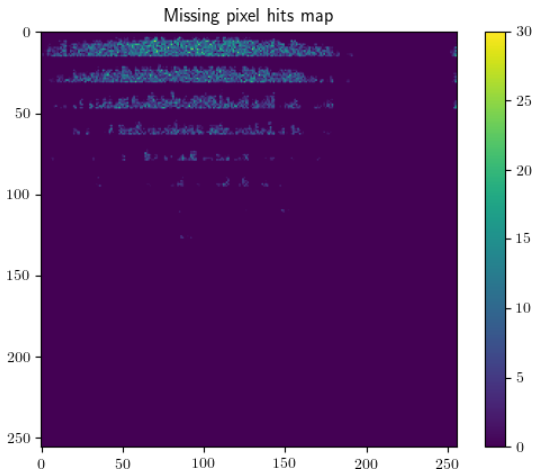

Figure 4. Map of the lost pixel hits for VeloPix.

able to sustain such a high input rate and present drastic congestion of data packets. To counteract this issue, the proposed architecture doubles the *region* columns, *EoC nodes*, and output channels as shown in table 1: this doubles the maximum throughput of the system from 64 to 128 packets/cycle. On the other hand, to mitigate the hardware overhead, the number of *regions* per column has been halved. The proposed architecture presents a 100% readout efficiency, meaning no event is lost, and an average latency of 17 cycles, seven times lower than the Velopix architecture.

4 Conclusions and future outlook

The PixESL framework proposes a rapid and efficient prototyping approach for system-level development using well-established open-source languages like SystemC and Python. The LHCb VeLo upgrade II study confirms the usefulness of this framework for design space exploration and architectural analysis, thanks to its high-level modelling and fast simulation capabilities. The release of the framework is currently foreseen at the end of 2024. The development plan aims to expand the modelling library, improve simulation speed, configurability, reusability, and include a performance estimation tool.

References

[1] M. Aleksa et al., *Strategic R&D Programme on Technologies for Future Experiments*, CERN-OPEN-2018-006 (2018) [DOI:10.17181/CERN.5PQI.KDL2].

[2] *IEEE Standard for Standard SystemC® Language Reference Manual*, IEEE 1666-2023.

[3] *OSCI TLM-2.0 language reference manual*, https://www.accellera.org/.

[4] T. Pajero, *VELO Upgrade II: The LHCb 4D pixel detector*, LHCb-TALK-2022-310 (2022).

[5] T. Poikela et al., *VeloPix: the pixel ASIC for the LHCb upgrade*, 2015 *JINST* **10** C01057.

[6] S. Spannagel et al., *Allpix^2: A Modular Simulation Framework for Silicon Detectors*, *Nucl. Instrum. Meth. A* **901** (2018) 164 [arXiv:1806.05813].

2024 JINST 19 C01039