Virtual Prototyping Framework for Pixel Detector Electronics in High Energy Physics

1st Francesco E. Brambilla
EP-ESE-ME
CERN - KU Leuven
Geneva, Switzerland
francesco.enrico.brambilla@cern.ch

2nd Davide Ceresa
EP-ESE-ME
CERN
Geneva, Switzerland

3rd Jashandeep Dhaliwal
EP-ESE-ME
CERN
Geneva, Switzerland

4th Stefano Esposito
EP-ESE-ME
CERN
Geneva, Switzerland

5th Kostas Kloukinas
EP-ESE-ME
CERN
Geneva, Switzerland

6th Jeffrey Prinzie
ESAT
KU Leuven
Leuven, Belgium

***Abstract*—This contribution outlines the development of PixESL, a virtual prototyping framework for pixel detectors based on C++/SystemC. It offers a platform for describing detector ASICs designed for High Energy Physics experiments at a high level of abstraction, with the capability to simulate the entire process from particle interaction in the sensor to the readout of a digital data packet to the experiment back-end. The framework supports modelling analog/digital front-end, readout networks, data processing and formatting. This paper details the implementation of the PixESL framework and its use in a pixel detector being developed for the LHCb experiment at CERN. The framework serves as an effective and rapid prototyping method, as well as a reference model for verification.**

***Index Terms*—Virtual Prototyping, SystemC, Pixel Detectors, Verification Reference Model, ASIC Modelling**

I. Introduction

Particle detectors for High Energy Physics (HEP) experiments exploit large and complex electronic systems to read out high-resolution pixelated detectors, composed of more than 50k pixels. As highlighted in [1], the next generation of hybrid pixel detector ASICs will face challenges in terms of power consumption and bandwidth due to the high-rate environment (up to 3–4 $\mathrm{GHz\,cm^{-2}}$) and measurement resolution, especially for applications with fast timing resolution below 50 ps.

Considering the typical power budget of a few $\mathrm{W\,cm^{-2}}$, an early optimization of the chip and system architecture is of utmost importance to fit the specifications of the experiment. This work proposes a novel approach in the High Energy Physics field, a high-level simulation platform to study the entire electronics system chain at an early stage of development.

The main components currently modelled in the PixESL framework are:

- *Pixel Analog Front-End (AFE)*: serves as the interface between the detector sensor and the digital processing system. It amplifies, shapes, and digitizes the analog signals generated by particle interactions with the sensor.
- *Pixel Digital front-end*: provides Time-to-Digital Converter (TDC) functionalities, converting analog signals from the AFE into digital packets.
- *Data Readout Networks*: handles the transfer of digital data from the pixel, through the pixel array to the chip's periphery and off-chip.
- *Data processors and formatter*: models digital logic blocks performing operations like hits clustering, time-walk correction, event sorting, and output frame formatting.

The proposed simulation platform enables comprehensive analysis and optimization of each of these components and their interactions within the overall electronics system. By simulating the entire chain early in the project life-cycle, designers can identify potential bottlenecks, evaluate design trade-offs, and refine specifications to meet the experiment's requirements within the given power budget and resource constraints. Compared to a project-dedicated virtual prototype, the PixESL framework proposes a standardized approach based on:

- *SystemC-based model* for behavioural description and fast simulation.
- *Python-based analysis tool* to extract relevant metrics from the models and evaluate performance.
- *Inputs from physics* supporting the injection of stimuli in the Analog Front-End from physical simulation of the particle interaction in the sensors.
- *Integration in the verification environment* through a verification IP to reuse the virtual prototype as the reference model.
- *Architecture configurability* through external files for faster design iteration and easier user access.
- *Open-source languages and code* for maximum redistribution.

The paper is organized as follows:

- Section II places the framework in the detector design flow and lists its features.

- Section III describes the modelling approach used for the digital and Analog and Mixed-Signals (AMS) circuits usually found in pixel detectors.
- Section IV focuses on the model's use during the early stage of development for design space exploration.
- Section V shows the integration of a reference model within a Universal Verification Methodology (UVM) verification environment.

II. Framework Motivation and Objectives

Pixel ReadOut Chips (ROC) are mixed-signal ASIC designs featuring high-performance analog front-ends, continuous and low-latency data readout, on-chip data processing, and high-speed output links. These heterogeneous functionalities exploit different design methodologies and tools, complicating the design process and requiring several iterations to validate specifications.

To facilitate the development process, the PixESL framework proposes adding an Electronic System-Level (ESL) design phase with a holistic system view through a higher level of abstraction. This phase is located between the specification draft and hardware design, as shown in Fig. 1, providing a C++ virtual prototype. This prototype helps the design team better match the system architecture to the physics experiment goals and understand critical issues in the ROC early in the design phase, such as required output bandwidth or possible data loss sources. Later in the development, the high-level model can be used as a guide for hardware design to obtain the required level of performance and act as a reference model during the design verification phase.

An additional goal of this work is to develop a common system design approach for the HEP community, building the framework on a hierarchical, configurable, and reusable structure based on open-source languages. For this reason, the PixESL framework features:

- *Components library*: A SystemC components library containing Front-End, readout, and data processing block models.
- *Network generation*: Network description is defined in human-readable configuration files instructing the generation of a network using components from library.
- *Data logging and metrics extraction*: integrated logging and analysis allow fast architecture evaluation through a Python tool-set.

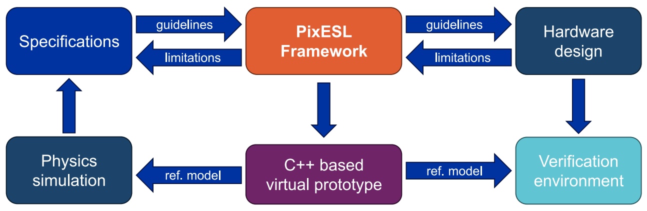

Fig. 1. Schematic view of the design flow with PixESL.

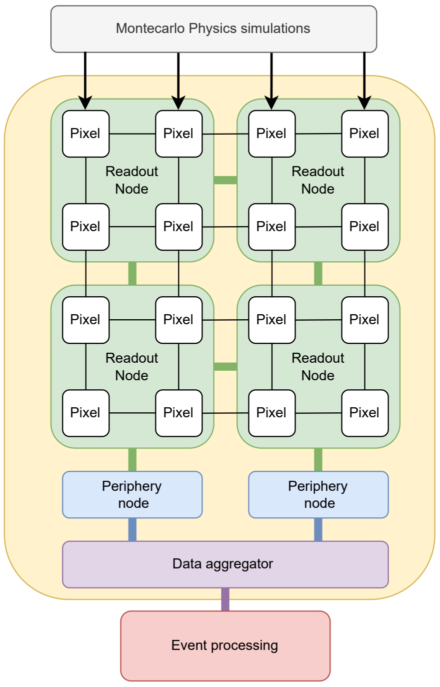

Fig. 2. Schematic view of the data-flow in a ROC model.

- *UVM integration*: the framework is shipped with an add-on to instantiate PixESL components in a SystemVerilog-UVM simulation.

The framework's objective is to provide a C++ virtual prototype for design space exploration and a reference model for verification as detailed respectively in Section IV and V.

III. Modelling Methodology

The ESL design approach prescribes a higher level of abstraction in the system description, enabling prototyping from a system-level perspective and facilitating faster design and simulation speed. SystemC was chosen as the modelling language because it provides a simulation kernel for concurrent processes and specialized classes to represent hardware blocks and communication protocols.

This approach guarantees a much higher simulation speed than RTL descriptions when the correct modelling methodology is applied, reducing the simulation runtime by around 50 times. However, due to its abstraction from the actual design's implementation and technology, this methodology provides only the number of transactions and nodes in the system, enabling architecture-level comparisons. Technological information is required to link these figures to actual power and area cost estimation.

This section highlights the different modelling techniques employed to represent various functions and modules in a ROC. As shown in Fig. 2, the ROC model is hierarchically structured, starting from the pixels, where sensor hits from physics simulation are injected into and digital packets are generated, to the readout network and periphery, which move these packets towards the system's output channels, aggregating and possibly processing them as they move through the data path. Analog, mixed-signal, and digital design encompass different design methodologies and tools, each reflected through distinct modelling approaches. These circuits and their modelling techniques are described in the following paragraphs:

A. *Analog pixel front-end*

The Analog Front-End (AFE) is a full-custom design converting the discharge signal from the pixel's sensor into a digital pulse, with a duration proportional to the input charge. These input charges, along with their position on the detector and timestamp, are extracted from physical simulation of the sensor, obtained with tools such as Allpix Squared [4], which couple the SystemC detector model with the High-Energy Physics simulation.

The AFE input corresponds to a bump pad connected to the sensor where the particle interaction discharge signal is injected. This signal is amplified by a charge-sensitive preamplifier and filtered through a band-pass filter. Then, a threshold discriminator detects the sensor hit and shapes the digital pulse, whose rise time corresponds to the sensor hit Time-of-Arrival (ToA) and the pulse length (Time-over-Threshold, ToT) is proportional to the sensor charge and particle energy [6]. This threshold is set as low as possible to maximize the detection efficiency while limiting the electronic noise.

The transfer function of the analog front-end, from the input charge to the output digital pulse, is extracted from analog simulation and modelled in a pure C++ class with point interpolation and linear fitting. Since the characteristics of the AFE are defined by the analog implementation and their study is out of the scope of this work, the transfer function model is a simpler and faster alternative to a SystemC AMS description of the individual AFE blocks.

Fig. 3 shows a plot from the C++ model of the threshold discriminator's pulse rise time (T_r) and fall time (T_f) as a function of the threshold, obtained in the model by injecting a fixed charge ($Q_{in} = 5\,\mathrm{ke^-}$) at time zero. The transition between point interpolation and linear fit modeling causes the discontinuity observable around 7000 ps. This curve also corresponds to the shape of the filtered signal fed to the discriminator. The pulse duration can be obtained as the time difference between T_f and T_r for a fixed threshold value. The Analog FE C++ class analytically computes and returns the rise and fall time of the discriminator output based on the input charge and its ToA.

As shown in the schematic view in Fig. 4, the Analog FE C++ class is wrapped in a generic `sc_module` derived class. The AFE wrapper's task is to enable the analytic AFE model to run inside the event-based simulation. During simulation time, it injects sensor hits in the analytical C++ model and sets an `sc_signal` corresponding to the pulse using the values for T_r and T_f returned from the AFE.

By integrating this model in the system-level simulation and running it with real physics data, designers can evaluate the

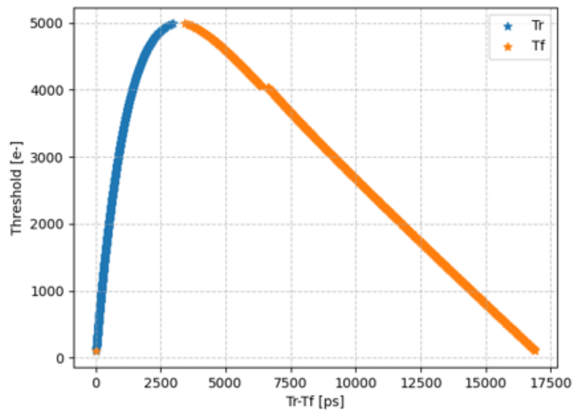

Fig. 3. Threshold scan of the Analog Front-End.

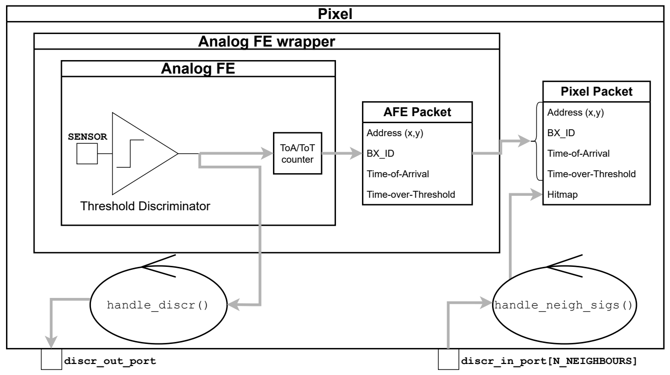

Fig. 4. Pixel and Analog Front-End schematic view as modeled.

impact of the analog front-end characteristics, such as peaking time and return-to-zero slope, on system efficiency.

B. *Digital pixel front-end*

The primary function of the pixel front-end digital part is to process the AFE discriminator output, generating a data packet that contains useful information such as location, timing, and energy. It employs a Time-to-Digital Converter (TDC) to measure the pulse rise time (ToA) and duration (ToT), which are proportional to the particle's interaction time and energy.

Additionally, neighboring pixels can be interconnected at the level of the digital pixel front-end, forming a local network. This network enables local data processing that can be performed before or after the TDC stage, such as the computation of a local hitmap representing the status of the nearby pixels. Once the conversion and the data processing are finished, the pixel injects this packet into the readout data path, adding the pixel location.

From a modeling perspective, the digital pixel front-end model includes the AFE wrapper models as shown in Fig. 4. Two `sc_method` processes, triggered by the discriminator outputs of the internal AFE and the local pixel network, model the design functionality behaviorally. The internal discriminator triggers the `handle_discr()` process to detect incoming particles, emulating the TDC by measuring ToA and ToT

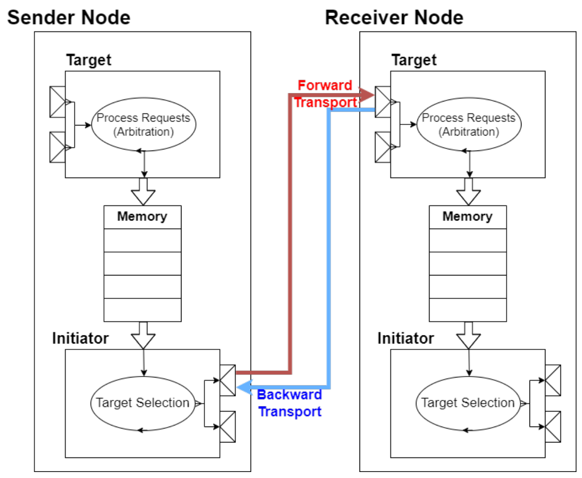

Fig. 5. Schematic view of communication between base nodes.

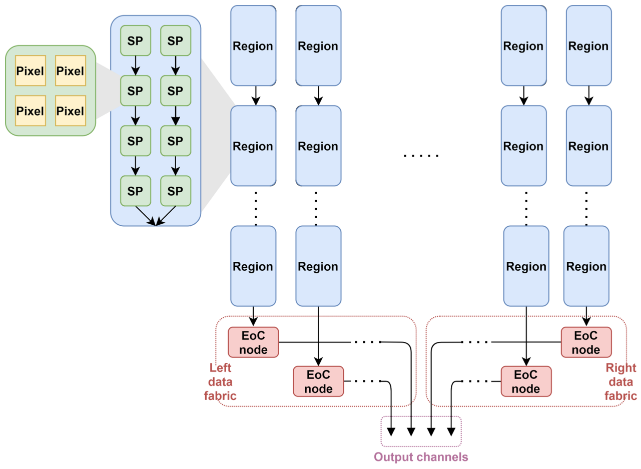

Fig. 6. Readout architecture example.

and formatting a pixel packet. Similarly, the discriminators in the local pixel network can trigger the data processing methods connected to the pixel via `sc_port` elements.

The primary goal of modelling these digital circuits is to understand the impact of the Time-to-Digital conversion performance (conversion delay, resolution, temporal pile-up) on the entire system and to assist designers in developing local data processing solutions.

C. Readout network

The goal of the readout network is to route the digital packets containing pixels' hit data from each pixel to the chip's output channels as efficiently and quickly as possible. This is typically achieved using a hierarchy of readout nodes connected in a network, which aggregates packets in the chip periphery where off-chip high-speed links are located.

A generic readout node receives, buffers, and transmits data packets to and from multiple other nodes. Different types of nodes are characterised by their position in the hierarchy, bandwidth, and arbitration or routing algorithms. In the PixESL framework, the model for the base readout node implements data transfer methods and memories for packet buffering, as seen in Fig. 5.

Communication is achieved with *initiator* and *target* sub-modules in each node that send and receive packets to and from other nodes. These sub-modules implement a packet transfer protocol using classes from the SystemC TLM 2.0 library, enabling multiple connections to and from each node. Both endpoints can be configured in terms of timing and bandwidth through runtime settings. Moreover, the base classes can be expanded to use different routing algorithms (for initiators) and arbitration strategies (for targets) or to include custom logic. The base node class implementation contains a member initiator and target, along with a base interface memory class to model the data buffer. This buffer allows data packets to be read and written by the TLM initiator and target, respectively.

Fig. 6 presents a readout architecture example featuring 3 types of nodes: SuperPixel (SP), Region, and End-of-Column (EoC). These nodes are organized in various layouts based on their hierarchical role. SPs aggregate data from four underlying Pixels and transfer it downward through double columns toward the Region's buffer. Regions are placed in tall columns to vertically relay packets towards End-of-Column nodes located at the chip's periphery, concentrating data into the final output channels.

The main challenge in the architectural design of the readout is effectively managing tens of thousands of input channels and pixels and concentrating their packets into a small number of output channels. A high-level model of the readout data path enables designers to carefully size bandwidth and buffers at each stage of the readout and establish optimal node connections to prevent congestion. An undersized and poorly optimized readout data path results in data losses and significant packet latency, requiring costly and complex back-end solutions to handle the data stream. Congestion within the network leads to storage overflow in nodes, causing data losses as new packets cannot be processed from pixels. Additionally, network congestion prevents packets from reaching the off-chip links, increasing latency. To aid designers in network optimization, configuration files allow users to swiftly customize the architecture by specifying the number and attributes of the nodes, and to generate node networks arranged in square grids.

IV. Design space exploration

In the design space exploration for pixel detectors, the chip's architecture is tailored to the application's requirements and optimized to match physical constraints such as power and area. The use of a comprehensive high-level model, from front-end to high-speed links, allows the designers to study the chip from a system-level view, evaluating how changes to any sub-module affect the whole device.

Thanks to the higher level of abstraction and fast simulation speed, the SystemC model can be used to simulate an architecture, evaluate its performance, and iterate on the ROC structure and the layout of pixels and readout nodes to reach a

TABLE I
PARAMETER SPACE OF VELOPIX UPGRADE

Parameter Name	**Node Group**	**Baseline**	**Proposal**
Array size [columns x rows]	Pixels	256x256	256x256
	SuperPixels	128x128	128x128
	Regions	64x16	128x8
	End-of-Columns	64	128
	Output ch.	8	16
Buffer depth [word]	Pixels	None	None
	SuperPixels	2	6
	Regions	2	6
	End-of-Columns	1	2
Clock [MHz]	SuperPixels	40	40
	Regions	40	40
	End-of-Columns	40	320
Max packet throughput [packet/cycle]	Pixels	1	1
	SuperPixels	1	1
	Regions	1	2
	End-of-Columns	1	2
Arbitration	Packet priority	Round-robin	Back-pressure
Key metrics results	Readout eff.	86%	99.98%
	Avg. latency	95 cy.	32 cy.
	Max. latency	1500 cy.	500 cy.

satisfactory level of performance. Compared to a system-level study carried out in SystemVerilog [3], this approach allows for simpler behavioural descriptions of the modules and faster simulation time, thus quicker design iteration to match the specification. For example, a simulation of the complete ROC system for a simulation time of 50 µs, from analog front-end to periphery, has a runtime less than 90 seconds, whereas an RTL model with a comparable simulation length would take around 2 hours.

In Table I, the parameter space for the Velopix upgrade ROC is detailed for each group of nodes, together with the key metrics of the baseline and upgrade proposal architectures.

A. *Readout network optimization*

PixESL provides the base classes to draw a rough architecture from front-end to readout, which can be customized to a specific system's functionality. During design space exploration, the layout and parameters of the chosen architecture can be parsed from external files. The architecture is parsed from file and it defines the virtual prototype's properties and structure, such as the number of rows and columns of each node group, the size of the packet buffers, or the data transmission parallelization and the clock speeds.

The virtual prototype is generated as a collection of groups of nodes, each characterized by its parameters and connection scheme, such as the network in Fig. 6.

During the design space exploration phase, PixESL's SystemC model generates detailed logs on the status of the nodes and their data buffers, and tracks the packets to extract relevant metrics about the device under test (see the results in Table I). The most relevant metrics to evaluate performance are the readout efficiency, which shows the percentage of packets exiting the model out of all of valid ones injected in the readout data path, and the average packet latency, which measures the average time spent by a packet in the systems from its generation to the output channels. These metrics and other plots (see Fig. 7 and 8) can be automatically extracted with set of Python scripts, shipped with PixESL to quickly evaluate and optimize the system architecture.

This modelling approach and integrated performance extraction tools were employed to study the readout architecture of an upcoming chip, Picopix, which targets the upgrade of the Velopix chip [2] with the latest requirements from the physics experiment. The study goal was to show that the upgraded front-end and readout system could sustain the data rates forecasted by simulations of the experiment's environment. Due the higher luminosity and smaller pixel pitch size (50 µm), the pixel hit rate would increase to 3.5 GHz cm^{-2}, and together with the added timing information with 25 ps resolution, this would bring the pixels' raw data throughput above 250 Gbit s^{-1} without any on-chip processing and data reduction, compared to 16 Gbit s^{-1} in Velopix [5].

The baseline Velopix architecture [5] was simulated with the latest stimuli and its key metrics are reported in Table I, and the analysis results from Fig. 7 show that the buffers in the central columns are often completely full (yellow color). This is due to the spatial hit distribution that peaks toward the top-center of the array and generates congestion in that area, which increases the latency for those packets and causes data loss in those columns.

The main result of this study proposes an output bandwidth increase to 100 Gbit s^{-1} to match the higher pixel hit rate, and an optimized arbitration algorithm with back-pressure to significantly reduce the packet congestion, thus minimizing latency and data losses. Thanks to pixel-level clustering and filtering, and data reduction in the periphery, the chip's bandwidth is halved compared to the raw pixel output.

Table I compares the baseline and proposed architectures showing the improvements in the key metrics thanks to a more robust readout data path: doubling the Region columns from 64 to 128 and the subsequent data path nodes removes the congestion in Fig. 7, thus improving the readout efficiency well above the 95 % specification and reducing the average latency. The shape of the latency distribution changes as well: the tail of the Poisson curve and the maximum latency are significantly shortened, thus allowing for faster and simpler data aggregation in the back-ends, as shown in the next subsection. Similar studies were carried out to optimize the memory buffer sizes in the nodes, maintaining a high readout efficiency and low latency.

B. *Data processing virtual prototyping*

Together with network studies, a high-level system model allows the design and integration of new modules and functionalities with the virtual prototype to evaluate their potential impact on the whole device. Thanks to the high-level description, the prototype's functionality development is simpler and faster than using an HDL. As an example, the design of an event sorter module is reported.

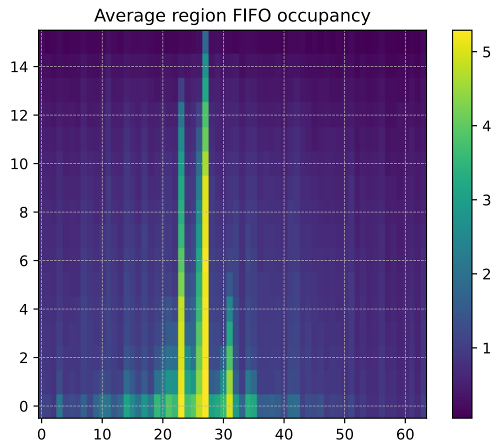

Fig. 7. FIFO buffer occupancy and congestion in baseline architecture.

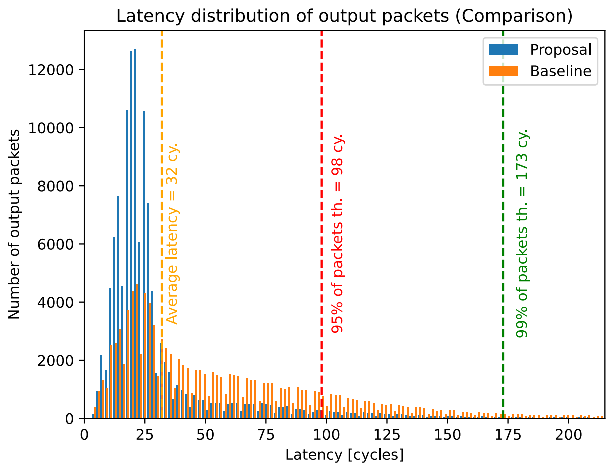

Fig. 8. Output packet latency, baseline and proposal architectures comparison.

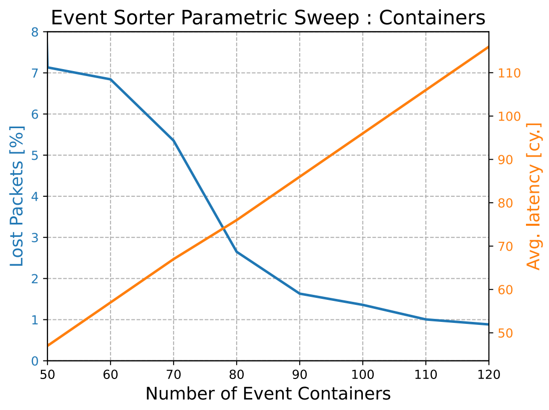

Fig. 9. Parametric sweep on the amount of containers in the event sorter.

The event sorter is a data processing module capable of gathering all the digital packets from the same event, grouping them in a single event data frame, which is then sent off-chip. In a data-driven architecture like Picopix, it reduces bandwidth by providing a timestamp tag per event frame instead of per digital packet, almost fixed latency instead of variable latency, and on-chip event reconstruction, offloading the back-end from this task.

This module stores pixel hits from the same physics event by collecting packets with the same event ID in an associative container implemented with one or more SRAM banks. After the event in the container has reached a certain latency, the container is timed out, and the packets inside are ready to be read out.

Thanks to the event sorter model in SystemC, the containers' amount, size, and packet throughput can be minimized to reduce the area and power costs while ensuring that target performance is met. In the specific Picopix case, the packet sorter was sized so that the maximum event latency, defined by the number of containers, would assure that less than 1 % of the packets were lost due to excessive latency and could not be stored in time in the corresponding container. A parametric sweep on the number of containers in the event sorter and its effect on the fraction of lost packets and overall packet latency after accumulation can be found in Fig. 9.

It also allows the event sorter to be co-optimized with the readout network. In fact, a limited bandwidth on the event sorter input can cause traffic congestion in the whole readout network, inducing packet loss and additional latency, which in turn also worsen the event sorter performance. In order to properly size the event sorter, this effect has to be taken into account, thus requiring a system level study including both readout network and sorter.

The full-chain simulation of the Picopix virtual prototype, including a model of the event sorter, demonstrated that, with the proposed architecture from Table I, an event sorter with a throughput of two packets per event per cycle does not create any measurable congestion. These results, together with the sizing of the depth (128 packets per event frame) and amount of containers (128), enabled the start of the hardware design.

V. Integration in UVM-based verification environment

One of the main advantages of the proposed approach is the reusability of the C++-based virtual prototype during the design verification phase. The virtual prototype can be adapted to act as a stimuli driver for the RTL simulation, and as a reference for the SystemVerilog implementation. The reuse of the same model in the design space exploration phase to match specifications, and to verify the design guarantees continuity between the design phases, as shown in Fig. 1.

The different models described in Sec. III provide some advantages also for the verification task. The Analog pixel front-end (III-A) avoids costly analog/mixed-signal simulations to drive the RTL simulation, and distinguishes between inefficiencies caused by front-end performance or design bugs to be fixed. Likewise, the Digital pixel front-end (III-B) allows the identification of data losses by design or bugs in the early stage of the processing and readout chain, simplifying the debug process. The SystemC Pixel reference models the correct functionality of the local pixel network and its effect on the packet predictions. The Readout network (III-C) provides cycle-accurate traffic information that allows to distinguish between congestion-related problems and design bugs.

Following the use described in Section IV, PixESL was also employed during the design verification phase in the Picopix project. Only the Analog and Digital pixel front-end models were integrated in the first implementation of the verification environment, whereas the readout network will be included once the hardware design and verification environment reach a more mature development stage.

In this project, the verification environment exploits a UVM-based design, and the PixESL integration proposed in this section aims to be easily adaptable to any environment of this kind. The pixel matrix is a SystemC module that contains Analog and Digital pixel front-end models and exposes several ports and methods to interface with SystemVerilog. Its size can be set at compile time and specific pixel functionalities can be configured with DPI calls through the UVM Verification Component (UVC) to provide reference in different operating modes.

The SystemC module is instantiated in the SystemVerilog testbench through a shell module that contains port definitions matching the `sc_port` found in the SystemC matrix. The simulator recognizes this shell as a SystemC module whose functionalities are defined by C++ code, and it is co-simulated with the SystemVerilog RTL design sources. Two communication mechanisms are employed to control and extract data from the matrix: ports and SystemVerilog DPI functions.

Ports allow seamless communication between an `sc_port` defined in SystemC and the corresponding port in the SystemVerilog shell. In the SystemC pixel matrix, they are used to align the timing between the UVM testbench and reference model through a clock port and transmit status and control signals between the reference model and the testbench, such as signals to report the end of the simulation or reset the reference.

SystemVerilog DPI functions are used for communication to and from the pixels and return the reference's predictions. DPI calls achieve the two main goals of the PixESL co-simulation: modelling the AFE to drive the RTL simulation, and predicting the output packets of the digital pixels. DPI functions are used to configure the pixels from the testbench, calling a SystemC function in the SystemVerilog simulation to pass a pixel address and its configuration. Similarly, the `get_predictions(...)` function allows the testbench to retrieve the packets predicted by the reference model. Conversely, SystemVerilog functions can also be called within the SystemC code: each pixel's AFE model drives the corresponding threshold discriminator signal in the testbench, by setting its value with function calls from SystemC to SystemVerilog.

In summary, the PixESL framework ships with a PixESL-UVC add-on that includes the shell, interface, and UVM agent to instantiate and run the pixel matrix within the Picopix verification environment.

VI. Conclusion

The PixESL framework is a SystemC-based modelling tool for prototyping pixel detector chips. It enables early architectural studies and design space exploration at the system level.

It features a components library of front-end and readout block models integrated into an environment to generate and simulate virtual prototypes and extract performance metrics. Thanks to the versatility of SystemC, it can accurately model analog, mixed-signal and digital circuits, while retaining the simulation speed and behavioural description of a high-level system model.

PixESL is currently employed in the upgrade of a HEP pixel detector, both as a virtual prototype for architectural studies and design space exploration, and as a reference model for functional verification.

Acknowledgements

This project is funded by CERN Experimental Physics department as part of EP-R&D Programme on Technologies for Future Experiments, under the Work Package 5 on IC Technologies.

References

[1] M. Garcia-Sciveres, "Hybrid pixel readout integrated circuits", in *Nuclear Instruments and Methods in Physics Research*, 2023, [Online] Available: https://doi.org/10.1016/j.nima.2023.168725

[2] E. L. Gkougkousis, "An LHCb Vertex Locator (VELO) for 2030s", in *Proceedings of the 31st International Workshop on Vertex Detectors* (VERTEX2022), 2023, [Online] Available: https://journals.jps.jp/doi/ref/10.7566/JPSCP.42.011028

[3] T. Poikela, "Design and Verification of Digital Architecture of 65K Pixel Readout Chip for High-Energy Physics", chap. 3-5, CERN-THESIS-2010-123.

[4] S. Spannagel et al., "Allpix2: A modular simulation framework for silicon detectors", in *Nuclear Instruments and Methods in Physics Research*, 2018, [Online] Available: https://doi.org/10.1016/j.nima.2018.06.020

[5] T. Poikela et al., "VeloPix: the pixel ASIC for the LHCb upgrade", in *Journal of Instrumentation*, 2015, [Online] Available: https://doi.org/10.1088/1748-0221/10/01/C01057

[6] L. Rossi, P. Fischer, T. Rohe, N. Wermes, "Pixel detectors: from fundamentals to applications", chap. 3, in "Particle acceleration and detection" series, Berlin, Springer, 2006