POLITECNICO DI TORINO, GRENOBLE-INP, EPFL

Master's Degree in Micro- and Nanotechnologies for Integrated Systems

Politecnico di Torino

Master's Degree Thesis

Modeling and implementation studies of pixel readout architecture for on-chip processing in HEP applications

Supervisors

Dr. Davide CERESA

Prof. Guido MASERA

Prof. Adil KOUKAB

Candidate

Francesco E. BRAMBILLA

September 2023

Abstract

EN:

As the complexity of particle detectors in High Energy Physics rises, new approaches to system prototyping become necessary to deal with higher data rates than ever before.

The first chapters of this thesis propose an Electronic-System Level approach to be applied to pixel detectors systems, and provide a SystemC framework to carry out architectural exploration at a level of abstraction above RTL. It can be used to study the efficiency of a chosen architecture, while sweeping the parameters of interest and obtaining figures of merit. With this tool, we were able to prototype a read-out chip able to support the hit rates for the next upgrades of the LHCb upgrade.

The last part of this thesis details the design flow of an on-chip processing module for the same upgrade, showing how the SystemC framework could be integrated as a starting step in system design for ASICs and how a physical layout could be derived from the abstract model.

Abstract

IT:

Con l'aumento della complessità dei rivelatori di particelle nella fisica delle alte energie, diventano necessari nuovi approcci alla prototipazione dei sistemi per gestire velocità di dati più elevate che mai.

I primi capitoli di questa tesi propongono un approccio a livello di sistema elettronico da applicare ai sistemi di rivelatori di pixel, e forniscono un framework SystemC per effettuare l'esplorazione architetturale a un livello di astrazione superiore all'RTL. Questo strumento può essere utilizzato per studiare l'efficienza di un'architettura, ottimizzando i parametri di interesse e ottenendo figure di merito. Con questo strumento, siamo stati in grado di prototipare un chip di lettura in grado di supportare il flusso di particelle per i prossimi aggiornamenti dell'upgrade di LHCb.

L'ultima parte di questa tesi illustra il flusso di progettazione di un modulo di elaborazione su silicio per lo stesso upgrade, mostrando come il framework SystemC possa essere integrato come passo iniziale nella progettazione ASIC, e come un layout fisico possa essere derivato dal modello astratto.

Abstract

FR:

Avec l'augmentation de la complexité des détecteurs de particules en physique des hautes énergies, de nouvelles approches de prototypage de systèmes deviennent nécessaires pour gérer une bande passante plus importante que jamais.

Les premiers chapitres de cette thèse proposent une approche au niveau du système électronique à appliquer aux systèmes de détecteurs de pixels, et fournissent un cadre SystemC pour effectuer l'exploration architecturale à un niveau d'abstraction supérieur au RTL. Il peut être utilisé pour étudier l'efficacité d'une architecture choisie, tout en optimisant les paramètres d'intérêt et en obtenant des figures de mérite. Grâce à cet outil, nous avons pu prototyper une microchip de lecture capable de supporter les taux de réussite des prochaines mises à niveau du LHCb.

La dernière partie de cette thèse détaille le flux de conception d'un module de traitement sur silicium pour la même mise à niveau, en montrant comment le cadre SystemC pourrait être intégré comme étape de départ dans la conception de systèmes pour ASIC et comment une réalisation physique pourrait être dérivée du modèle abstrait.

Acknowledgements

This thesis work has been carried out in a team and I would not have been able to achieve the same level of results without the collaboration and help of my supervisor, Dr. Davide Ceresa, and my colleague Jashandeep Dhaliwal. I would like to thank them for the opportunity of working here at CERN, what they have taught me about detectors and digital systems, and the help in developing this thesis.

I would like to thank my academic supervisors, prof. Guido Masera and prof. Adil Koukab for their corrections and tips in writing this thesis, and prof. Carlo Ricciardi for coordinating and overseeing the Nanotech for ICT master degree.

I am grateful to the other people in EP-ESE-ME for the technical help and advice in software and hardware design, and the less technical coffee chats that we had: Marco Andorno, Alessandro Caratelli, Stefano Esposito, Risto Pejasinovic, Anvesh Nookala.

I am glad to have met and spent time with my fellow Nano18 students, who have been a fantastic and much needed support during these 2 years of studies, and everyone in EP-ESE for their welcome and interesting discussions.

Lastly, I send my thanks to my family and friends in Italy and abroad, for their support and time spent together.

Francesco E. Brambilla
Geneva, Sept. 2023

Table of Contents

List of Tables v

List of Figures vi

Acronyms ix

1 Introduction to read-out chips in High Energy Physics 1
- 1.1 CERN . . . 1
  - 1.1.1 Working at CERN in EP-ESE-ME . . . 1
- 1.2 Pixel detectors at CERN . . . 2
- 1.3 Hybrid pixel sensors architecture . . . 3
- 1.4 Designing read-out chips . . . 7
- 1.5 State of the art . . . 8

2 Modeling read-out in SystemC: Pix-ESL framework 10
- 2.1 High-level models for detectors' architectural exploration . . . 10
  - 2.1.1 Previous works on high-level models . . . 11
- 2.2 Electronic System Design approach . . . 11
- 2.3 Modeling language choice . . . 12
  - 2.3.1 Limitations of high level modeling . . . 12
  - 2.3.2 The SystemC library . . . 13
  - 2.3.3 TLM 2.0 in SystemC . . . 14
- 2.4 Pix-ESL: pixel read-out prototyping at high level . . . 16
  - 2.4.1 Core Pix-ESL functionalities . . . 17
  - 2.4.2 Architectural exploration tools in Pix-ESL . . . 19

3 Data-driven read-out: Velopix-2 case study 24
- 3.1 Velopix2 overview . . . 24
  - 3.1.1 LHCb and VELO . . . 24
  - 3.1.2 Velopix and Velopix2 chips specifications . . . 26
- 3.2 Velopix architecture . . . 27

3.2.1 Velopix2 layers . . . 28
3.2.2 Velopix network . . . 31
3.3 Study results . . . 32
3.3.1 Input data analysis . . . 32
3.3.2 Configurations and results . . . 32

4 On chip Sort&Bin: from prototyping to implementation 41
4.1 On-chip processing advantages . . . 41
4.2 Architecture . . . 42
4.3 Design flow . . . 44
4.3.1 Pix-ESL modeling . . . 44
4.3.2 RTL design . . . 45
4.3.3 Implementation in a 28 nm technology . . . 47
4.4 Results . . . 51
4.4.1 Verification . . . 51
4.4.2 Implementation . . . 52
4.5 Next steps . . . 52

Bibliography 55

List of Tables

3.1 Velopix design space parameters 31
3.2 Parameters in the 3 configurations of interest. 33
3.3 Results from the 3 configurations 34

List of Figures

1.1 Atlas Inner Detector Cutout 2
1.2 Read-out and sensor chips bonded with solder bumps[1] 4
1.3 Analog&Digital front-end and super pixel node on a read-out chip[2] 4
1.4 Simple Pixel Matrix partitioning into layers 6
1.5 Pixel hits map, each dot in the image represents the number of hits on a certain pixel over the time of the study (∼900 BX), coloured bar indicates the amount of hits in the study period per pixel. . . . 8

2.1 SystemC language overview [11]. 13
2.2 TLM protocol with phases and timing delays [10]. 15
2.3 Pix-ESL overview. 16
2.4 Base and functional layers schematic representations. 17
2.5 Abstract Pix-ESL read-out nodes in the SystemC core, note that layers here represent a generic tree structure. 18
2.6 Pixel hits injected in the model, after front-end processing. 20
2.7 Average pixel hits per column. 21
2.8 Latency distribution example. 22
2.9 FIFO occupancy in the region layer for different configurations, full and deadlocked elements in yellow. 23

3.1 LHCb schematic view[13]. 25
3.2 VELO upgrade cut-out. 26
3.3 Velopix chips positioned in an L-shape around the beam. 27
3.4 Velopix read-out chip architecture 28
3.5 Velopix off-chip connections 29
3.6 Pixels grouped in a Superpixel (a), superpixels grouped in a region (b) 29
3.7 Regions connected in columns and with EoC data nodes 30
3.8 Distribution of the number of pixel hit per BX 33
3.9 C1: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region]. 35

3.10 C1: Packet latency distribution (a), output channel utilization (b). . 36
3.11 C2: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region]. 37
3.12 C2: Packet latency distribution (a), output channel utilization (b). . 38
3.13 C3: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region]. 39
3.14 C3: Packet latency distribution (a), output channel utilization (b). . 40

4.1 Sort&Bin module schematic view. 42
4.2 Bin sub-module schematic view. 43
4.3 Frame size distribution. 45
4.4 Packet latency distribution at the Sort&Bin input. 46
4.5 Effects of the number of bins on grouping efficiency (left axis) and total latency (right). 46
4.6 Sort&Bin floorplan with bin sub-module detail 49
4.7 Clock tree delays 50
4.8 Layout after routing 50

Acronyms

$\eta_{readout}$ Read-out Efficiency

ASIC Application-Specific Integrated Circuit

ATCA Advanced Telecommunications Computing Architecture

ATLAS A Toroidal LHC ApparatuS

BX Bunch Crossing

CERN Conseil Européen pour la Recherche Nucléaire (en. European Council for Nuclear Research)

CMOS Complementary Metal Oxide Semiconductor

CMS Compact Muon Solenoid (general-purpose detector experiment at LHC)

CTS Clock Tree Synthesis

DAQ Data Acquisition

DN DataNode

EDA Electronic Design Automation (tools)

EoC End-of-Column

EP-ESE-ME Experimental Physics - Electronic Systems for Experiments - MicroElectronics (CERN department)

ESL Electronic System Level

FE Front-End

FIFO First-In First-Out memory

FoM Figure of Merit

FPGA Field Programmable Gate Array

FSM Finite-State Machine

GaAs Gallium Arsenide

HL-LHC High Luminosity - LHC, planned upgrades

HLD High-Level Design

INFN Istituto Nazionale di Fisica Nucleare

IP Intellectual Property

LHC Large Hadron Collider

LHCb Large Hadron Collider Beauty, experiment on beauty quarks

PCIe Peripheral Component Interconnect express

PLL Phase-Locked Loop

PR Pull Request

R&D Research and Development

RTL Register-Transfer Level

SC SystemC

SoC System-On-Chip

SP SuperPixel

SRAM Static Random-Access Memory

SysC see SC

TCL Tool Command Language

TLM Transaction-Level Modeling

ToA Time-of-Arrival

ToT Time-Over-Thresold

VELO VErtex LOcator, a LHCb particle tracker

VHDL VHSIC (Very High-Speed Integrated Circuit) Hardware Description Language

WP WorkPackage

Chapter 1

Introduction to read-out chips in High Energy Physics

1.1 CERN

The European Organization for Nuclear Research, known as CERN from the French acronym, is one of the world's largest and most renowned research institutions in particle physics. It is situated on the border between Switzerland and France, and it is funded by a council of member states, including many European countries and further away ones (such as India and Israel).

Pushing the boundaries of knowledge in particle physics requires expertise in many engineering fields, in fact most of CERN personnel works as engineers designing and maintaining the LHC complex: from the particle beams or the cooling systems, to the IT services or detector sensors, everything is achieved through a collaboration of hundreds of organisations and thanks to tens of thousands of people.

1.1.1 Working at CERN in EP-ESE-ME

This thesis work has been carried out at CERN in the EP-ESE-ME department as part of my Master's degree, under the local supervision of Dr. Davide Ceresa (CERN, Staff) and in close collaboration with Jashandeep Dhaliwal (CERN, Fellow). The EP-ESE-ME department devotes parts of its workforce towards R&D goals, subdivided in workpackages (WP): these activities include testing new technologies for radiation hardness, developing IP blocks and SoCs for the detectors. My tasks in EP-ESE-ME fit into the WP5.4, which aims to design detectors for the next generation of particle accelerators, such as High-Luminosity LHC (HL-LHC) upgrades. These upgrades plan on increasing the energy of the collisions and

particles generated to improve the chances of finding exotic particles, but require a technological leap in detector performance and data rates.

1.2 Pixel detectors at CERN

In the particle physics research carried out at CERN, many different types of particle detectors are employed: calorimeters, spectrometers, trackers use different technologies and are placed in different spots around the particle beam (e.g.: calorimeters and muon spectrometers are bulky and placed in the external ring, trackers need to be as close as possible to the collision spot and are placed in the inner detector, see Fig.1.1).

Particle detectors are synchronous systems, where the main timing reference is the rate at which particles are colliding in the beam: these events are called bunch crossings (BX) and the time between them is fixed in LHC to 25 ns, thus corresponding to a 40 MHz clock in the system. This signal is used as a timing reference throughout the detectors, and will be continuously referenced in this work.

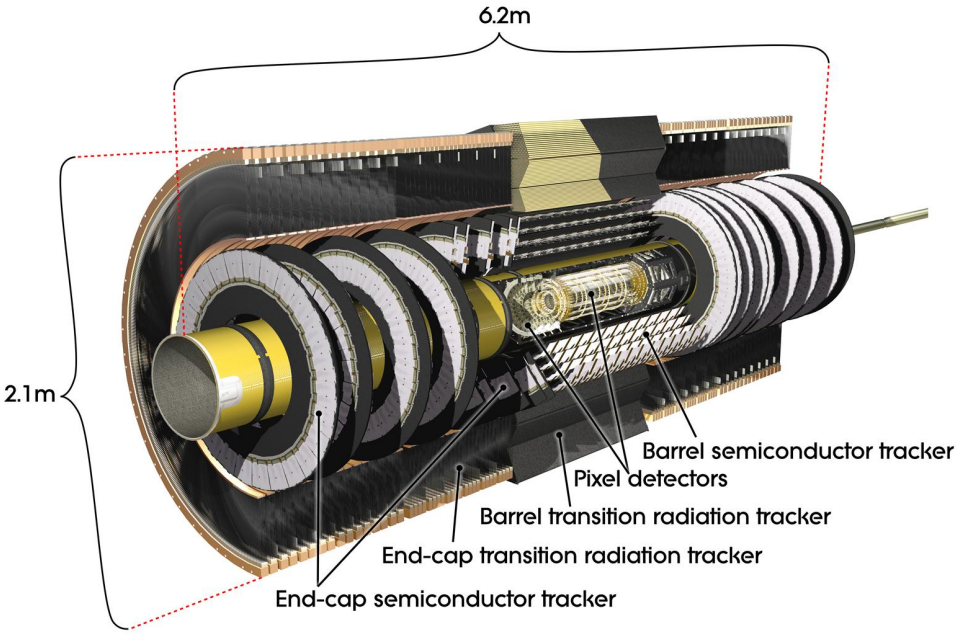


Figure 1.1: Atlas Inner Detector Cutout

From a system level point of view, particle detectors can be split into a front-end and back-end: the former comprises the particle sensors and the first layer of read-out electronics, the latter refers to the infrastructure that stores and elaborates

the data provided by the front-end. Physically, the front-end corresponds to a set of sensors with their read-out chips, connected through an optical fibre to a large server or CPU farm, the back-end.

Pixel detectors are a kind of particle detectors that employ active pixels, rectangular regions of a few hundred square microns on semiconducting substrates, inside a large matrix to observe particles.

This thesis will focus mostly on the pixel detectors used in the inner section, where greater precision is required to pin-point the time and location of particle interaction with the sensors, but its findings could be applied to other kind of detector systems. Pixel detectors work like a digital camera sensor, whose goal is to reconstruct the image created by the interaction of high energy particles or photons on a pixel matrix. This is then read out and data is transferred off-chip to be processed and stored for analysis.

The pixel detector chips can be monolithic, meaning that the read-out and the sensing are performed on the same piece of silicon: this allows for economic savings as there are significant costs in manufacturing and assembling the other kind of detector chips.

Instead, hybrid detectors rely on a separate pixel sensor chip, which can be customized for the given imaging application and has less constraints, and a dedicated read-out chip, which has to convert the raw analog signal coming from each pixel into a data packet and transfer it off-chip. There are many advantages to this approach, the main ones being the customizability of the sensor chip (f.e. diamond substrates) and the ability to use standard CMOS processes for the read-out one. Since CERN uses mostly this kind of pixel chips, a more detailed explanation on their architecture will be given in the next section.

1.3 Hybrid pixel sensors architecture

In Fig 1.2 the sensor and read-out chips are bonded together with solder bumps directly on top of the read-out one, and more modern chips are using through-silicon-vias (TSVs) to connect the backside of the read-out chip to their power supplies and FPGAs for connectivity.

The sensor chip usually has no active electronics on board and may use bias voltages to improve the collection of charges in the substrate. The major advantage of using a separate sensor is that its material can be tailored to the energy of the particles and photons to be collected, for example in [1] GaAs is used as sensing substrate, whereas it's non-standard material for CMOS processes and would be very hard to integrate analog and digital circuitry on it.

The read-out chip follows an ordered structure: sensors' signals are elaborated in the front-end, then digitized data is passed to the superpixels (SP), which group

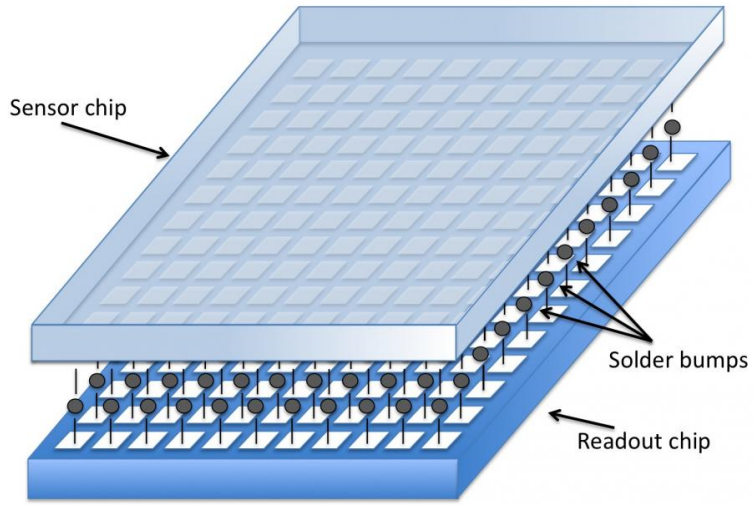


Figure 1.2: Read-out and sensor chips bonded with solder bumps[1]

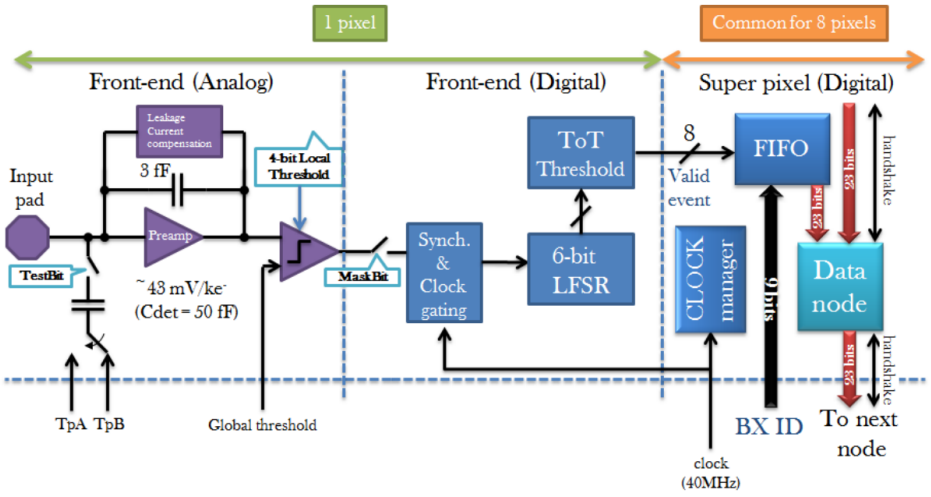


Figure 1.3: Analog&Digital front-end and super pixel node on a read-out chip[2]

together different pixels and may perform some clustering or processing, data is

then routed through a certain data-flow to reach the periphery. The study and optimization of this data-flow or data-path is the main focus of this thesis work.

It is important to note that front-end has a different meaning based on the context: at system level it represents the first part of the data-path, the one were the sensing and raw data transmission are carried out; on the other hand, from the point of view of a pixel read-out chip, its front-end comprises the analog and analog-to-digital circuits connected directly to the pixel sensors.

The read-out chip has to connect its analog front-end to the pixels on the sensor, then transfer the data digitally to the periphery and out-of-chip. A schematic representation of the front-end and the first section of the digital data-flow can be seen in Fig.1.3. In this example each pixel has a corresponding input pad and a front-end (FE) that amplifies the signal generated by the charge on the sensor, this signal is then converted into a digital step-like function with a threshold discriminator and the impulse duration is recorded. This simpler design only checks for events with a duration longer than a certain Time-Over-Threshold (ToT) value, and injects a valid event in the data-flow. More advanced FEs can include functions like event clustering and rejection: for example Velopix can cluster together nearby pixel hits in a single data packet, or spot unwanted hit patterns and remove their packets. Other FEs can measure the Time-of-Arrival with resolution in the tens of picoseconds.

The digital data-path is design such that an equilibrium is struck between efficiency and cost in power and area. Elements are grouped inside a higher-order element class to share hardware when possible and reduce the amount of long connections: for example pixel FEs in Fig.1.3 can share some signals for power and area savings, like a high-frequency local clock for timestamping, and with low enough pixel hit rates a single Superpixel's bandwidth can efficiently transmit data from its underlying pixels. Specifically for Superpixels in newer detectors (Velopix and Timepix4), data clustering allows also to perform some basic processing and filtering as close as possible to the data source.

The data-path architecture is based upon the pixel matrix in the sensor(see Fig.1.4): firstly pixels are grouped together in superpixels (f.e. 4 pixels in a 2x2 area can be considered a single SP, where some circuitry is shared), then multiple SPs are connected to a Region via series or parallel connections (f.e. 4 SPs in a 2x2 area can be run to their Region in 2 series connections in 2 parallel columns, where the last element of each column connects to the upper Region). Then Regions can be connected in series and form a Column element, which eventually connects to and End-of-Column node which is part of an output channel.

For comparison, a full scale chip would be typically have a 256x256 or larger pixel array, and 8-16 output channels totalling $\sim 100 - 200\,\mathrm{Gbit\,s^{-1}}$ per chip depending on the pixel hit rate, and up to a hundred chips can be mounted in a detector system.

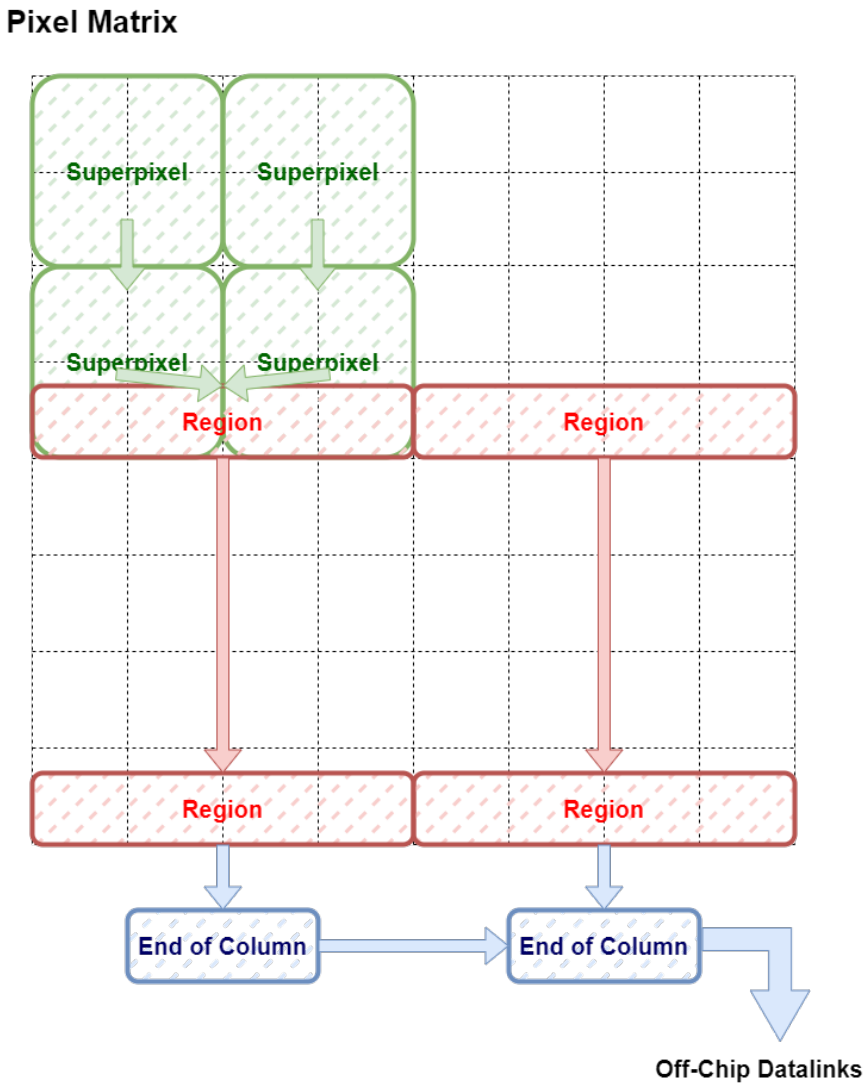


Figure 1.4: Simple Pixel Matrix partitioning into layers

Thus, the data-path is formed by nodes with different purposes and characteristics: pixels generate packets from the sensors, superpixels filter and cluster data, regions are stacked vertically to move data vertically and End-of-Column nodes create output channels with horizontal connections. These different kinds of nodes will be called *layers*, to represent how data moves through each layer to reach the output of the system.

Pixels, superpixels, regions, End-of-Column nodes are some of the layers used in the data-path of a read-out chip, but this approach could be extended to model communication layers above the pixel detector chips, for example optical

transceivers and data acquisition boards.

In fact, from a system-level point of view the read-out chip is part of the detector front-end, which has to interface with the data acquisition boards downstream. Due to radiation damage, the DAQ boards are far away from the detector ($50 - 60\,\mathrm{m}$) and connected with long fiber cables to the front-end via dedicated optical boards (GigaBit Transceiver, GBT), which interface through copper wires to the read-out chips. The data acquisition boards also transmit control and trigger signals to the front-end, and elaborate the data before sending it to storage or CPU farms for further analysis. A more specific system overview will be given in Sect.3.2.

1.4 Designing read-out chips

The design of read-out chips consists in finding the appropriate dimensions for each of these classes of elements, since the grouping of elements is necessary to extract data from the FEs, but at the same time causes issues due to the congestion and latency of the data packets in the various nodes through the data-path.

For example, a design with few elements in each layer (so with large grouping between elements) may reduce the latency because fewer hops are required for each packet to reach the end of the data-path, but may lose efficiency due to congestion caused by bottlenecks in the data-flow: since most read-out systems are column-based, if more packets are injected per cycle than the amount of columns the chip will eventually reach a dead-lock with fully saturated memories.

Their design must also take into account the area and power constraints, which may be violated by designs with smaller grouping sizes (i.e. a large amount of elements in each layer).

Another design variable is the size of the memory elements that can be added to each layer, usually in the form of First-In First-Out (FIFO). They are necessary to buffer the incoming data, and are very useful to smooth out the peaks in input rate. Since their implementation is expensive in terms of power and area, their size should be minimized.

The future objective for read-out chips is the integration of more advanced processing functions directly on chip, either at the end of the data-flow or between data-path nodes. This will be discussed more in detail in Chap.4, and has also been studied by J. Dhaliwal in his thesis works[3], where the use of tailored RISC-V cores for superpixel-level processing has been investigated.

The starting point for read-out chip design is the input data on the pixel sensors: every design choice is highly dependant on the amount of pixel hits generated per bunch crossing, known as the occupancy of the pixel matrix, and can be influenced also by the spatial distribution of hits, as that defines the local hit-rates. The hit matrix in Fig.1.5 is typical for head-on hadron collisions occurring as close as 5 mm

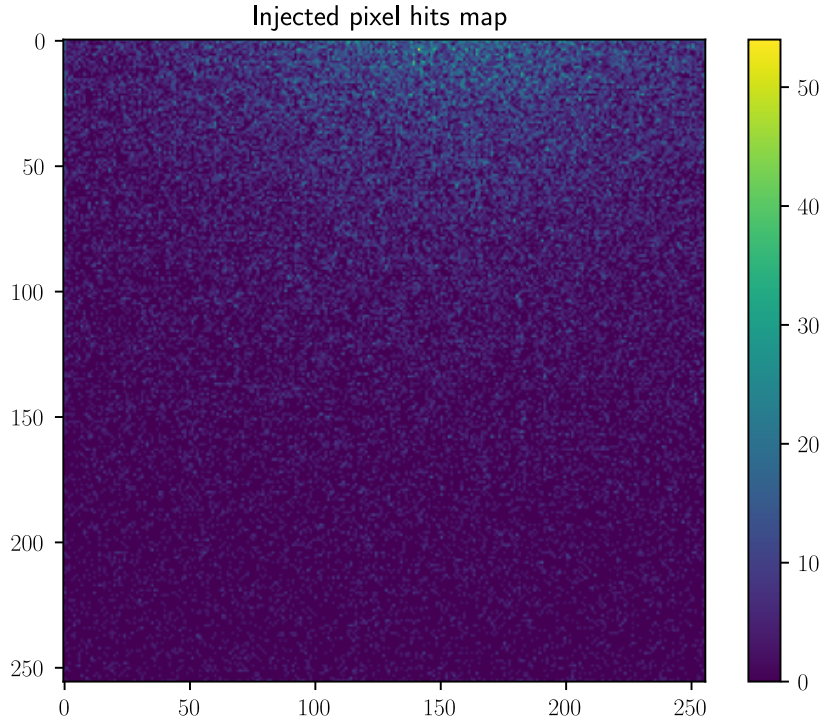


Figure 1.5: Pixel hits map, each dot in the image represents the number of hits on a certain pixel over the time of the study (∼900 BX), coloured bar indicates the amount of hits in the study period per pixel.

above the upper portion of the sensor (see also Fig.3.3): the extreme proximity of the chip to the collision point causes most hits on the sensor to be still close together. Due to the angled trajectories, other detectors downstream will not track the same amount of hits and their distribution will be more spread out.

shows that most hits occur in the top central region, thus meaning that the central columns must have enough bandwidth to support the critical hit-rates.

1.5 State of the art

Research in the field of pixel detectors at CERN has been ongoing for more than 25 years by now, and it has evolved beyond the boundaries of High-Energy Physics research, forming collaborations with multiple fields: radiation monitoring, astrophysics, spectroscopic X-ray imaging for Life Sciences, dosimetry for space

exploration and even education are some of the application where pixel detectors from the Medipix and Timepix family designed at CERN have been deployed.

This wide range of application is due to the versatility of their read-out ASICs, which can be mounted on different sensor substrates and the number of readings they can provide: photon counting, Time-of-Arrival and Time-over-Threshold. The photon counting mode is commonly used in X-ray imaging, where the hit count can be used to reconstruct structures or variations on density. The Time-over-Threshold measurement can be used to filter pixel hits below a programmable energy threshold, to reduce noise, perform spectroscopic sweeps or identify a particle by the charge generated on the sensor. Precise Time-of-Arrival measures allow to faithfully track particles passing through multiple chips in large detector systems, like the ones built at CERN.

Compared to the front-end of a commercial CMOS camera, these ASICs integrate many more functions and resources per pixel, typically in the form of a Time-to-Digital converter, counters, a programmable threshold comparator and a digital PLL used as a clock multiplier for precise time measurements.

The newest member in the family is Timepix4, an ASIC for read-out which can provide time-stamping precision $< 200\,\text{ps}$ and can be tiled on all sides to create large area detectors[4]. For tracking applications this is one of the most critical figures of merit, which is the reason why Velopix2 and its prototypes derive directly from Timepix4, pushing even more the time resolution to $< 30\,\text{ps}$ in the radiation-heavy conditions of LHC.

Over the various generations, Timepix and Velopix have evolved to cope with the increasing hit rates that were requested by CERN experiments. The first significant innovation was the introduction in Timepix3 and Velopix1 of event-based read-out architectures instead of frame-based ones: by instead of generating data in every pixel, only active pixels inject packets into the system. This zero-suppressed architecture increases the complexity of the read-out data-path, but allows the chip to be continuously read out instead of being able to provide only 1 frame every 40 cycles, which was the aim of the first Velopix chip.

The amount of features in the front-end is limited by the available space per pixel, which is determined by the $55\,\mu\text{m}$ pixel pitch in both directions used by the Timepix/Medipix family. But as more advanced nodes become available and radiation hardness tests are carried out, pixels can become smarter: the Velopix2 prototype includes a front-end capable of clustering nearby hits in a single packet or reject hits if particular patterns are detected. Additional processing or filtering can be also added inside or at the end of the data-path thanks to the 28nm node now available at CERN.

Chapter 2

Modeling read-out in SystemC: Pix-ESL framework

This chapter will describe the main focus of this thesis work, the modeling of the data-flow in particle detectors. The model and framework introduce a new way to prototype particle detectors at CERN, studying a high-level system description to assess performance and subsystem's specifications at an early design stage.

2.1 High-level models for detectors' architectural exploration

Due to the size and complexity of the next generation of particle detectors, architectural exploration at the detector-system level has emerged as a necessary step in their design flow, to increase the chance of first-time-right designs. Detector systems are composed by many different sub-systems and modules, whose specifications and requirements are impossible to determine beforehand, but must be found through system-level design space exploration.

The main goal of this project was to provide a tool that could improve this design flow to:

- **describe the detector** at system-level to identify bandwidth requirements, buffer sizing and data latency.
- **identify specifications** for individual sub-systems at the system architecture exploration step, before RTL description.

- **assess performance** early in the design process and meet the experiment's requirements. This is increasingly relevant as new technologies' costs reach the tens of M€.
- **provide a reference model** to simplify the functional verification step, where the high-level model can be co-simulated against the RTL.
- create a **virtual prototyping platform**, to study the usefulness and feasibility of new algorithms and processing capabilities from a purely functional point of view, detached from the complexity of implementation.

2.1.1 Previous works on high-level models

The kind of approach to architectural design presented in Pix-ESL is novel to CERN, where most of these studies are carried out directly in RTL, mostly SystemVerilog. For example, in the architectural studies for the pixel detectors in the next CMS upgrade, Dr. A. Caratelli used an RTL model with TLM for both prototyping and verification [5], which is the most widely adopted methodology at CERN.

The main inspiration for this project comes from earlier work by Dr. Tuomas Poikela [6], who modeled similar architectures in a purpose-built C++ simulator with a cycle-based and sequential model and using a TLM-like approach to data transfer.
Even though our framework followed an approximately timed approach and implemented actual SystemC TLM-2.0, we took many of the considerations in his doctoral thesis to abstract a more generic and reusable framework.

2.2 Electronic System Design approach

A working definition of ESL can be found in [7]: "the utilization of appropriate abstraction in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner, while meeting necessary constraints". This generic description fits perfectly with the scope and methodology of Pix-ESL: providing an accurate enough model to evaluate system prototypes and understand better the challenges in system design.

Given the goal set for this project in Sect.2.1, ESL seemed the best way to describe this modeling approach: the key ESL notions of **abstraction**, **exploration**, **reuse** and **automation** presented in [8] reflect those objectives. In fact, there is a need to abstract the system representation to a high enough level to be able to quickly explore architectures; in addition to that, abstract modules can be reused or integrated with IPs and should provide a golden reference for automated verification.

2.3 Modeling language choice

The choice fell on SystemC because it could provide a good enough description of the read-out chips architecture without requiring much knowledge about the RTL implementation, being one of the most promising ESL open-source languages [8]. It allows us to vary the number of modules, their properties and the arbitration between them with a greater degree of flexibility and ease, compared to using parametric modules in SystemVerilog or generic entities in VHDL.

The main reason for using a high level model lies in the need to quickly prototype and evaluate read-out designs in the early architectural exploration, where such massive layouts would be very expensive to simulate and would require unnecessary details in the RTL description at that stage. Nonetheless, simulation in SystemC is much faster and less resource intensive than RTL: the simulation runtime and memory utilization differ by almost a couple of orders of magnitude. Compared to a Verilog simulation of the new Velopix-2 chip, that was able to simulate an average of 0.1 BX per second of runtime, our model could be compiled in less than 30 seconds and run at 15 BX per second of runtime. Here BX refers to the injection event that occurs every 25 ns and is the timing reference for the whole detector system.

2.3.1 Limitations of high level modeling

It is important to recognize the limits of a high-level design (HLD) in order to relate simulation results to real attainable performance.

The obtained results can be compared relatively to each other, so a design with an higher in-model efficiency translates into a better real-world implementation, but their accuracy to real-world performance is highly dependant on the implementation itself. For example, in our model we could assume that all transactions take a fixed amount of clock cycles (one cycle for simplicity) since we expect all transactions to be resolved in the same time, however in an actual read-out chip the transaction could be split into multiple cycles because of design constraints. Thus, it is hard to precisely estimate latencies in an HLD.

In SystemC there are no limits on usable functions and data structures, so theoretically anything available in C++ can be simulated. Whereas this removes the burden of defining algorithms or physical memory structures, it may lead to oversights and designs which are impossible to reproduce on silicon or even describe in RTL.

Conversely, there are often constraints coming from the physical design of the analog and digital front-ends which may be overlooked in a HLD: for example accurate time-stamping requires higher clock signals generated locally, and the precision of this clock distribution net puts an upper limit on the sizing of certain

blocks.

Power consumption estimation is not present in base SystemC, but there are publications such as [9] that integrate this functionality in SystemC with minimal overhead. At the moment, we are investigating whether the number of read and write operations could provide useful power metrics when combined with technology-specific power estimates.

2.3.2 The SystemC library

The SystemC library is a C++ collection of functions and classes, defined by the IEEE 1666-2011 standard [10], that was developed out of the need to describe large system-on-chip (SoC) architectures with a level of detail suitable for simulation, design space exploration and verification.

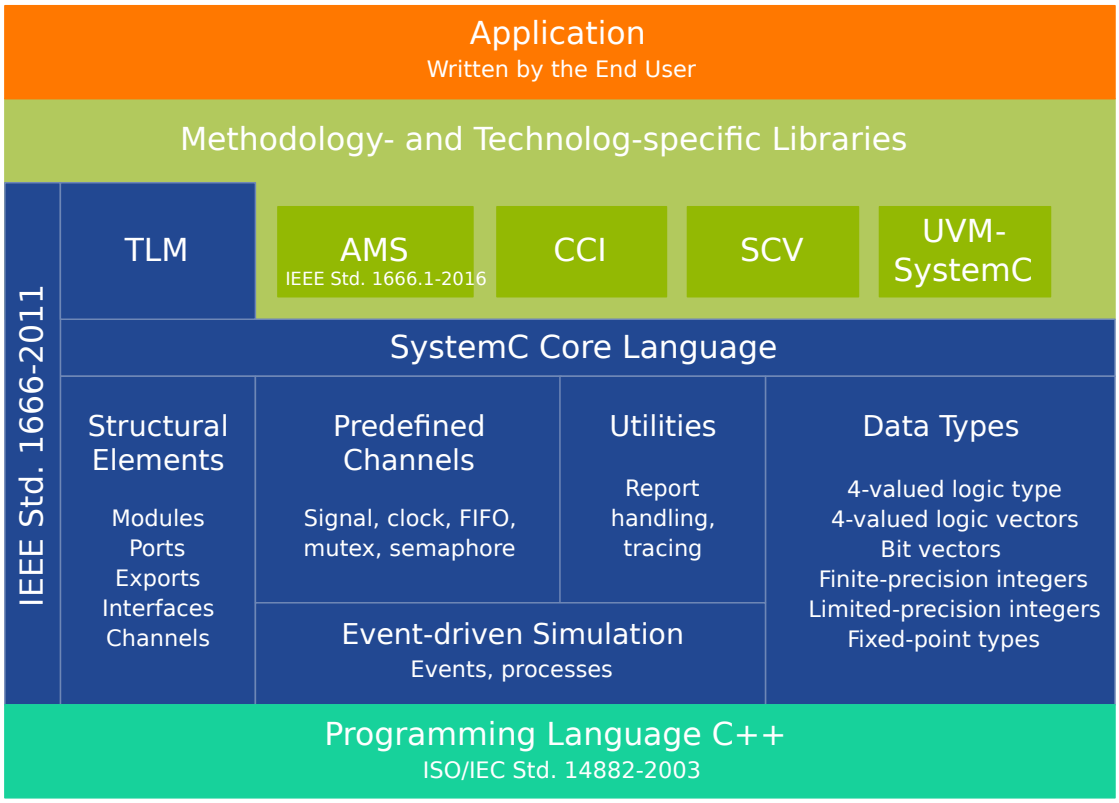


Figure 2.1: SystemC language overview [11].

The SystemC Core contains:

- **Structural Elements**: primitive classes to represent elements/modules and their data exchanges ports and protocols.

- **Predefined Channels**: implementation of data transfer and storage structures derived from primitive channels, i.e. memories and signals.
- **Data Types**: used to represent data values in the systems, similar types to those found in Verilog or VHDL.
- **Event-driven Simulation**: contains the SystemC simulation kernel, which uses events to schedule and trigger process execution and progress the simulation.
- **Utilities**: reporting and tracking classes for debugging.

Modules are a SC class that represent a design entity, and model parallel execution by registering certain class methods as `SC_METHOD` or `SC_THREAD` and making their execution sensitive to certain events, such as clock signals or incoming data requests. The SystemC kernel is intrinsically event-driven, which allows for substantial simulation performance gains if appropriate considerations are taken to avoid using a timing reference to trigger all sequential elements at every cycle.

The modeling of logic functions can be written in plain C++ methods called within the module upon their execution, thus algorithms can be studied without the complexity of their RTL implementation. Whereas this allows for fast prototyping and relative performance estimates, the designer must be aware that rarely the SystemC simulation will be cycle accurate to the implemented result, and that some operations or data structures might be impossible or prohibitively expensive in the latter phases of design.

2.3.3 TLM 2.0 in SystemC

Transaction-Level Modeling (TLM) is an approach to data transmission that encapsulates computing modules in a communication wrapper, to separate processing and transmission and use high-level function calls instead of slow RTL signals [8].

TLM 2.0 refers to a group of SysC classes which create an *interoperability layer*, comprised of ports, known as *sockets*, and *payloads*, data structures used to model data packets. This allows EDA and IP vendors to work with common methods to interface their modules: for example a vendor could provide its customers with a SystemC description for complex modules, such as RISC-V cores that would be instantiated in SoC prototyping and that would implement SC TLM 2.0 methods to communicate with other IPs or user-made modules.

Data transport in SC TLM-2.0 is achieved with payloads moving between *initiator* and *target* sockets, that act as start- and end-points for the communication protocol during its different phases.

This is clear in Fig.2.2, where an example of communication with the 4-phase standard TLM protocol is shown:

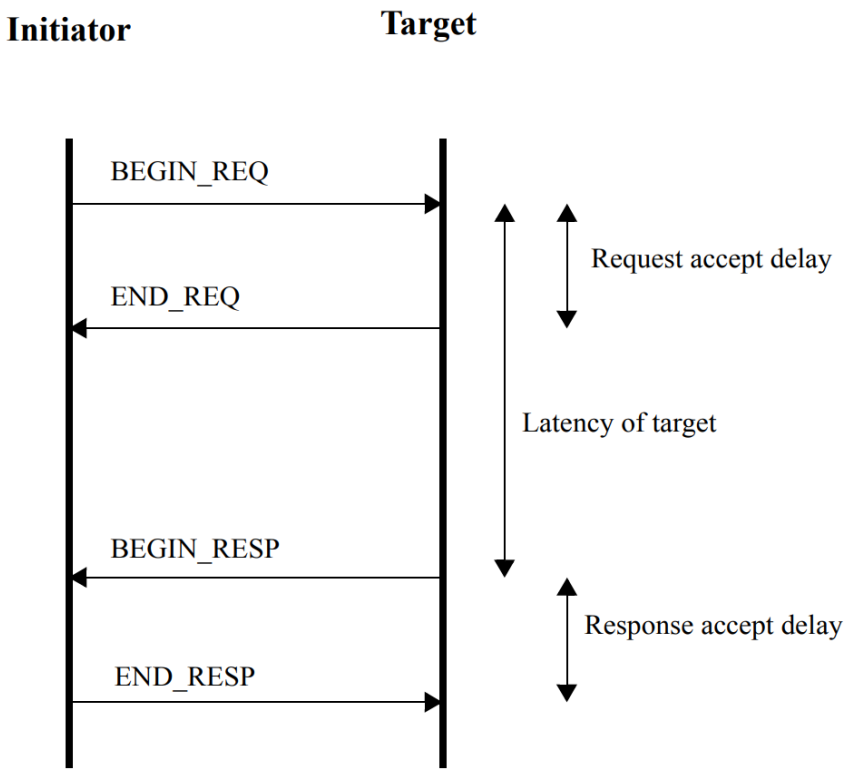


Figure 2.2: TLM protocol with phases and timing delays [10].

1. To start the transaction, the initiator's call to the *forward transport* method with the `BEGIN_REQ` phase triggers the execution of a function in the target module, registered in the target socket as the callback of the forward transport.

2. The target module answers the request with a *backward transport* call with the `END_REQ` phase, to signal its ability or otherwise to answer the request.

3. The target initiates the actual data transfer in the `BEGIN_RESP` phase on the backward path (i.e passing a payload in the backward transport call).

4. The initiator closes the transaction with the `END_RESP` phase.

It is up to the user to define the behaviour upon the transport calls, for example a *forward transport* method with the `BEGIN_REQ` phase could signal an incoming pull request for the target, which could answer with an acknowledge or denied response to the PR based on its internal state. A successful PR would then end with the data being received by the initiator, which would then store or process the incoming payload based on its purpose.

2.4 Pix-ESL: pixel read-out prototyping at high level

Pix-ESL is an architectural design space exploration tool, and it stands for **Pixel-Electronic System Level** and is an attempt at applying an ESL-like approach to prototype the next generation of detectors.

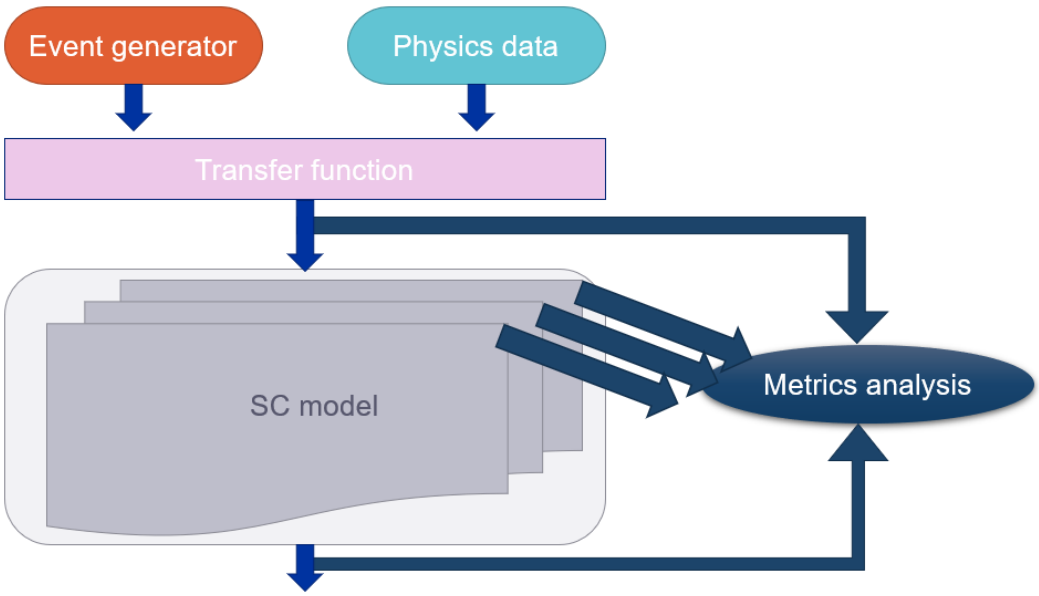


Figure 2.3: Pix-ESL overview.

Pix-ESL consists of a C++/SystemC core that models and simulates the flow of data through a generic path, which is hierarchically divided in different *layers* (see Fig.1.4 and Fig.2.5) that transfer data-packets from source to sink, for example in a read-out chip from the pixels' front-end to the chip periphery. Input data (i.e. pixel hits) can come from physics simulations or a built-in event generator, and can be passed to a *transfer function*, whose purpose is to model the analog front-end that feeds packets to the digital data-path.

The framework also includes a toolset (see Fig.2.3, *metrics analysis*) of Python scripts to analyse I/O and data occupation at any moment of the simulation, and evaluate the performance of each architecture. It allows the user to close the loop from input to output and perform suitable changes to its design to improve performance.

The SystemC model core is the most interesting module of the framework and will be treated in detail in Section 2.4.1.

2.4.1 Core Pix-ESL functionalities

The Pix-ESL core is a library of `SC_MODULE` derived classes that model functionally the layers in a read-out chip, since most of the work on Pix-ESL was focused on this first section of the detector data-path.

`SC_MODULE` is a SystemC core class that represent a design entity, similarly to a VHDL `entity` or Verilog `module` by using processes, communication instances (ports), variables and signals .

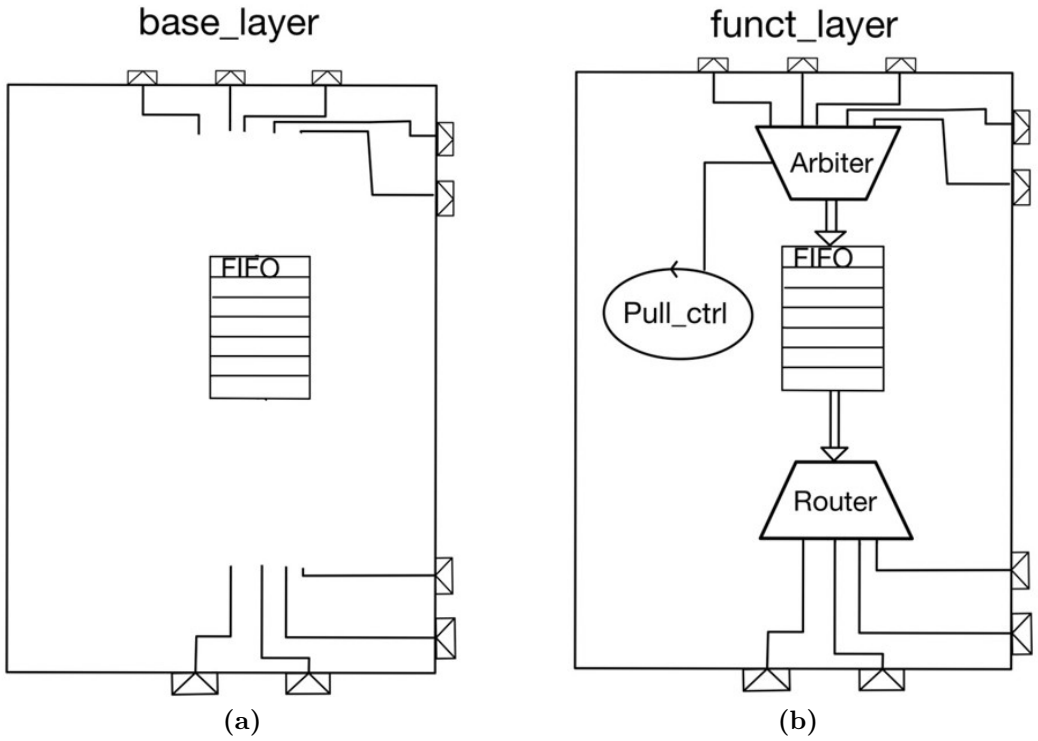


Figure 2.4: Base and functional layers schematic representations.

In order to lay the foundations for a reusable code base, were used C++ inheritance mechanisms were used to define different attributes of these modules in different stages of description.Two main classes/modules were developed: the `base_layer` and `funct_layer` as seen in Fig.2.4.

The base layer is the most generic representation of a read-out node: a structure with some memory in the form of FIFO elements and predisposition for communication using TLM-2.0 sockets. It declares virtual methods to be defined in the layers that implement those functionalities, like for example the *transport functions* that define how packets are moved between modules. It also includes members and

methods for debugging and metric logging, to be used in each function that needs to be monitored for performance extraction or debug.

The functional layer is derived from the base layer and its purpose is to implement the various processes and functions that define the operations of the node:

- **Arbitration**: the arbiter shall decide which of the possible sources can write into the module's FIFO. Since the model works on a pull-request (PR) basis, this function essentially issues pull requests to the chosen packet source.

- **Routing**: the router shall answer positively only one of the incoming PRs. In case a module contains multiple memories, it should also decide which memory to read from.

- **Transport**: in TLM-2.0 each initiator and target socket must register a method to interface, respectively, with incoming backward and forward transport calls from outside the module.

The functional layer is then used as a template to model the specific functions in each hierarchical layer that defines the architecture, like the chain of *pixels, superpixels, regions and end-of-columns* found in Fig.1.4. It can be used to model processing nodes as well, by defining which functions to apply to the data whenever it is read or written.

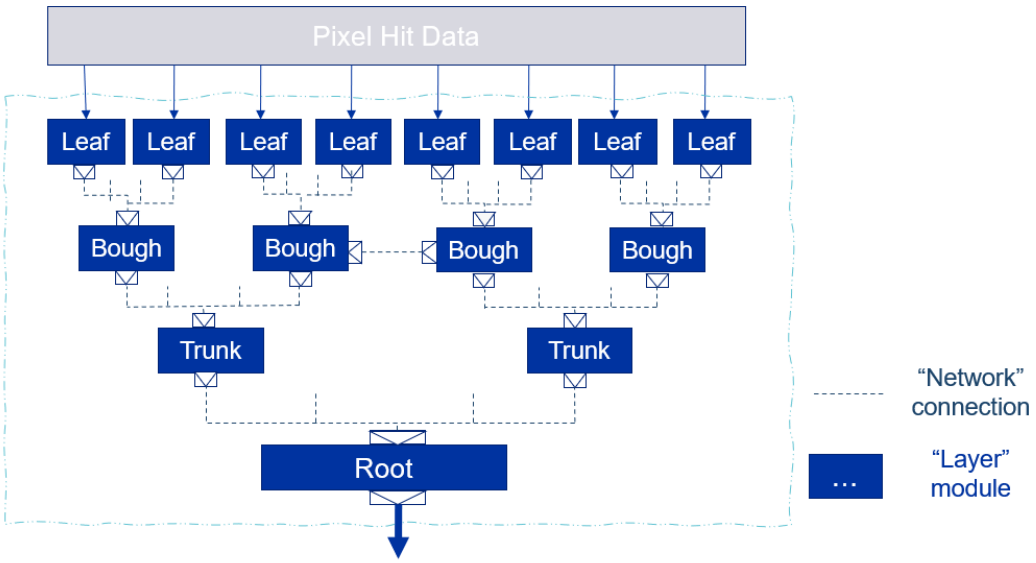


Figure 2.5: Abstract Pix-ESL read-out nodes in the SystemC core, note that layers here represent a generic tree structure.

After these modules are defined, we need to specify how they are connected to each other and what are the hierarchical relationships. Each module in every layer has to be able to communicate in four directions: with its **children** modules (in the layer underneath, hierarchically down, RX only), with its **sibling** modules (same layer, bidirectionally), and with its **parent** modules (layer above, TX only). In an abstract representation (see Fig.2.5) the architecture can be seen as a tree-like structure, where data propagates from the *leaves* to the *root*, moving through layers to be collected in a central spot at the end of the read-out chain.

2.4.2 Architectural exploration tools in Pix-ESL

The data analysis and performance extraction tools are fundamental to judge the quality of a certain architecture and to find its parameters to obtain the best performance.

Together with Pix-ESL, a collection of Python scripts was developed to pinpoint the inefficiencies in the system. I developed quickly a first version of this toolset in MATLAB in contemporary to the core model, but the images in this thesis will come from the more advanced Python version, developed by J. Dhaliwal.

Figures of Merit in read-out chips

Since the read-out chip is a sort of data-path, the main objectives of Pix-ESL is to make sure that it can support the given occupancy, to guarantee that most of the pixel hits can reach the chip periphery. Due to being modeled as an *always-pull* system, i.e. a node will request data until it has available space, if the FIFOs in the system are filled up more quickly than they are read, the resulting obstruction will propagate to the previous elements upstream. When this deadlock reaches the superpixel elements and they fill up, the result is that future pixel hits cannot be registered anymore, which is the reason why these system are optimized to extract the most out of their limited bandwidth.

This is also a consequence of the nature of data-driven pixel detectors: in fact pixel hits follow statistical distributions both spatially (see Fig.1.5) and temporally, which means that some regions will have higher occupancy in their FIFO memory due to the zero-suppression performed at pixel level.

To evaluate the designs, here are listed the figures of merit by importance for read-out chips:

1. **Read-out Efficiency**($\eta_{readout}$): defined as the number of packets that reach the end of the data-path over the number of packets generated by the front-end. The system must obtain the highest $\eta_{readout}$ since image reconstruction is the main focus of pixel detectors, typical targets are in the $> 99.5\%$ range.

2. **Average Latency**: indicates the average number of cycles for a packet to travel from its source to the chip periphery.

3. **Maximum Latency**: the longest time a packet stays in the system before reaching the end of the data-path.

4. **Output channel occupancy**: how much of the output bandwidth is actually in use.

The $\eta_{readout}$ is critical to the main objective of read-out chip, which is collecting and transmitting packet data. The **latency** figures are used to define the bit width of the timestamping fields (i.e. the number of bits to represent the BX ID), so lower latencies allow for smaller buses and overall power and area savings. The **output occupancy** shows whether the off-chip links are saturated or oversized, to better estimate the required bandwidth.

Input analysis tools

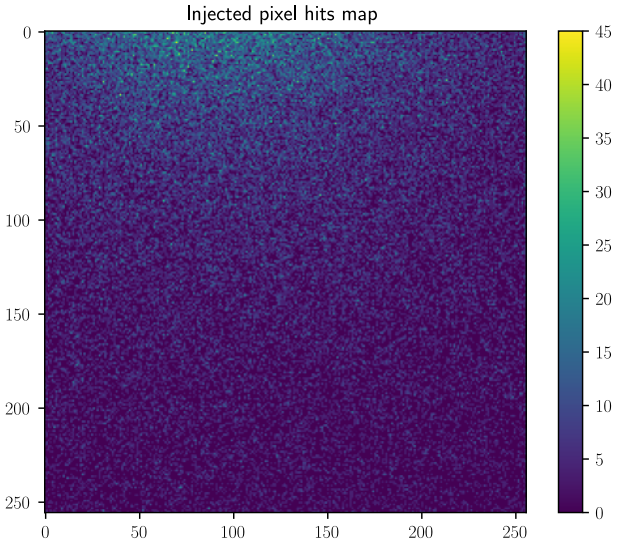


Figure 2.6: Pixel hits injected in the model, after front-end processing.

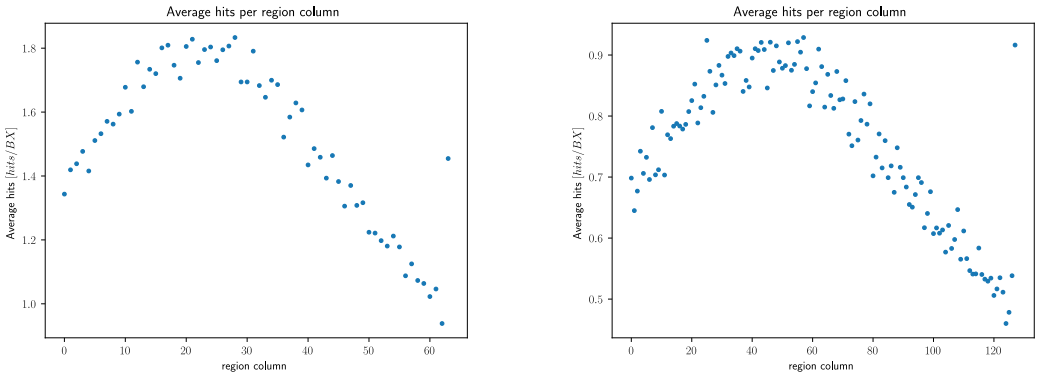
The analysis starts from the pixel hit files that are provided by the physicists running simulation of the experiments. Tools like **Allpix Squared** [12] are able to simulate the conditions inside a particle accelerator, and predict the result of high energy particle interactions with the pixel arrays, to generate pixel hit maps.

The pixel detectors are then designed to be able to reconstruct these images with satisfactory faithfulness.

Analysing these hit maps is the first step to understand what are the possible data reduction and rejection techniques that can be employed directly by the front-end, and which data rates should be targeted by the read-out infrastructure. As of now, the starting input for Pix-ESL are provided to us by the front-end designers (see Fig.2.6), but there are plans to simulate the filtering and clustering functions in SystemC or Python.

From these maps we can also estimate the necessary bandwidth: for example in Fig.2.7 the plot represents the average amount of pixel hits per column of the pixel matrix; since the column clock is the same as the BX rate (40 MHz) and the read-out is vertical, this means that each column can output at most 1 pk/BX, which is exceeded by few central columns in Fig.2.7a. It can be solved by doubling the amount of columns, as seen in Fig.2.7b.

These plots allow the designer to properly choose the number of columns or increase the clock before defining any of their functionality.



(a) **(b)**

Figure 2.7: Average pixel hits per column.

Figures of Merit extraction

The first three FoM listed in 2.4.2 are computed directly in the SystemC model, since they are simple scalars. The same FoM are also evaluated within a certain latency threshold, because packets that linger too long in the read-out data-path are effectively lost and harmful to the image reconstruction: a first packet with $BX_{ID,pk1} = t$ cannot be distinguished from a second packet with $BX_{ID,pk2} = t + 2^{N_{bits}}$, where t is any BX timestamp for an event, and N_{bits} is the bitwidth of

the BX timestamp. This causes aliasing since data belonging to a certain frame BX shows up in the next frame $BX + 2^{N_{bits}}$.

In the Python toolset the latency distribution can be plotted to find a suitable value for the maximum latency by looking at its tails, and anomalies can be found if the shape is too different from a Poisson or Gaussian curve.

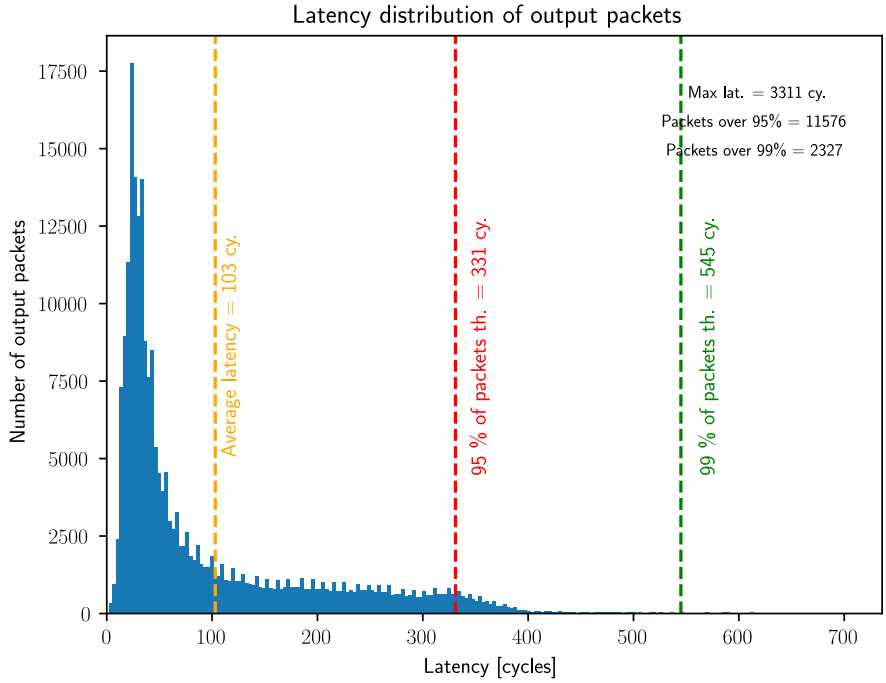
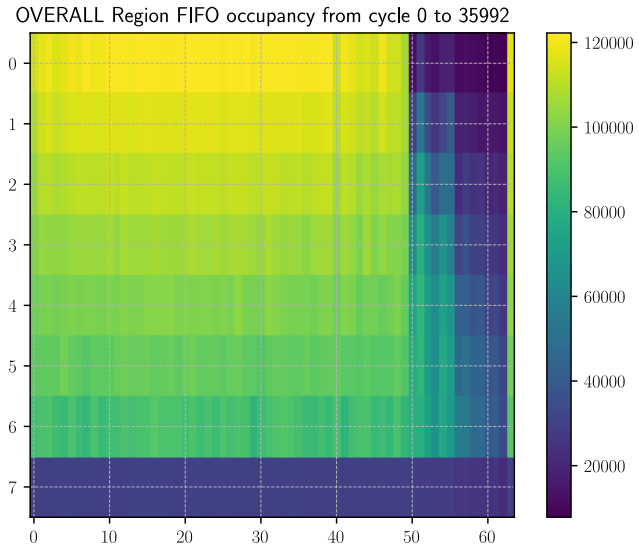


Figure 2.8: Latency distribution example.

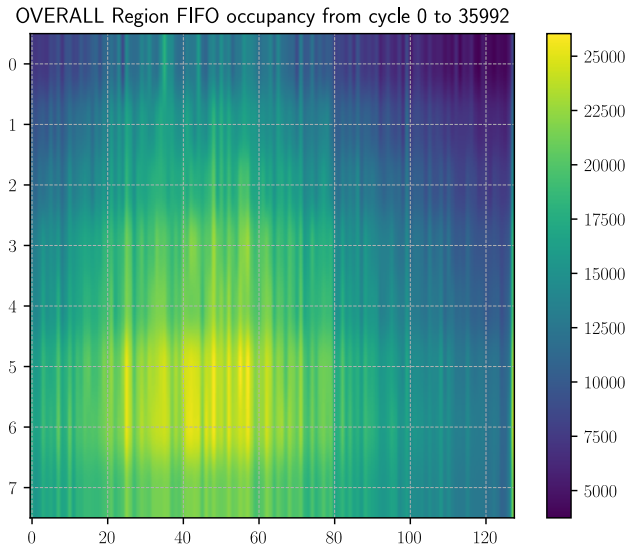
FIFO occupancy logging

The figures of merit listed previously are valid in judging the quality of a certain system, but to improve its performance we need find the causes of bottlenecks in the data-path. As mentioned earlier, these bottlenecks are caused by full FIFOs, which create a cascade of stoppages and ultimately data loss.

For this reason, we log the occupancy in each layer instance and create a map to spot the position and cause of these deadlocks. In Fig.2.9 we can see that the top and bottom configurations have a different amount of full FIFOs: the bottom one has double the amount of columns and optimized column arbitration, which result more even distribution in FIFO occupation and overall lower amount of full memories.



(a)



(b)

Figure 2.9: FIFO occupancy in the region layer for different configurations, full and deadlocked elements in yellow.

Chapter 3

Data-driven read-out: Velopix-2 case study

This chapter will present a real use case for the Pix-ESL framework to show its capabilities in design space exploration for a new data-driven (trigger-less) architecture.

3.1 Velopix2 overview

Velopix2 is a planned upgrade to the recently installed Velopix in the VErtex LOcator detector [2] assembly in the LHCb experiment. Velopix is the most important component of the VELO detector, the pixel detector that tracks particles entering the experiment. At the moment of writing, multiple teams are developing competing solutions to the Velopix2 specifications: INFN with their TIMESPOT ASIC and CERN's EP-ESE-ME with their demonstrator PicoPix.

This case study will try to be agnostic from either solution, as it focuses on the system level design, i.e. the number of elements in the read-out chain and their functionality; the INFN and CERN teams are focusing mostly on the challenges posed by high resolution of time measurements and other issues at the pixel/superpixel level, which have little relevance to system design.

3.1.1 LHCb and VELO

LHCb studies the matter-antimatter asymmetry by investigating b-quarks and their decay. The schematic view for the experiment can be found in Fig.3.1: it is clearly different from the other experiments at CERN like ATLAS (see Fig.1.1) or CMS, which are cylindrical volume detectors, where the particles of interest are the ones travelling almost perpendicular to the beam after the collision. Instead in

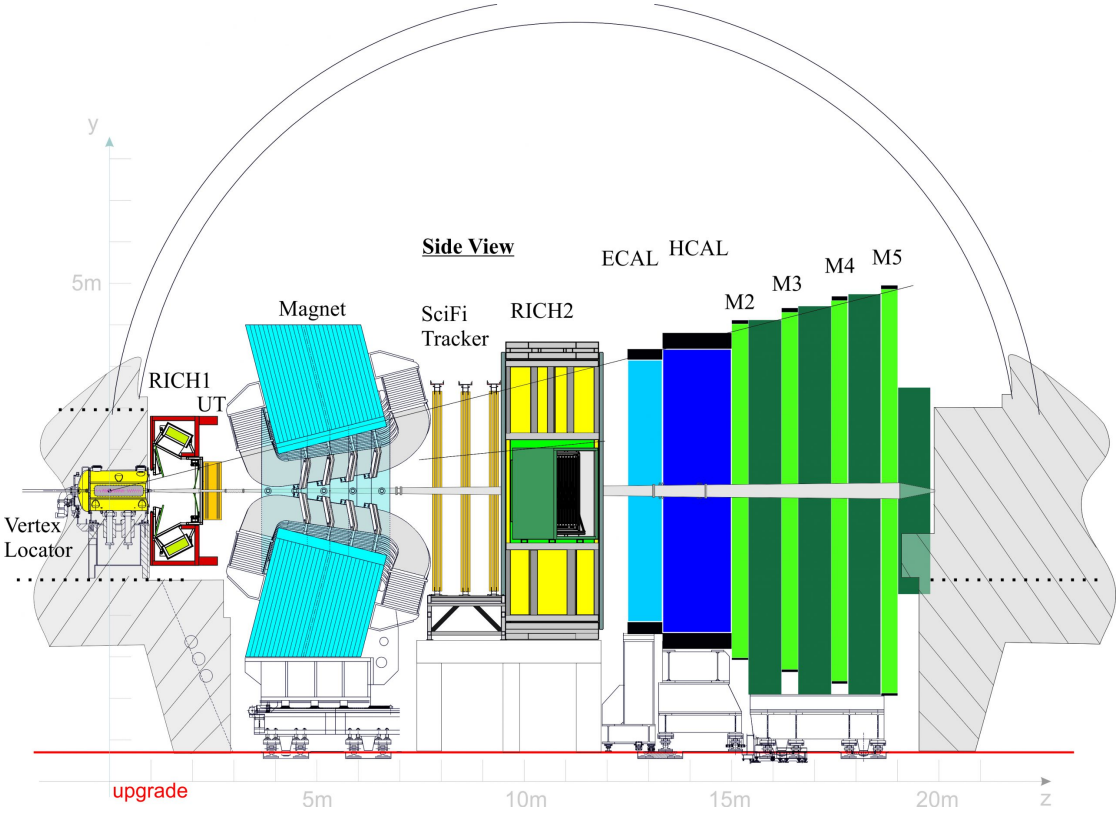


Figure 3.1: LHCb schematic view[13].

LHCb b-quarks particles and their decay byproducts are found mostly in a narrow cone along the beam axis, which is the reason for its peculiar design.

Looking at Fig.3.1, the beam cuts across the experiment horizontally and the collision takes place on the left side, where the VErtex LOcator detects and tracks the particles for the first time. The large magnets to the right of VELO bend the charged particles's trajectory, which are then detected by the SciFi (Scintillating Fibers) tracker with an altered path. This difference between the VELO and SciFi detectors measurements allows the determination of the particle charge. Mass values can be obtained by the calorimeters positioned after the trackers.

The VELO is the main tracking device before the magnet, and its data is also used in the trigger for the whole LHCb experiment, which decides whether to keep or discard data from collisions. It is composed by a number of modules placed on planes perpendicular to the beam, which can be retracted far away from the beam during its calibration and injection, and then moved as close as 7 mm during data-taking.

The goal of this upgrade to VELO is to allow for increased luminosity, thus increased data rates and radiation levels in the pixel detector.

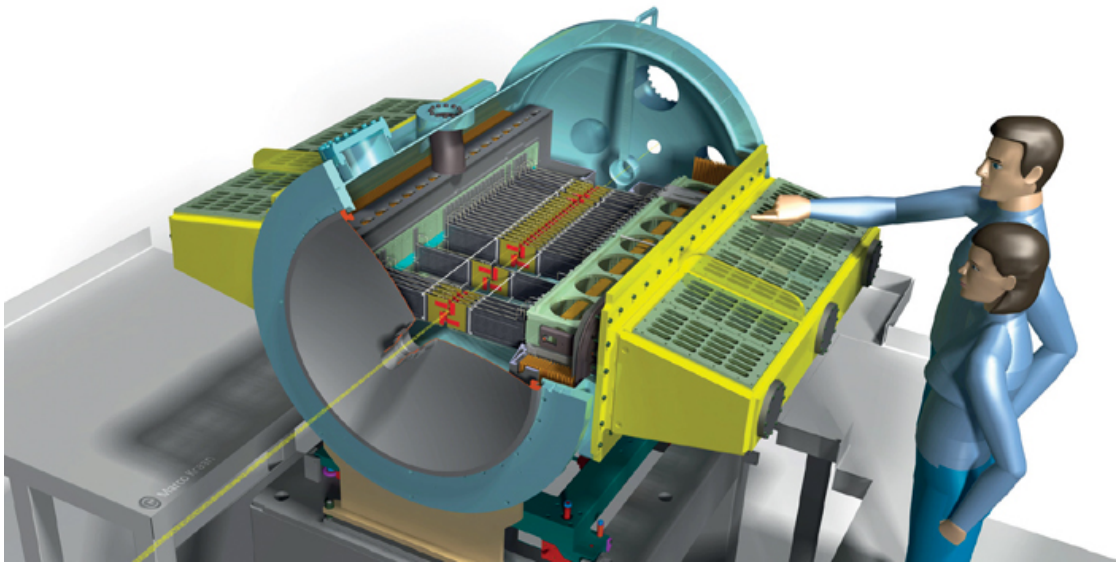


Figure 3.2: VELO upgrade cut-out.

3.1.2 Velopix and Velopix2 chips specifications

Velopix2 refers to the pixel detectors chips, formed by the pixel sensors and their read-out chips.

In Fig.3.2 we can spot in red, very close to the beam, the pixel detectors: they are mounted in an L-shape on modules that host power delivery, cooling, and connectivity (in yellow). The same boards can be moved aside to the left and right when not in use for data-taking.

Multiple Velopix chips are mounted on the same module, as shown in Fig.3.3, to cover a large area with relatively small chips. The Velopix chips shown in the figure are *4-side tileable*, meaning that they can be joined on any side with another Velopix chip to form larger assemblies.

Velopix itself was a significant upgrade over the original VELO detector, moving from a 1 MHz triggered non-zero-suppressed read-out, to a continuous 40 MHz zero-suppressed one[14]. This means that pixel hit data can be read out at the same frequency of the collision, without the need of an external trigger to select few interesting events. It is achieved thanks to a

Velopix2[15] requires much higher precision in the time measurements, from 25 ns in Velopix to < 30 ps resolution, and a leap in data rates from $10\,\mathrm{Gbit\,s^{-1}\,cm^{-2}}$ to $\sim 200\,\mathrm{Gbit\,s^{-1}\,cm^{-2}}$, due to the larger number of pixel hits and quantities measured per hit. These improvements require innovative techniques to measure time with such precision, and on-pixel hit clustering to reduce the amount of packets generated.

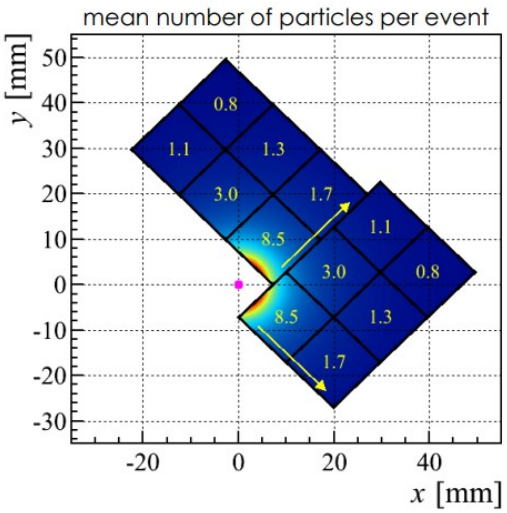


Figure 3.3: Velopix chips positioned in an L-shape around the beam.

The specifications include physics simulation data that contains examples of pixel hits over 1000 BXs. Analyzing the pixel hits allows the designer to determine rough requirements for the system's bandwidth across all layers.

Unfortunately, the amount of data packets injected in the read-out data-path depends on the clustering and filtering techniques applied in the pixels' front-ends, so we used the FE modeled by the CERN team and the filtered data provided by X. Llopart. This causes the data rates to be much lower than the pixel hit rates: in the worst case ~ 200 hits/BX are filtered down to ~ 70 pk/BX, which is the target occupancy for the read-out architecture presented in the next sections.

3.2 Velopix architecture

Velopix architecture can be defined as a **trigger-less, data-driven, column read-out**. *Trigger-less* refers to the absence of a trigger and continuous read-out: every packet generated by the FE is injected in the read-out chain. In other detectors with triggered architectures, the system is always in data-taking mode using a circular buffer but only some BXs are marked as relevant and read-out.

Data-driven means that zero-suppression is applied to the pixel data, such that only pixels hit by particles generate packets, which are tagged with the position of the pixel. On the other hand, non-zero-suppressed architectures require each pixel in a frame to be read-out, even though most of them do not contain particles hit data.

It is important to note that the terms top and bottom of a pixel detector do

not refer to the spatial orientation, but rather to the regions with the highest and lowest amount of radiation; i.e. the top is always the part mounted closest to the particle beam, and the bottom (or periphery) is where the more sensitive communication and elaboration ASICs are placed.

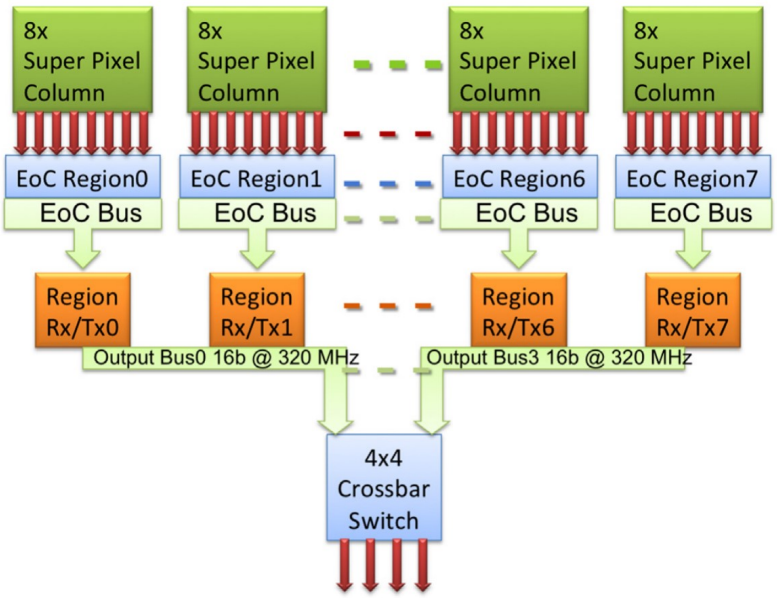


Figure 3.4: Velopix read-out chip architecture

Column read-out means that the pixel matrix is grouped vertically in a number of columns, formed by superpixels stacked on top of each other; packets are extracted at the bottom of each column, in the *end-of-column* regions. EoC are then grouped in 4 couples which share the output buses and are routed off-chip through a crossbar switch.

Each chip is connected to 4 GigaBit Transceivers (the *optical links* in Fig.3.5), and data from all 24 chips is transferred to a data acquisition system with FPGAs on PCIe or ATCA boards, called TELL40 due to their continuous read-out at 40 MHz. These boards take care of some elaboration, like sorting packets by BX and creating frames on FPGAs, and control the read-out chips timing and trigger.

3.2.1 Velopix2 layers

As mentioned previously, the read-out architectures has a hierarchy based on layers, which move data from the lowest layers to the periphery output channels. The compartmentalization is useful to have a system's view of the architecture and to decouple each layer's functions (arbitration, routing and transport). These are the

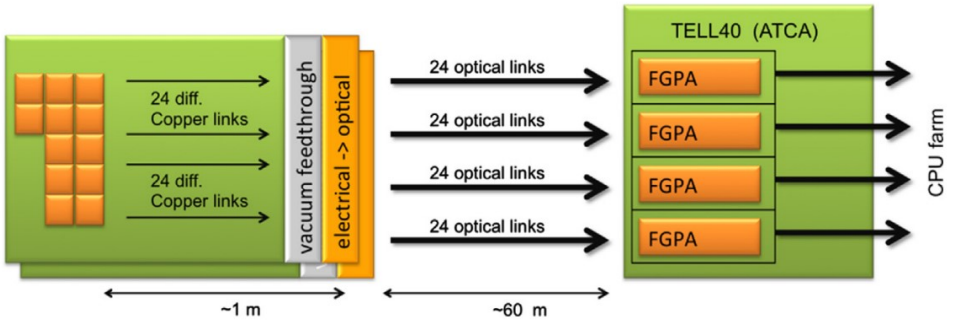


Figure 3.5: Velopix off-chip connections

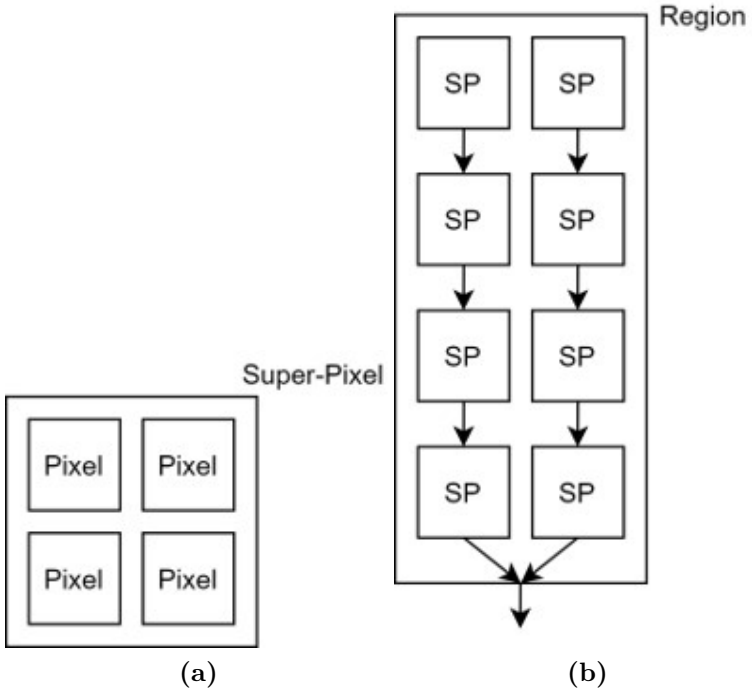


Figure 3.6: Pixels grouped in a Superpixel (a), superpixels grouped in a region (b)

layers used to model Velopix2, derived from the `base_layer` and `funct_layer`:

- **Pixels**: connected to the pixel sensors via bump-spots on top of the chip. Contain the analog front-end and discriminator.
- **Superpixels**(*SP*): contain 2x2 pixels and digital circuits to timestamp pixel hits, cluster or reject hits based on nearby pixels. Since data is clustered

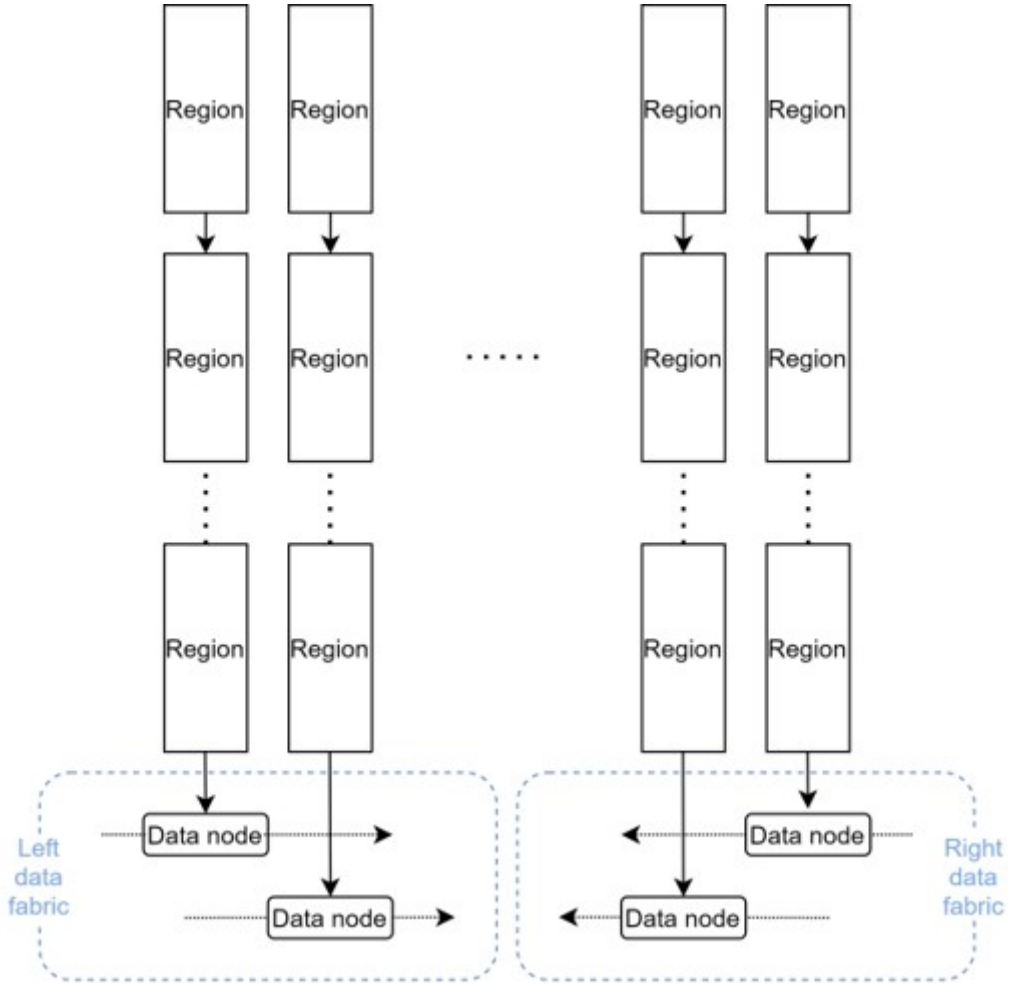


Figure 3.7: Regions connected in columns and with EoC data nodes

at the SP level, only SP are able to inject packets in the read-out chain. Typical packets include information about the pixel address (16 bit), bunch cross timestamp (BX ID), Time-of-Arrival (ToA) and Time-over-Threshold (ToT).

- **Regions**(*reg*): contain a set number of SPs connected with a mix of series (SP-to-SP) and parallel (SP-to-reg) links, as seen in the SPs forming the region in Fig.3.6b. The number of SPs columns and their length before connecting to the parent region are some of the design parameters that were explored to optimize the structure.
- **Datanodes**(*DN*): connected to the last region in a column, can receive packets from one DN and pass it to the next DN in one of the output channels,

which are formed by series connections of DNs. These modules have simpler arbitration and smaller FIFO because they are supposed to work at higher frequency than the rest of the read-out chain, and serialize the large amount of data incoming from the regions into fewer, high-speed channels.

Each layer's instances can be seen as an array with a certain number of columns and rows, whose values can be found in Table 3.1. The size of a given layer's array relative to the layer below determines the number and connection from the *children* to the *parent* modules: for example, a 128x128 superpixel array and a 32x64 region array result in regions that contain 2 SPs columns, each 4 elements long (as the region in Fig.3.6b).

Layer Name	Rows		Columns		FIFO size	
	Min.	Max	Min.	Max	Min.	Max
Pixel	256		256		N.A.	N.A.
Superpixel	128		128		2	8
Region	16	64	64	128	2	8
Datanodes	8	16	8	16	1	2

Table 3.1: Velopix design space parameters

3.2.2 Velopix network

The network in Velopix defines how each module instance is connected to other instances in the *sibling, children and parent* layers. The network is created by a set of functions that define the rules for connections, and are written to accommodate for varying dimensions in each layer. For example, the network function for SP-to-Region connections may connect only the bottom 2 SPs in a 2x2 sector to the region above, where the SP-to-SP network function will connect the top 2 modules in the 2x2 sector to the SPs below.

The connections are created so that each packet can only take one route to reach the output, and it comes from a conscious system level decision to try to limit the routing complexity in the physical implementation. This is a limitation that could be foregone to move towards a Network-on-Chip architecture, where modules can route data in multiple directions to overcome blockages: in fact if the pixel hits are unevenly distributed there may be local blockages in some columns, even if the total system bandwidth should support those data-rates at the global level, and allowing multiple routes would allow a fuller exploitation of the output bandwidth.

Arbitration functions in the Velopix2 model decide the order in which each module will send pull requests. In fact, a certain module may be connected to multiple other modules which it can request data from, and the arbiter in that

module chooses which source to select. In the current version, arbitration is static and programmed during the instantiation of each module. This means that the module will split its requests between the sources using predefined and static values. The sources can be other modules in the *sibling* layer or *children* modules.

3.3 Study results

The result of this study consists in an optimized configuration, with specific values for the number of rows, columns and FIFOs per layer. The various architectures have been ranked based on read-out efficiency, average latency and maximum latency, in this order. Here will be presented some meaningful configurations with some considerations about their performance.

3.3.1 Input data analysis

The input data is part of the specifications of the system, and being able to analyze it correctly will reduce significantly the design space. For example, in Velopix2 the worst case scenario has a very high occupancy, meaning that many of the superpixels in the system generate valid packets in each BX and the read-out chain is under high stress. The specific occupancy value is 72.3 pk/BX (see average value in Fig.3.8) and it sets a hard minimum value for the system bandwidth at any layer. For example, the communication from the matrix to the periphery is achieved by connecting the last region in each column to a datanode: it is clear how the number of columns will limit the data transfer between the 2 layers, which leads us to define a minimum amount of columns in the region array to at least 70 units. An example where this condition is unmet will be presented.

If the spatial hit distribution is not uniform some columns may have a hit rate higher than their bandwidth, that is set at 1 pk/BX/col if the matrix clock corresponds to the BX rate of 40 MHz. This can be estimated by extracting the hit probability per region column, as was done in Fig.2.7: the left plot shows how 64 region columns would result in many columns hit more than once per BX and subsequent congestion; on the other hand, 128 columns are just enough to maintain a hit rate below 1 pk/BX/col even in the busiest central columns.

3.3.2 Configurations and results

Out of the whole parameters space shown in Tab.3.1, here some significant configurations will be detailed. In Tab.3.2 are shown the configurations' parameters that were chosen as examples of design space exploration with Pix-ESL:

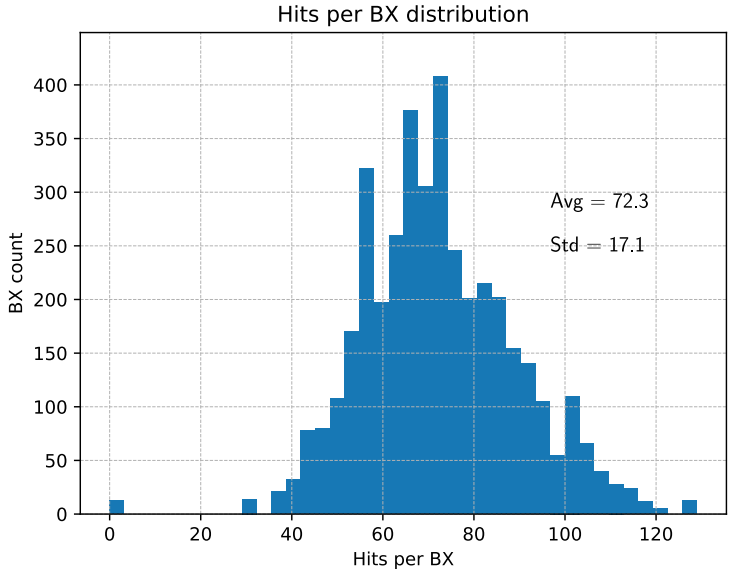


Figure 3.8: Distribution of the number of pixel hit per BX

Parameter Name	Configurations		
	C1	C2	C3
Pixels	256x256	256x256	256x256
Superpixels	128x128	128x128	128x128
SP FIFO (Int.|Ext.)	4|2	4|2	4|2
Regions	16x64	8x128	8x128
Reg. FIFO	4	4	4
Reg. Arbiter	Linear weights	Optimal weights	Optimal weights
Datanodes/EoC	64	128	128
DN FIFO	2	2	2
DN Clk Mult	8	8	8
Output Channels	8	8	16

Table 3.2: Parameters in the 3 configurations of interest.

- **C1**: originally developed for an occupation < 60 pk/BX, has only 64 region columns. Shows how

- **C2**: increased the matrix throughput by using 128 columns, but kept the periphery throughput at 64 pk/BX.

- **C3**: best solution yet, with 128 pk/BX bandwidth in both periphery and matrix. Very promising efficiency and latency, adequate for on-chip processing.

Result	C1	C2	C3
Read-out Efficiency	82.6%	84.0%	99.5%
Average Latency	119 cy.	100 cy.	21 cy.
Max Latency	>3000 cy.	>2800 cy.	440 cy.

Table 3.3: Results from the 3 configurations

C1: insufficient matrix bandwidth

The first configuration was chosen because it shows how an insufficient matrix bandwidth causes excessive pile-up along the columns that are hit more than once per BX. In Fig.3.9 the left plot shows how most of the region columns exceed the matrix bandwidth, and causes most of the FIFO columns to saturate and stop the data-flow. Interestingly, the few columns below the 1 pk/BX are rarely blocked.

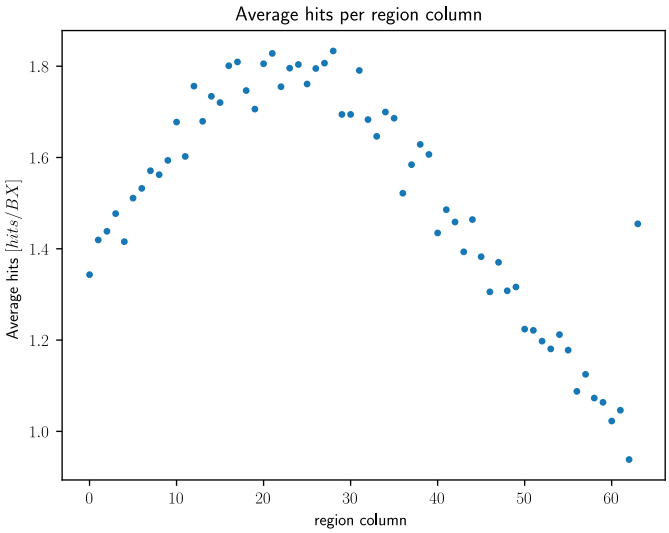
These considerations back up the outputs' metrics, which show a long-tailed latency distribution resulting from the packets stuck in the matrix for many cycles, and a high utilization of the output bandwidth. Even though the periphery has an insufficient bandwidth, in this configuration the bottleneck is still the matrix with its 64 region columns.

C2: insufficient periphery bandwidth

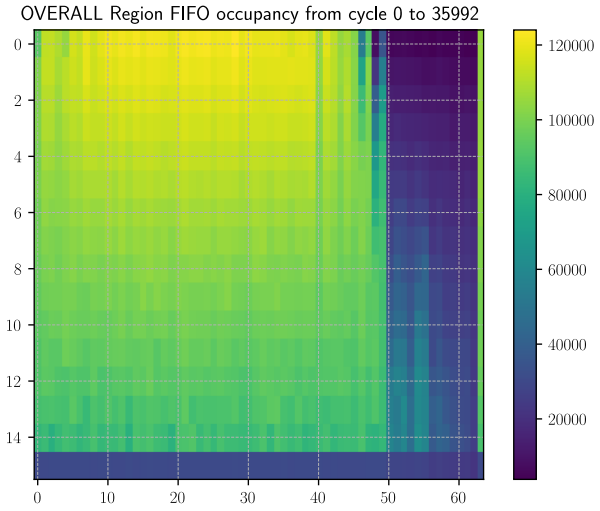
This configuration aims to solve the previous setup's issues by splitting each column in 2 and increasing the matrix bandwidth. This configuration is of interest because the periphery bandwidth has not been scaled up accordingly: the datanodes are arranged in 8 output channels running at 8x the matrix clock (for a total of 64 pk/BX as before), but each of those channels has been extended to be 16 DNs long, instead of 8 as in C1. This means if that the matrix is running at full bandwidth of 128 pk/BX, the periphery will be able to extract only half of that. The results is a pile-up at the bottom of each external column, since the DNs prioritize the read-out of the central columns.

In Fig.3.11b only half of the regions can be read-out efficiently due to the limited periphery bandwidth: the 2 possible solutions are either rearranging the 128 datanodes in 16 channels and increasing the output parallelism, or doubling the periphery clock to 16x the BX rate.

Fig.3.12a shows that compared to C1 the amount of packets with a low latency is increased, but at the same time the right tail is more extended, due to the fact that the external columns are read much less often than before.



(a)

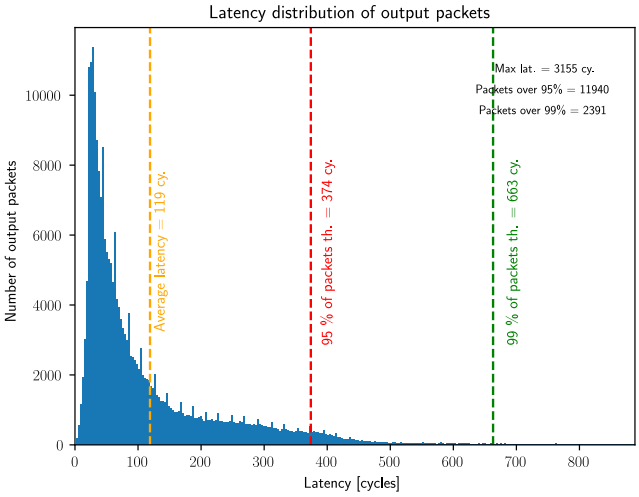


(b)

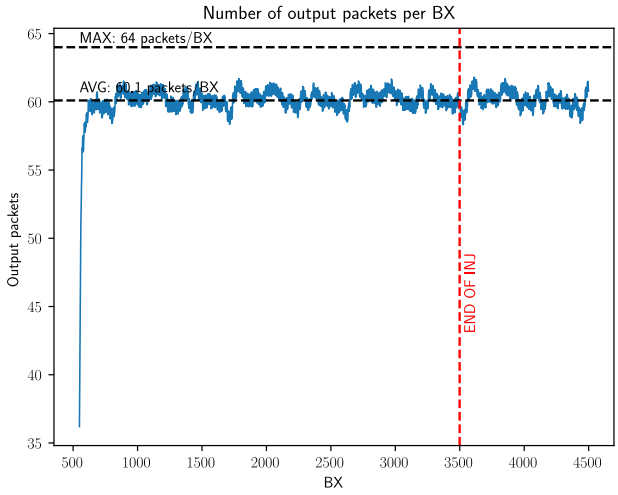
Figure 3.9: C1: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region].

C3: specifications satisfied

The next natural step was to increase the periphery throughput, which was brought in line with the matrix to 128 pk/BX by doubling the amount of output channels.



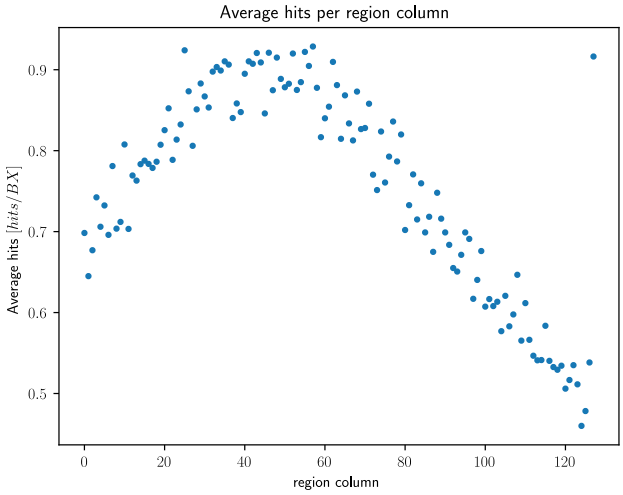
(a)



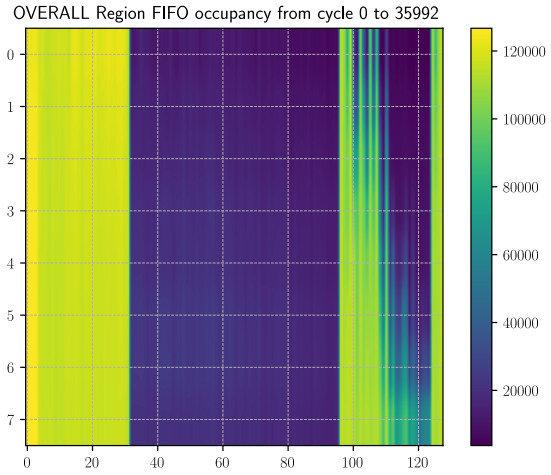
(b)

Figure 3.10: C1: Packet latency distribution (a), output channel utilization (b).

As shown in Tab.3.3 and by Fig.3.14b, the output rate averages a value very close to the input rate of 72.3 pk/BX. This is largely thanks to the oversized bandwidth of the system, which can not be scaled up so easily in the off-chip datalinks due to power concerns. In fact, this bandwidth would result in $> 200\,\mathrm{Gbit\,s^{-1}}$ and would require 8 of the datalinks developed at CERN, each carrying $> 25.6\,\mathrm{Gbit\,s^{-1}}$ and consuming in total from 800 mW to 1,600 mW.



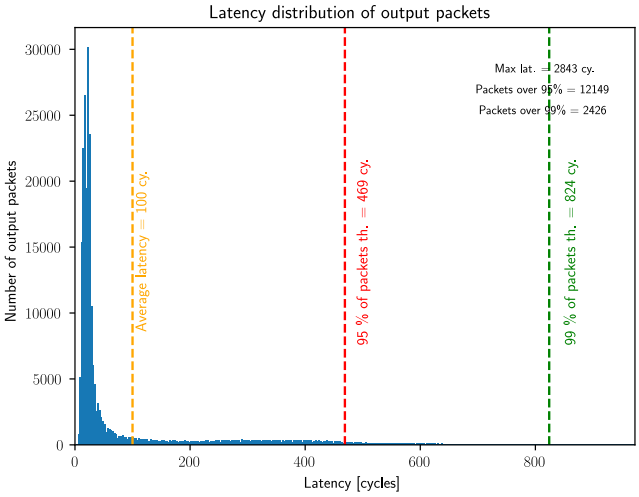
(a)



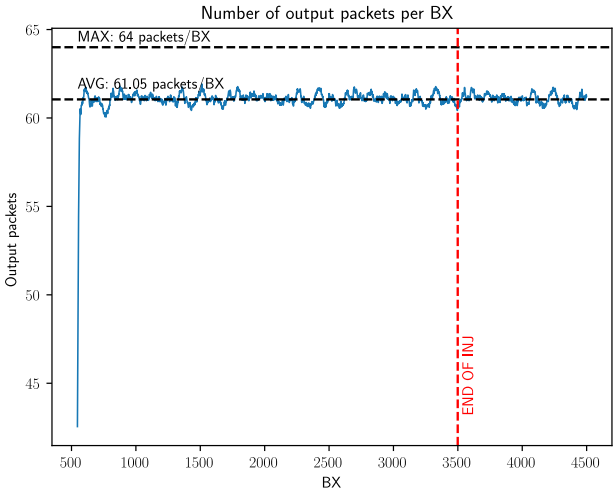
(b)

Figure 3.11: C2: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region].

In this setup, we also optimized the arbitration weights given to the region elements along each column based on the hit probability of the pixels inside each region. This should make sure that the fraction of time allocated to read-out each region corresponds to its chance of having a pixel hit, thus decreasing dead times when pull requests are issued to empty regions and superpixels. Its effect in C3



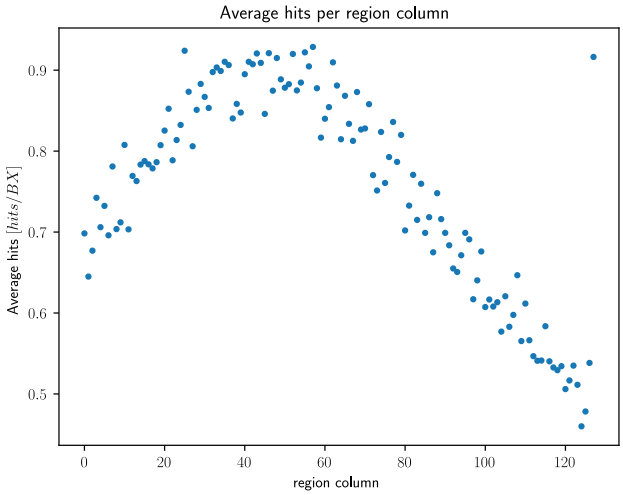
(a)



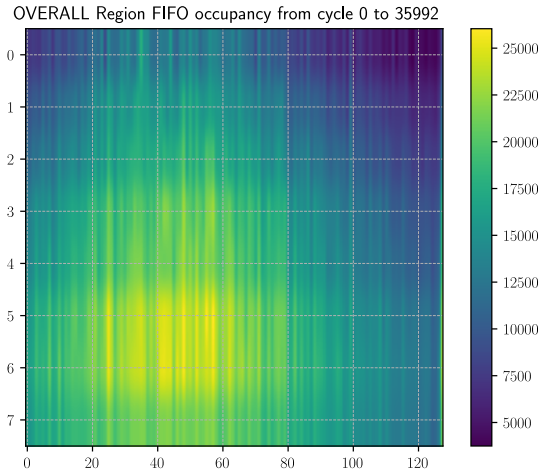
(b)

Figure 3.12: C2: Packet latency distribution (a), output channel utilization (b).

can be seen in its region FIFO occupancy (see Fig.3.13b), which is more evenly distributed than C1 (see Fig.3.9b, higher occupancy in the top regions).

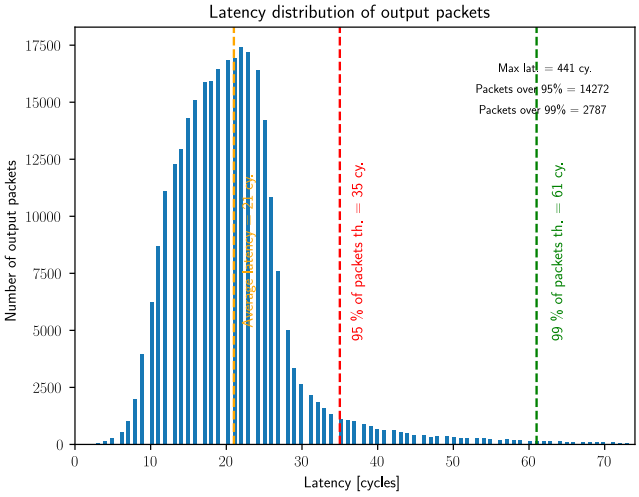


(a)

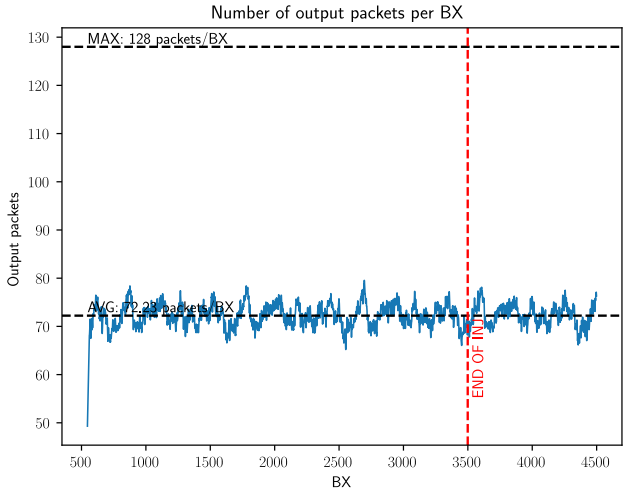


(b)

Figure 3.13: C3: comparison between hit probability (a) and congestion in the region FIFO map (b)[colorbar indicates number of packets written to a region].



(a)



(b)

Figure 3.14: C3: Packet latency distribution (a), output channel utilization (b).

Chapter 4

On chip Sort&Bin: from prototyping to implementation

This chapter proposes a module, named Sort&Bin, for ordering and grouping data packets from the same physics event on pixel read-out chips. It is useful in data-driven and zero-suppressed architecture such as Velopix2, where output packets are read out-of-order and require sorting and grouping off-chip. The Sort&Bin module instantiates memories at the periphery of the chip to accumulate data packets over a period of time and build one large multiple-packet frame per physics event.

This development is an enabler for further data processing by concentrating the entire physics event in a single location. In addition, it simplifies the back-end architecture by providing already ordered and grouped data. Finally, it allows us to study the design flow from prototyping in Pix-ESL to the RTL description and its physical implementation to investigate the flow from a SystemC model to a physical-level design.

4.1 On-chip processing advantages

Currently, the biggest challenges with read-out chips are their large bandwidths ($\sim 200\,\mathrm{Gbit\,s^{-1}}$), high power consumption from data transmission and off-chip processing, and the complexity of working in a variable-latency system where packets injected at the same time reach the end of the read-out at different times.

The Sort&Bin module solves these by :

- reducing the required bandwidth since grouped packets (data-frames) only need an event identifier (`BX_ID`) to be transmitted once per frame.

- saving the computing cost of sorting the packets on FPGAs.
- transforming the read-out chain in a fixed latency system, allowing further on-chip processing with frames instead of loose packets.

4.2 Architecture

The Sort&Bin module gathers data from all the output channels of a read-out chip. In the Velopix2 architecture, it connects at the data nodes and run at the same clock frequency of 320 MHz. It takes as input 8 channels, each carrying 1 pk/cycle with its `BX_ID`, and outputs at 8 pk/cycle a variable amount of packets, forming a data frame containing all the packets from the same `BX_ID`.

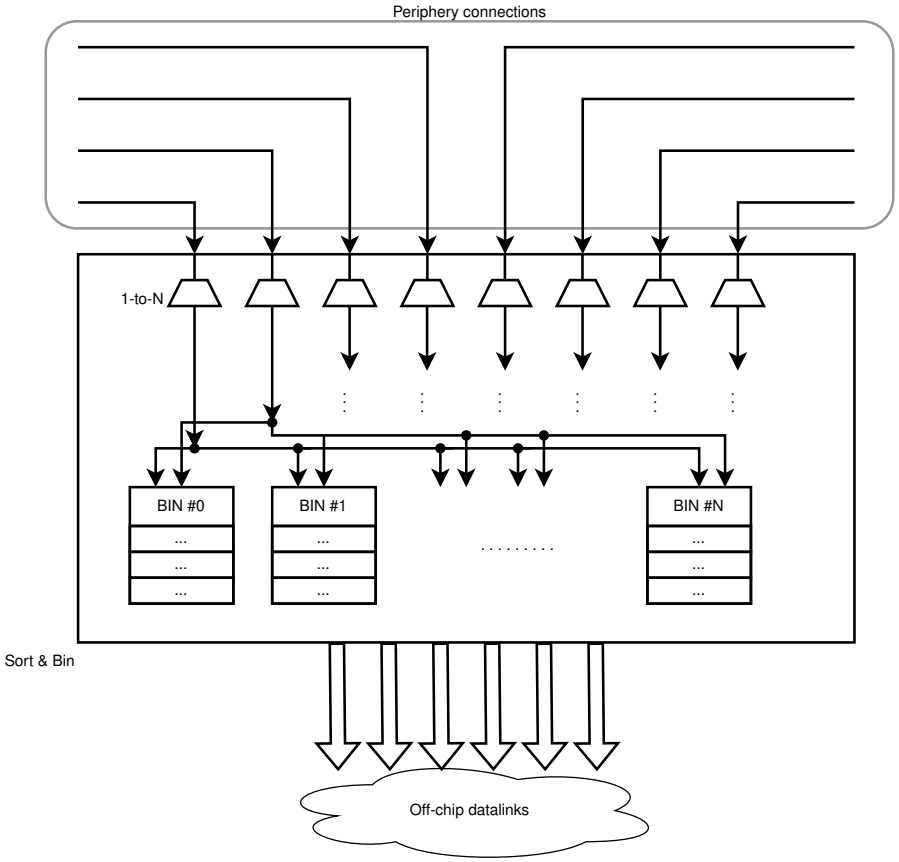


Figure 4.1: Sort&Bin module schematic view.

Ideally, the module fixes the packet latency by accumulating packet data over a certain time window in its memory: incoming packets are written inside *bins*, each storing packet groups with the same `BX_ID` field until a certain time has

elapsed, thus fixing the whole frame latency to this time period. Afterwards, the bin reaches a time-out condition and content is sent to the Sort&Bin output.

This time-out latency is defined by the amount of bins in the design minus the time needed to read out the bin itself, which can be longer than one BX (i.e 8 cycles) if more than 64 packets are stored inside it. After a bin is timed out and emptied, it is ready to receive data from the newest BX. Packets that reach the Sort&Bin module after their BX was timed out are considered *out-of-order* and are not saved into any bin but instead immediately sent to an alternative output channel.

Since all the input data need to be treated in parallel, in the worst case where all the incoming packets originate from the same BX all of them are routed and written in the same bin. This means that each bin must have the same read and write throughput as the whole system, corresponding to 8 pk/cycle.

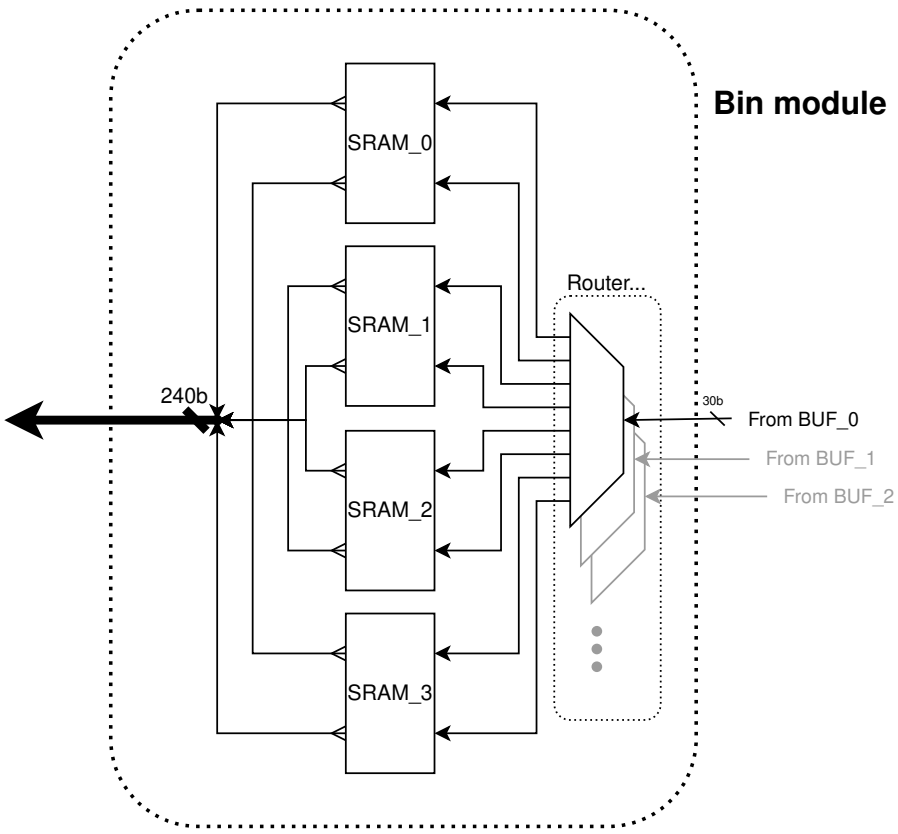


Figure 4.2: Bin sub-module schematic view.

Multiple Dual-Port SRAM banks were used in each bin, to achieve a total of 8 ports, each able to read or write a single reduced packet. Another feature that was implemented was a simple *load balancing* algorithm to evenly fill the SRAM banks

so that the bin could be read in the shortest time possible. This requires an 8x8 router that connects any input channel to any SRAM port in the bin and controls the 4 SRAM banks as FIFO allowing up to 8 parallel write operations.

4.3 Design flow

The design approach was to create a high-level prototype in SystemC to obtain values for the design parameters, such as the number of bins and their size, and then try to design it in RTL. This allows the prototype's parameters to be optimized to reach a sufficient level of performance from a functional point of view. Then, the design with those parameters can be implemented in SystemVerilog and verified. The final step is synthesis and physical implementation, which were carried out with Cadence EDA tools with the design flow provided by CERN's ASIC Support group.

4.3.1 Pix-ESL modeling

The SystemC Sort&Bin module was derived from the `funct_layer` like the read-out modules. The main difference was that instead of just one FIFO element, in this case, there were as many FIFO memories as the number of bins, with each one of them corresponding to one bin: in this level of abstraction FIFOs are accessible multiple times per cycle if *non-blocking* methods are used, thus there is no need to instantiate 4 SRAMs like in the RTL schematic in Fig.4.2.

Similarly, instead of using a cycle-accurate description for the insertion and read-out of the bins, delays were calculated based on the number of operations, and then module activity was paused for the length of the delay instead of simulating each cycle. The bin read-out process, for example, was executed instantly when the bin was timed out, and the delay was calculated as in Eq.4.1, where $N_{\text{pk in FIFO}}$ is the number of packets that need to be read from the bin, $N_{\text{out channels}}$ is the module's output parallelism and T_{clk} is the clock period.

$$T_{read-out} = \frac{N_{\text{pk in FIFO}}}{N_{\text{out channels}}} \cdot T_{\text{clk}} \tag{4.1}$$

Design space exploration

The SystemC design had two main parameters and two main metrics: the number of bins and the number of packets stored per bin, the grouping efficiency, and system latency. The two metrics respectively evaluate the percentage of packets that can be grouped into bins within the accumulation time and the total packet latency with the added time spent in the bins.

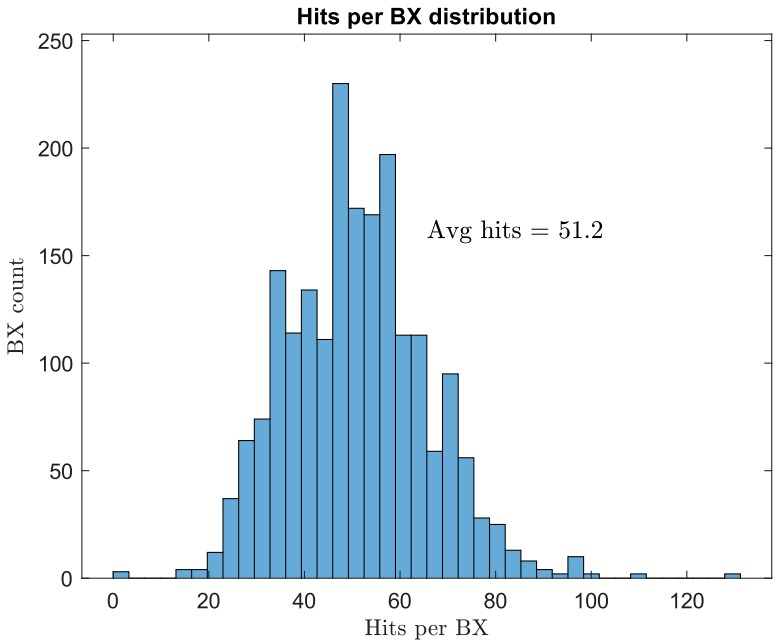


Figure 4.3: Frame size distribution.

The bin size must be large enough to store as many packets as the largest frames, whose size distribution is shown in Fig.4.3: 128 packets should be enough for most frames, given that the few very large frames above that value will most likely have some packets with extreme latency, that could not be binned anyway.

This value may seem too large given that very few frames exceed 100 packets, but given that we must use 4 SRAMs, as explained before, and that the foundry specifications indicate that each bank must contain at least 32 words, 128 packets is also the minimum implementable bin size.

We set the target grouping efficiency to $> 90\%$, and looking at the latency distribution in Fig.4.4, it would seem that most of the packets reached the module within 60 to 70 BXs. This was confirmed by running a parametric sweep on the bin number (see Fig.4.5), where the grouping efficiency curve resembles the integral function of the previous plot. The number of bins was set to 64 as it satisfied our requirement for $> 90\%$ grouping.

4.3.2 RTL design

The RTL design needed to convert the simplified and functionally defined SystemC module into a synthesizable RTL design. Out of the supported language by our synthesis tool (Cadence Genus), the choice fell on SystemVerilog because it

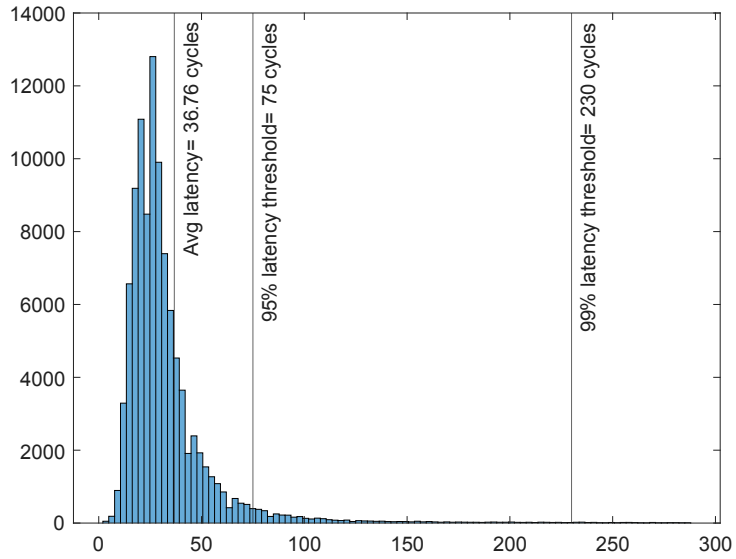


Figure 4.4: Packet latency distribution at the Sort&Bin input.

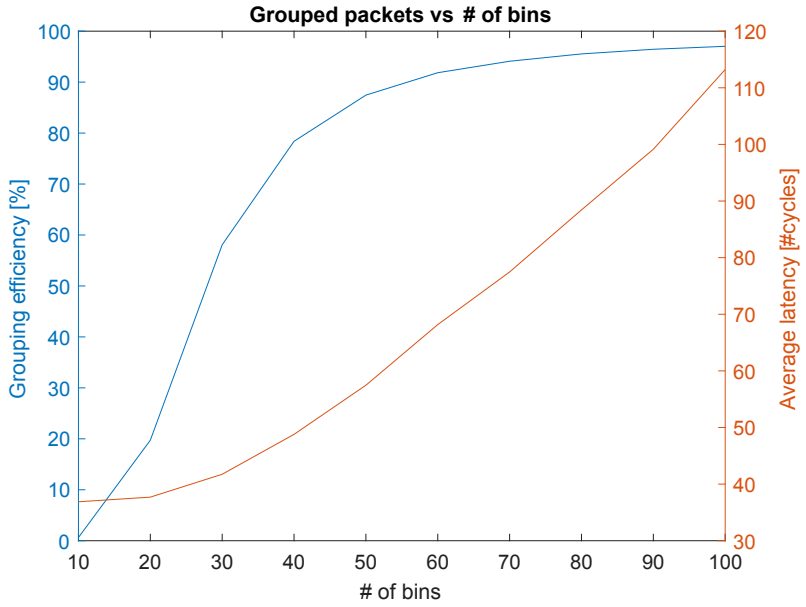


Figure 4.5: Effects of the number of bins on grouping efficiency (left axis) and total latency (right).

supported some more modern features, for example multiple-dimension port arrays.

The design also included a Verilog model of the SRAM modules provided by the vendor, compiled with the correct word length and number of locations. Since the shortest word length was 32 bit and the reduced packet size was only 30 bit, the extra bits could implement some error checking and/or correction, which is needed to mitigate the radiation effects on the memories. The SRAM macro size was 250 µm · 30 µm ($h \cdot w$).

Top module: Sort&Bin

The top module describes the Sort&Bin module itself, which contains a counter for the `BX_ID`, a controller sub-module to decide where packets would be routed by sending `DATA_ENABLE` signals to the bins, an $8 \cdot 40$ bit register array to buffer the input data, and 64 bins.

The routing from the input buffer to the bins is a simple fan-out, where each channel's data (except for the `BX_ID`) is connected to every bin, while the controller uses the `BX_ID` to determine which bins will catch the packets from the channels.

The top module's *grouped* output multiplexes the outputs from all bins to select the bin that is currently being read out, whereas the *out-of-order* output receives all the packets stored in the input buffer that exceed the maximum latency.

Sub-module: Bin

The bin is composed of 4 SRAM banks, and its main task is to distribute evenly the incoming packets to the 8 SRAM ports. The read and write operations are controlled by a simple FSM, which resets the write and read pointers, enables writes during the accumulation phase, and after the time-out, waits for the previous bin to finish reading its content before occupying the output channels.

This is necessary because every 8 clock cycles (at 320 MHz), one BX passes, and a new bin reaches time-out, but it can not be guaranteed that the previous bin has ended its read-out phase, since some bins may contain more than 64 packets, that can be read to 8 channels during those 8 cycles. Statistically, most data-frames should take less than 8 cycles to be read out, but to guarantee that any bin can be emptied before data from a new BX reaches it, the actual timeout latency (or accumulation period) lasts only for 62 bunch crossings, with the remaining 2 being used for read-out and some slack if the previous bins were late.

4.3.3 Implementation in a 28 nm technology

The synthesis and implementation steps transform the RTL design first into a netlist mapped to the technology's gates and then place and route all the cells' and

macro elements' instances to obtain a physical layout. At this stage, it is possible to extract metrics on the timing, area, and power costs of the design to evaluate whether the goals of the module were achieved and the constraints were satisfied.

The whole flow was carried out using the "CERN Digital Implementation Flow" for 28 nm, provided by the ASIC Support group. This tool simplifies the flow with ready-use TCL scripts for each step and a manager script.

Constraints

Here are listed the constraints for the physical implementation:

- **Timing**: the module clock is set to 320 MHz, corresponding to a period of 3.125 ns. The input and output delays were set to 0.5 ns.
- **Area**: the module position was set to the bottom of the read-out chip, where an area 12 mm wide and 2 mm tall can be spared for processing. The die area was specified to $2.6 \cdot 1.4\,\mathrm{mm}^2$, with most of it taken up by the memory macros.
- **Power**: the goal was to save power compared to the data-link needed to transmit the extra bandwidth without the module, which corresponds to around 150 mW.
- **Library**: the foundry sets a limit of 4 different transistor flavours out of all the available threshold voltages. As a first implementation run only the standard V_{th} library was used, with plans to add more in case of timing or power issues.

Synthesis

The synthesis's objective is to extract a standard cell netlist from the SystemVerilog description. This is achieved by first transforming the RTL logic into a generic set of logic functions, then this set is then mapped onto the logic functions provided by the foundry's cell library. At the end, further optimizations by the synthesis tool can recover power and area consumption.

The netlist obtained is a Verilog design that can linked to the Verilog model of the standard cells, and then verified before moving to the actual implementation steps. Some functional bugs in the RTL were corrected and un-synthesizable code was changed during this phase.

Floorplan

In this step, the module area was defined, the position of macro-blocks and pins was fixed, and power nets were added using TCL scripts.

As can be seen in Fig.4.6, input packets are entering from the top, *ordered* packets are output on the bottom, and *out-of-order* packets are on the right. This version of the module is non-configurable and only has *clock* and *reset* control signals on the left of its perimeter.

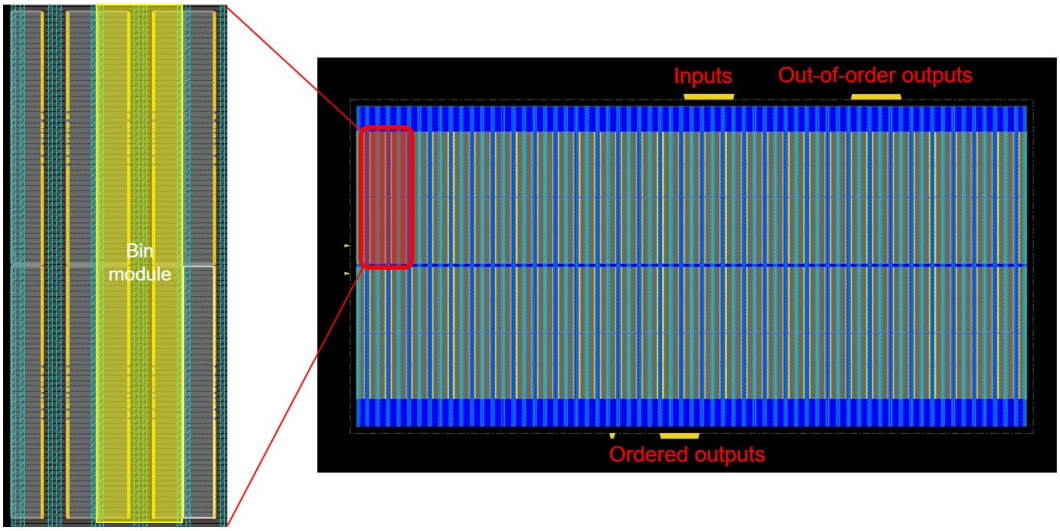


Figure 4.6: Sort&Bin floorplan with bin sub-module detail

The 256 SRAM macros were placed in a 4x64 array: each 2x2 quadruple of blocks forms a bin, with some space left in between the left and right side to place the sub-module logic. Placement blockages were set around each macro to avoid routing congestion on the SRAM pins.

Power nets were run vertically in stripes on the M8 layer on top of the SRAM between 2 different bins and above the space reserved for the bin logic.

Place, Clock Tree Synthesis & Route

Since the next steps were already automated and required almost no input from the user, they will be described shortly in order:

1. **Place**: the cell instances from the netlist were assigned positions on the layout based on the nets they are connected to and timing estimates

2. **Clock Tree Synthesis**(CTS): the clock distribution net was synthesized to minimize the skew between sequential elements.

3. **Route**: nets are transformed in physical wires connecting cells and are optimized to satisfy timing constraints in an iterative process. During this

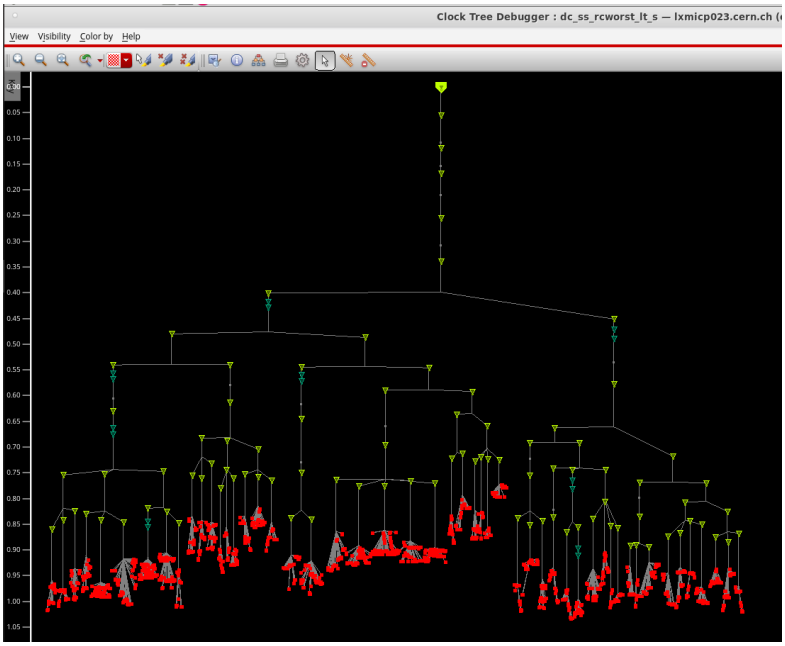


Figure 4.7: Clock tree delays

optimization, buffers may be inserted in some paths, or cells may be moved in order to remove timing violations, especially *hold* violations.

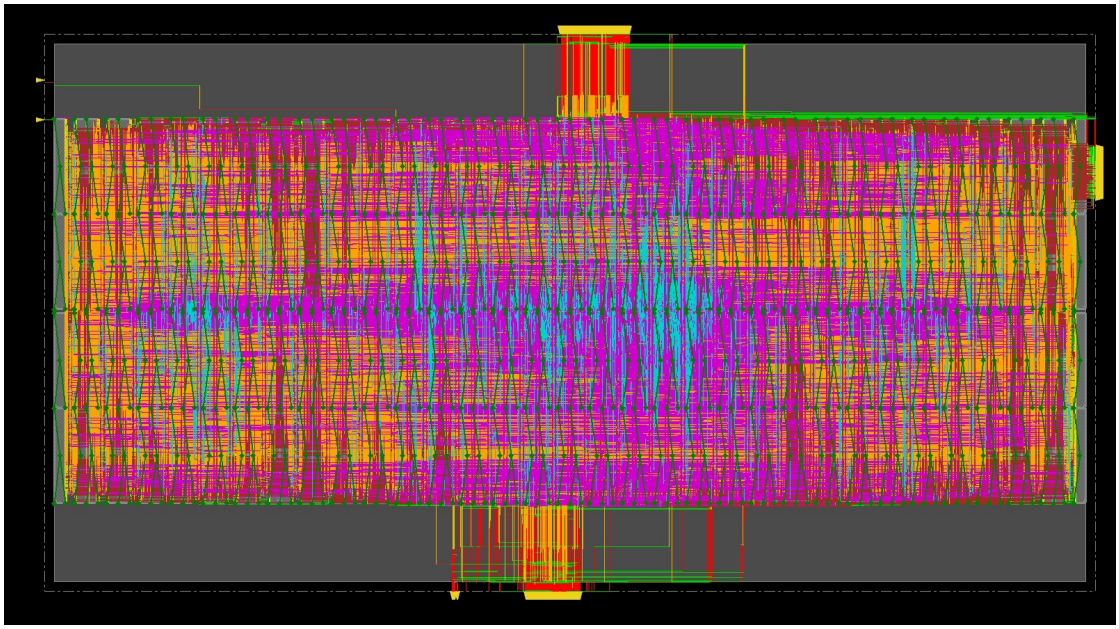


Figure 4.8: Layout after routing

4.4 Results

The three main figures of merit to estimate the usefulness and feasibility of this module were the efficiency of the algorithm in RTL, the area occupied, the power consumption and whether this design could be routed. The grouping efficiency has to be high enough to significantly reduce the load on sorting FPGA in the data acquisition boards, the area should be around a few square millimeters, and the power should not exceed the power consumption of the off-chip data links and the overall budget for the chip.

4.4.1 Verification

The Sort&Bin RTL was tested with a SystemVerilog testbench composed of a generator, to read input files containing packet data from Pix-ESL simulations, a driver, to inject said packets in the DUT, a monitor, to observe the transactions on the DUT, and a scoreboard to check the correctness of those transactions. This testbench takes the same stimuli as the SystemC model, allowing a direct comparison between the performance of the high-level and RTL descriptions.

The grouping efficiency of the RTL design, with the number of bins fixed at 64, reached 92.5%, matching the result expected from previous design space exploration with this parameter value, see Fig.4.5. In a future implementation, the scoreboard could co-simulate the SystemC description as the golden reference of the module.

The bandwidth requirement could be reduced by $\sim 22.5\%$ thanks this result, since the `BX_ID`, accounting for 25% of the packet size, would be specified for less than 10% of the transactions.

The post-layout simulation was carried out with the same testbench, to verify the functional integrity of the physical implementation, that showed $\sim 5\%$ of the output packets to be incorrect, dropping the grouping efficiency to 87.9%. Investigations are under way to determine and solve the causes of this issue.

Listing 4.1: Pre-implementation RTL verification results

```
[ SCOREBOARD ]
[ SCOREBOARD ] ------------------------------------------------------
[ SCOREBOARD ] -- Transaction report                  No.      % --
[ SCOREBOARD ] -- #Input trans:                     100000    100 --
[ SCOREBOARD ] -- #Ordered trans:                    92516   92.5 --
[ SCOREBOARD ] -- #Out-of-order trans:                7484    7.5 --
[ SCOREBOARD ] -- #MISSING trans:                        0    0.0 --
[ SCOREBOARD ] ------------------------------------------------------
[ SCOREBOARD ]
```

Listing 4.2: Post-layout verification results

```
[ SCOREBOARD ] ------------------------------------------------------
[ SCOREBOARD ] -- Transaction report                    No.      % --
[ SCOREBOARD ] -- #Input trans:                       100000   100 --
[ SCOREBOARD ] -- #Ordered trans:                      87894  87.9 --
[ SCOREBOARD ] -- #Out-of-order trans:                  6805   6.8 --
[ SCOREBOARD ] -- #MISSING trans:                       5301   5.3 --
[ SCOREBOARD ] ------------------------------------------------------
```

4.4.2 Implementation

As a proof-of-concept design, the results are indicative of the achievable performance but should be taken qualitatively. Nonetheless, the design was successful in its main objectives:

- estimating the **area** to 3.6 mm^2, that fits in the read-out chip periphery.
- estimating the **power** consumption to be around 50 mW, lower than the power consumed by the data-links that it would substitute ($\sim$ 100 mW/link). Since the Sort&Bin module saves a percentage on the bandwidth, chips with higher hit rates would benefit the most in terms of data-links saved. This result was obtained by simulating the Verilog netlist with real input data, and then backtracing its activity to precisely estimate with Voltus the power consumption of each net.
- confirming the **routability** of the design.

4.5 Next steps

Given the positive results, in the future this design will be refined and may be the stepping stone for the integration of on-chip processing. Some additional features are planned:

- solve the **inconsistencies** between RTL and post-layout simulations, rearranging the RTL design or adding low V_{th} cell libraries.
- **radiation hardening** studies must be carried out. To reduce radiation effects, logic can be triplicated with minimal area and power losses and error checking and/or correction can be added to memory.
- the module should be **configurable** , to reuse it in chips with different parameters and sizes, and **programmable**, to change some functionalities on the fly or shut it down.
- add further **on-chip processing** with ad-hoc cores or in-memory/near-memory computing.

Conclusions and future plans

This chapter presents the conclusions of this thesis on Pix-ESL and SystemC modeling for read-out architectures,on the design space exploration for Velopix2, and on the Sort&Bin periphery module.

Pix-ESL

In this thesis a new approach to particle detector systems design was detailed, proposing a unified framework for system-level architectural exploration and prototyping based on a SystemC core and a Python analysis toolset. With this approach, the first step in particle detectors is designing the whole system, to determine each module's requirements and identify bottlenecks. At the time of writing, the framework can be used to build a model for read-out chips, with description granularity at the pixel level but flexibility and run-time sufficient for quick exploration of large systems.

Its features include real physics data stimuli, configurability, TLM 2.0 support, data logging and statistics inside the model, plus an external Python analysis and plotting toolset, support for processing modules, memory occupation statistics and resource consumption estimates. The roadmap for Pix-ESL plans to expand the types of modules provided, refine the TLM interfaces and model other architectures, such as triggered read-out.

Velopix2 read-out prototype

Pix-ESL is being used in Velopix2 studies to determine the optimal read-out architecture and parameters that define the amount of data-path elements, their placement and connections, the memory sizing, the routing and arbitration functions. An optimal design corresponds to the architecture and its parameter set maximising the read-out efficiency while minimising the latency and amount of data-path and memory elements.

The simulations were able to find bottlenecks in the bandwidth, which were solved in the periphery by increasing the clock frequency and in the pixel matrix

by doubling the amount of parallel columns. Momentary surges in pixel hit-rate were smoothed out with appropriately sized FIFO buffers.

Due to the high pixel hit-rate, around $2.5\,\mathrm{Gpk\,s^{-1}}$ are generated in Velopix2 to achieve the target read-out efficiency of 99.5%, that result in a bandwidth requirement of $100\,\mathrm{Gbit\,s^{-1}}$. This throughput can be provided by multiple $25\,\mathrm{Gbit\,s^{-1}}$ optical data links developed at CERN, but it comes with higher power consumption and increased complexity in the particle detector back-end.

On-chip Sort&Bin module

To solve the issues emerging on bandwidth and power consumption in the Velopix2 design, this work suggests the addition of on-chip processing modules to aggregate packets from the same event in a single frame.

Using Pix-ESL as a prototyping platform, a sorting and binning algorithm was tested and appropriate parameters were identified. The whole design flow is presented in detail, from abstract high-level description to physical implementation.

Its results seem promising, with the proof-of-concept design being able to demonstrate its functionality and satisfy the area and power budgets. This module also paves the way for further on-chip processing, as it concentrates data in a single spot where processing functions can be applied to the whole frame.

Furthermore, the performance expected from the SystemC module closely matched the grouping efficiency of the RTL, thus validating Pix-ESL as a prototyping tool.

Future plans

Pix-ESL is being developed to improve its run-time performance, robustness, and add support for power and area estimation. The module library will be expanded as more processing modules and detector sub-systems are modeled.

Sort&Bin might be integrated in the upcoming Picopix and Velopix2 chips, and will need to be transformed into a reusable design, to adapt to other read-out chips such as Timepix or Medipix. It could be used as a basis for further processing modules in the periphery, which, for example, could be configured to perform filtering or clustering.

Bibliography

[1] Panos Charitos. *Designing pixel readout chips at CERN: From dream to reality.* CERN. 2013. URL: https://ep-news.web.cern.ch/content/designing-pixel-readout-chips-cern-dream-reality# (cit. on pp. 3, 4).

[2] Tuomas Poikela et al. «VeloPix: the pixel ASIC for the LHCb upgrade». In: *JINST* 10.01 (2015), p. C01057. DOI: 10.1088/1748-0221/10/01/C01057. URL: https://cds.cern.ch/record/2189759 (cit. on pp. 4, 24).

[3] Jashandeep Dhaliwal. *ON-CHIP PROCESSING FOR PIXEL IMAGERS IN A 28NM TECHNOLOGY FOR HEP APPLICATION.* 2022 (cit. on p. 7).

[4] X. Llopart et al. «Timepix4, a large area pixel detector readout chip which can be tiled on 4 sides providing sub-200 ps timestamp binning». In: *Journal of Instrumentation* 17.01 (Jan. 2022), p. C01044. DOI: 10.1088/1748-0221/17/01/C01044. URL: https://dx.doi.org/10.1088/1748-0221/17/01/C01044 (cit. on p. 9).

[5] Alessandro Caratelli. «Research and development of an intelligent particle tracker detector electronic system». PhD thesis. Jan. 2019. DOI: 10.5075/epfl-thesis-9702 (cit. on p. 11).

[6] Tuomas Poikela. «Readout Architectures for Hybrid Pixel Detector Readout Chips». PhD thesis. 2015. ISBN: 978-952-12-3235-0 (cit. on p. 11).

[7] Andrew Piziali Grant Martin Brian Bailey. *ESL Design and Verification: A Prescription for Electronic System Level Methodology.* Elsevier Science, 2010, p. 3 (cit. on p. 11).

[8] Rodolfo Azevedo Sandro Rigo and Luiz Santos. *Electronic System Level Design : An Open-Source Approach.* First. Springer Netherlands, 2011 (cit. on pp. 11, 12, 14).

[9] Felipe Klein, Guido Araujo, Rodolfo Azevedo, Roberto Leao, and Luiz C. V. dos Santos. «An efficient framework for high-level power exploration». In: *2007 50th Midwest Symposium on Circuits and Systems.* 2007, pp. 1046–1049. DOI: 10.1109/MWSCAS.2007.4488741 (cit. on p. 13).

[10] «IEEE Standard for Standard SystemC(R) Analog/Mixed-Signal Extensions Language Reference Manual». In: *IEEE Std 1666.1-2016* (2016), pp. 1–236. DOI: 10.1109/IEEESTD.2016.7448795 (cit. on pp. 13, 15).

[11] Accelera Systems Initiative. *SystemC. The language for System-level design, modeling and verification*. Accelera Systems Initiative. URL: https://systemc.org/overview/systemc/ (cit. on p. 13).

[12] S. Spannagel et al. «Allpix2: A modular simulation framework for silicon detectors». In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 901 (2018), pp. 164–172. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.06.020. URL: https://www.sciencedirect.com/science/article/pii/S0168900218307411 (cit. on p. 20).

[13] A Augusto Alves et al. «The LHCb Detector at the LHC». In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08005. DOI: 10.1088/1748-0221/3/08/S08005. URL: https://cds.cern.ch/record/1129809 (cit. on p. 25).

[14] M. van Beuzekom et al. «VeloPix ASIC development for LHCb VELO upgrade». In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 731 (2013). PIXEL 2012, pp. 92–96. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2013.04.016. URL: https://www.sciencedirect.com/science/article/pii/S0168900213004117 (cit. on p. 26).

[15] Tommaso Pajero. «VELO Upgrade II - The LHCb 4D pixel detector». In: *10th International Workshop on Pixel Detectors for Particles and Imaging - Santa Fe, New Mexico*. 2022 (cit. on p. 26).