UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Lucca Oliveira Facio Viccini

Simulation and Testing of the Upgraded Digital Signal Processing Firmware Using High-Level Synthesis on the Real-Time Trigger Path of the ATLAS Liquid Argon Calorimeters

Juiz de Fora

2025

Lucca Oliveira Facio Viccini

Simulation and Testing of the Upgraded Digital Signal Processing Firmware Using High-Level Synthesis on the Real-Time Trigger Path of the ATLAS Liquid Argon Calorimeters

Dissertação apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Orientador: Prof. Dr Luciano Manhães de Andrade Filho

Coorientador: Dr. Marcos Vinícius Silva Oliveira

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Oliveira Facio Viccini, Lucca.
Simulation and Testing of the Upgraded Digital Signal Processing Firmware Using High-Level Synthesis on the Real-Time Trigger Path of the ATLAS Liquid Argon Calorimeters / Lucca Oliveira Facio Viccini. – 2025.
111 f. : il.

Orientador: Luciano Manhães de Andrade Filho
Coorientador: Marcos Vinícius Silva Oliveira
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós Graduação em Engenharia Elétrica, 2025.

1.High Energy Physics. 2. FPGA. 3. High-Level Synthesis. 4. Functional Simulation, 5. Low latency. I. Manhães de Andrade Filho, Luciano, orient. II. Silva Oliveira, Marcos Vinícius. coorient. III. Título.

Lucca Oliveira Facio Viccini

Simulation and Testing of the Upgraded Digital Signal Processing Firmware Using High-Level Synthesis on the Real-Time Trigger Path of the ATLAS Liquid Argon Calorimeters

Dissertação apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Aprovada em (dia) de (mês) de (ano)

BANCA EXAMINADORA

Prof. Dr Luciano Manhães de Andrade Filho - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Leandro Rodrigues Manso Silva
Universidade Federal de Juiz de Fora

Prof. Dr. Victor Araujo Ferraz
Universidade Federal do Rio Grande do Norte

AGRADECIMENTOS

Agradeço, em primeiro lugar, à minha família pelo apoio incondicional e incentivo em todas as etapas da minha vida acadêmica. Sem o suporte deles, esta conquista jamais seria possível.

À minha noiva, Nathalia, por ser meu porto seguro, meu ponto de luz, por me motivar e por estar sempre ao meu lado em todos os momentos.

Aos colegas da equipe LATOME HLS, sempre solícitos e dispostos a ajudar, independentemente do momento.

Ao meu orientador, Luciano, por todos os aprendizados e confiança depositada em mim.

Ao meu coorientador, Marcos, pela oportunidade oferecida e com quem tanto aprendi nesta jornada.

A todos os professores, pelos ensinamentos e pelo apoio neste período de estudos.

Ao professor André Marcato, por seus ensinamentos, apoio, amizade e paciência ao longo deste período.

Ao professor Ivo, por toda a paciência e apoio durante esta jornada.

À Universidade Federal de Juiz de Fora, à Faculdade de Engenharia Elétrica, ao grupo do Calorímetro de Argônio Líquido, ao ATLAS e ao Centro Europeu de Pesquisa Nuclear (CERN), que me proporcionaram toda a infraestrutura para o desenvolvimento deste trabalho.

"We are a way for the cosmos to know itself."
— Carl Sagan

RESUMO

Esta dissertação apresenta a validação e simulação do pipeline digital de processamento de sinais no sistema de gatilho do Calorímetro de Argônio Líquido (LAr) do experimento ATLAS, com foco específico no firmware LATOME. A implementação utiliza High-Level Synthesis (HLS) para gerar os blocos Input Switch Matrix (ISM) e Output Summing (OSUM), responsáveis pela reorganização, processamento e agregação dos dados de Super Células em tempo real, atravessando múltiplos domínios de relógio. As versões de firmware 6.0 a 6.3 foram analisadas em profundidade, com as versões iniciais centradas no caminho de monitoramento e na validação do bloco REMAP, enquanto versões posteriores introduziram suporte aos caminhos de processamento eFEX e jFEX. Adotou-se uma estratégia de simulação em múltiplas camadas, combinando modelos Firmware-Agnostic e Firmware-Aware para possibilitar testes abrangentes desde componentes HLS isolados até o firmware integrado completo. Os principais esforços de validação incluíram a calibração do atraso do sinal SOP, a identificação da janela de metastabilidade e a verificação funcional dos mecanismos de sincronização entre domínios de relógio. A integração do módulo de diagnóstico Mini-FEX permitiu a observação em tempo real de erros de SOP e CRC, tanto em simulações quanto durante a operação do sistema ATLAS. A metodologia desenvolvida ao longo deste trabalho oferece uma estrutura reutilizável e escalável para validar futuras versões do firmware—como o caminho gFEX (v6.4) e o bloco User Code (v7.x)—e serve como referência para a adoção de HLS em outros sistemas FPGA de alta taxa de dados e baixa latência em experimentos de física de altas energias.

Palavras-chave: high-level synthesis; firmware latome; calorímetro de argônio líquido do ATLAS; sistema de gatilho em tempo real; validação e simulação em fpga.

ABSTRACT

This thesis presents the validation and simulation of the upgraded digital signal processing pipeline deployed in the ATLAS Liquid Argon Calorimeter trigger system, with a particular focus on the LATOME firmware. The implementation relies on High-Level Synthesis (HLS) to generate the Input Switch Matrix (ISM) and Output Summing (OSUM) blocks, which are responsible for reorganizing, processing, and aggregating real-time Super Cell data across multiple clock domains. Firmware versions 6.0 through 6.3 were analyzed in depth, with early versions centered on the monitoring path and REMAP validation, and later iterations introducing support for eFEX and jFEX trigger paths. A multi-layered simulation strategy was adopted, combining Firmware-Agnostic and Firmware-Aware models to enable comprehensive testing from standalone HLS components to fully integrated firmware deployments. Key validation efforts included SOP delay calibration, metastability window identification, and functional verification of clock synchronization logic. The integration of the Mini-FEX diagnostic module provided real-time observability of SOP and CRC errors, both in simulation and during ATLAS system operation. The methodology developed throughout this work offers a reusable and scalable framework for validating future firmware upgrades—such as the gFEX path (v6.4) and the User Code block (v7.x)—and serves as a reference for adopting HLS in other high-throughput, low-latency FPGA-based systems in high-energy physics experiments.

Keywords: high-level synthesis; latome firmware; atlas liquid argon calorimeter; real-time trigger system; fpga validation and simulation.

List of Figures

List of Tables

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog-to-Digital Converter
ALICE	A Large Ion Collider Experiment
AMC	Advanced Mezzanine Card
ATCA	Advanced Telecommunications Computing Architecture
ATLAS	A Toroidal LHC ApparatuS
BC	Bunch Crossing
BCID	Bunch Crossing Identifier
CERN	Conseil Européen pour la Recherche Nucléaire
CMS	Compact Muon Solenoid
CRC	Cyclic Redundancy Check
CDC	Clock Domain Crossing
DAQ	Data Acquisition
EMEC	Electromagnetic Endcap Calorimeter
eFEX	Electromagnetic Feature Extractor
FEX	Feature Extractor
FPGA	Field-Programmable Gate Array
gFEX	Global Feature Extractor
HEC	Hadronic Endcap Calorimeter
HEP	High-Energy Physics
HLS	High-Level Synthesis
HLT	High-Level Trigger
jFEX	Jet Feature Extractor
LAr	Liquid Argon
LArC	LAr Carrier Board
LATOME	LAr Trigger prOcessing MEzzanine
LDPB	LAr Digital Processing Blade
LHC	Large Hadron Collider
LHCb	LHC beauty experiment
L1	Level-1 (Trigger)
LLI	Low-Level Interface
LTDB	LAr Trigger Digitizer Board
OSUM	Output Summing
PU	Processing Unit
SC	Super Cell
SM	Switch Matrix
SOP	Start of Packet
TDAQ	Trigger and Data Acquisition
TDR	Technical Design Report
TTC	Timing, Trigger, and Control
UC	User Code

Contents

1 Introduction

High-energy physics (HEP) is the field dedicated to studying the fundamental constituents of matter and the forces governing their interactions. Its primary goal is to answer open questions about the nature of mass, the composition of dark matter, and the possible unification of fundamental forces. Over the past decades, research in this area has not only advanced our understanding of the universe but also driven technological developments in computing, medical imaging, and materials science.

Founded in 1954, the European Organization for Nuclear Research (*Conseil Européen pour la Recherche Nucléaire*, CERN) is the world's largest center for particle physics research. CERN hosts the Large Hadron Collider (LHC), the most powerful particle accelerator ever built, which enables the study of matter under extreme conditions by colliding proton beams at unprecedented energies [1, 2].

The LHC consists of a 27-kilometer underground ring located at the border between France and Switzerland, as shown in Figure 1. Proton beams are accelerated in opposite directions and are steered to collide at four designated interaction points. Each interaction point hosts a major experiment: ATLAS and CMS are general-purpose detectors designed to explore a broad range of physics phenomena, while ALICE and LHCb are dedicated to more specialized studies [3].

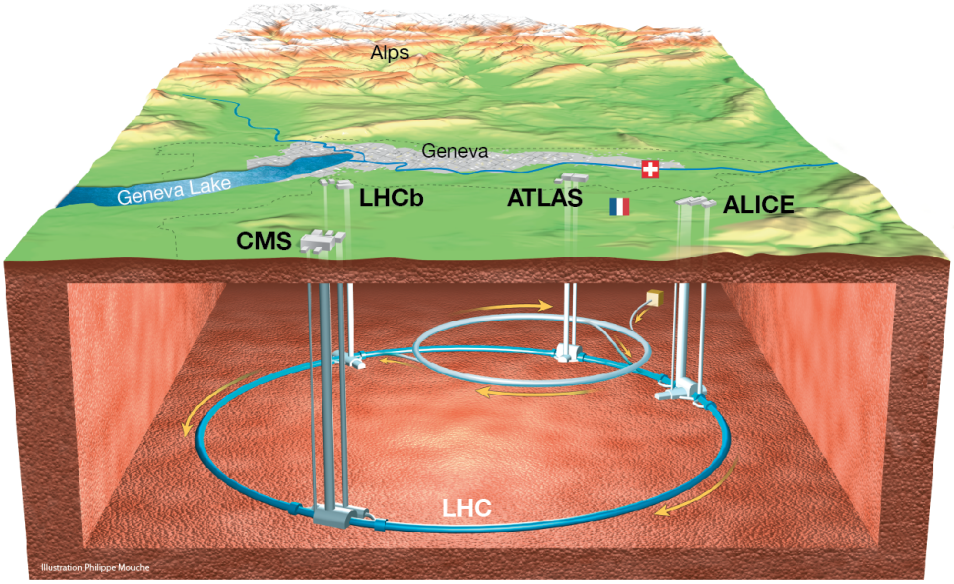

Figure 1 – Overview of the LHC accelerator ring and its four main experimental sites: ATLAS, CMS, ALICE, and LHCb [5].

1.1 Motivation

Having established the broader context of the LHC and its experiments, this section narrows the focus to the technical challenges that motivate the development and validation of the firmware described in this thesis.

The efficacy of high-energy physics experiments, such as ATLAS, is contingent upon instrumentation capable of processing data at extreme rates with stringent precision and reliability requirements. The LHC operates in multi-year data-taking periods known as *Runs*, each separated by long shutdowns dedicated to maintenance and upgrades. With each successive Run, the LHC delivers higher luminosities, leading to increased data throughput and placing more stringent demands on real-time data processing systems.

A key performance indicator of a collider is its luminosity, which quantifies the number of potential particle collisions per unit area per second [4]. During Run 3, the LHC operates with proton-proton collisions at a center-of-mass energy of 13.6 TeV and achieves a peak luminosity of $3 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ [6]. This high luminosity increases the probability of observing rare physics processes but also results in significant pile-up—an average of about 80 simultaneous interactions per bunch crossing. Such conditions place considerable demands on the ATLAS detector's data acquisition and trigger systems, which must process vast amounts of information in real time while maintaining high accuracy and low latency.

The ATLAS calorimeters play a central role in the trigger decision by providing precise energy measurements. In particular, the Liquid Argon (LAr) calorimeter, responsible for detecting electromagnetic showers from particles such as electrons and photons, delivers fine-granularity energy data that is essential for accurate event selection at the hardware trigger level [7].

Operating under the LHC's high collision rate of 40 MHz, the ATLAS detector generates an enormous volume of data every second [8]. To manage this, the Level-1 (L1) Trigger system plays a vital role by reducing the event rate from 40 MHz to a manageable 100 kHz. This rapid filtering ensures that only the most relevant events are retained for further processing, significantly reducing the burden on downstream systems. The L1 Trigger relies on precise energy measurements from the LAr calorimeter, making real-time signal processing a crucial component of ATLAS's functionality.

In preparation for Run 3, the LAr calorimeter underwent a significant upgrade to handle the increased collision rates and improve trigger decision performance [6]. This included the introduction of Super Cells (SC)—aggregations of multiple calorimeter cells—which provide finer spatial granularity for more precise energy reconstruction. To process this increased data volume, the LATOME (LAr Trigger prOcessing MEzzanine) boards were developed as a central element of the upgraded back-end electronics. These

boards are responsible for aggregating, filtering, and preparing SC data for the Level-1 trigger system, enabling fast and accurate event selection.

To cope with the increased data throughput and architectural complexity, the LATOME firmware is undergoing a major redesign. Two critical processing blocks—REMAP and OSUM—are being re-implemented using High-Level Synthesis (HLS), enabling a more modular and flexible architecture. This approach aims to improve both maintainability and performance while meeting the stringent latency and reliability requirements of the ATLAS trigger system. Ensuring the correctness of these upgrades requires a rigorous validation and simulation framework.

This thesis is dedicated to the simulation and verification of these HLS-based upgrades to the LATOME firmware. By implementing a multi-layered verification strategy, it verifies functionality and timing across increasing levels of integration. This work contributes to ensuring the reliability and performance of the LATOME firmware as part of the ATLAS Phase-I trigger upgrade. The specific objectives of this thesis are outlined in the next section.

1.2 Objectives

This work focuses on the HLS-based upgrades to the LATOME firmware. Its primary objectives are:

- To redesign and implement the critical REMAP and OSUM blocks of the LATOME firmware utilizing High-Level Synthesis (HLS). These components are crucial for reorganizing, processing, and aggregating real-time Super Cell data across multiple clock domains.
- To leverage HLS for enhanced development efficiency, improved maintainability, and optimized firmware performance. HLS allows for describing functionality in high-level languages, reducing development effort and promoting rapid design iteration.
- To conduct functional simulation and verification of the upgraded LATOME firmware, specifically validating the integrated HLS-based components . This involves C-level simulations and Python-based testbenches using Cocotb and QuestaSim.
- To establish and apply a robust multi-layered verification methodology, systematically addressing various levels of integration and complexity within the firmware. This includes a strategy with four distinct layers, from unit-level tests to full system simulations.
- To ensure that the LATOME firmware adheres to stringent functional and timing requirements for stable operation within the demanding environment of the ATLAS

trigger system . This objective is supported by extensive hardware validation and real-time ATLAS tests, including SOP delay calibration and CRC error monitoring.

These objectives collectively support the readiness of the LATOME firmware to handle the increased data rates and processing demands of the upgraded ATLAS trigger system.

1.3 Thesis Structure

This thesis is organized into seven chapters, designed to progressively build the reader's understanding from the high-level context of the ATLAS experiment to the detailed validation results of the LATOME firmware upgrade.

- **Chapter 1: Introduction.** This chapter presents the motivation behind the research, outlining the technical challenges in real-time signal processing for the ATLAS experiment and the context of the firmware upgrade. It also defines the objectives of this thesis and details its overall structure.
- **Chapter 2: The ATLAS Experiment.** This chapter provides a broader context for the work by giving an overview of the ATLAS experiment and the Large Hadron Collider (LHC). It describes the ATLAS detector's subsystems, the role of the Trigger and Data Acquisition (TDAQ) system, and the ATLAS Detector Upgrade Program.
- **Chapter 3: The Liquid Argon Calorimeter Trigger Readout Architecture.** This chapter delves into the technical background of the ATLAS Liquid Argon (LAr) calorimeter and its upgraded trigger readout chain. It covers the physical structure of the LAr calorimeters, the overview of the LAr trigger readout path, the Phase-I upgrade with the introduction of Super Cells, and the design of the front-end and back-end electronics (LTDBs, LDPB, and LATOME boards). It also explains the timing and synchronization infrastructure and monitoring and real-time trigger paths essential for system operation and validation.
- **Chapter 4: LATOME Firmware.** This chapter describes the overall functional overview and architectural foundation of the LATOME firmware, including its various processing blocks, data flow, and clock domain organization. It also details the characteristics of the legacy firmware (Version 5), providing essential context for understanding the subsequent High-Level Synthesis (HLS)-based upgrades.
- **Chapter 5: Methodology and Tools.** This chapter introduces the key hardware platforms and software tools employed in this work, including Field-Programmable Gate Arrays (FPGAs) and the principles of High-Level Synthesis (HLS) using Siemens Catapult. It also presents the HLS Wrapper Generator for VHDL integration, the

Switch Matrix architecture as a foundational design pattern, and the Coroutine-Based Co-Simulation with Cocotb and QuestaSim. This chapter concludes with an overview of the multi-layered simulation and validation strategy that underpins the verification efforts.

- **Chapter 6: Firmware Evolution and Validation: Versions 6.0 to 6.3.** This comprehensive chapter details the evolution of the LATOME firmware from Version 6.0 to Version 6.3, outlining the architectural modifications and the corresponding simulation and validation strategies for each release. It presents the HLS-driven redesigns of key components such as the **Remapping (REMAP)** and **Output Summing (OSUM)** blocks, and their integration and validation on the monitoring and trigger paths (eFEX and jFEX). This chapter also provides a detailed explanation and application of the multi-layered simulation strategy (Layers 0 to 3), encompassing C-level simulations, RTL co-simulations, and full-firmware validation with both Firmware-Agnostic and Firmware-Aware models. Furthermore, it includes hardware tests, both in laboratory and deployed in ATLAS, particularly the SOP delay calibration campaigns and the integration of the Mini-FEX diagnostic module for real-time error observability.

- **Chapter 7: Conclusion.** This final chapter summarizes the main contributions of this thesis, discusses its impact on the ATLAS trigger system, and outlines potential future improvements to the firmware and validation methodologies.

2 The ATLAS Experiment

The ATLAS (*A Toroidal LHC ApparatuS*) experiment, where this work is situated, is the largest general-purpose particle detector ever built. It is operated by a collaboration of more than 5,500 scientists from 245 institutes across 42 countries [9]. The detector is designed to identify and measure the properties of particles produced in proton-proton collisions at the LHC by reconstructing their trajectories, momenta, and decay products [10]. ATLAS enables a broad physics program, ranging from precision tests of the Standard Model—including the discovery of the Higgs boson in 2012—to searches for physics beyond the Standard Model, such as dark matter candidates and extra spatial dimensions.

2.1 ATLAS Overview

Figure 2 shows the ATLAS detector, a cylindrical structure measuring approximately 25 meters in height and 44 meters in length and weighing approximately 7000 tons. It is designed to provide nearly complete coverage around the collision point, spanning a solid angle close to 4π. This hermetic coverage enables the precise reconstruction of particle trajectories and energy deposits across a wide range of physics processes [17].

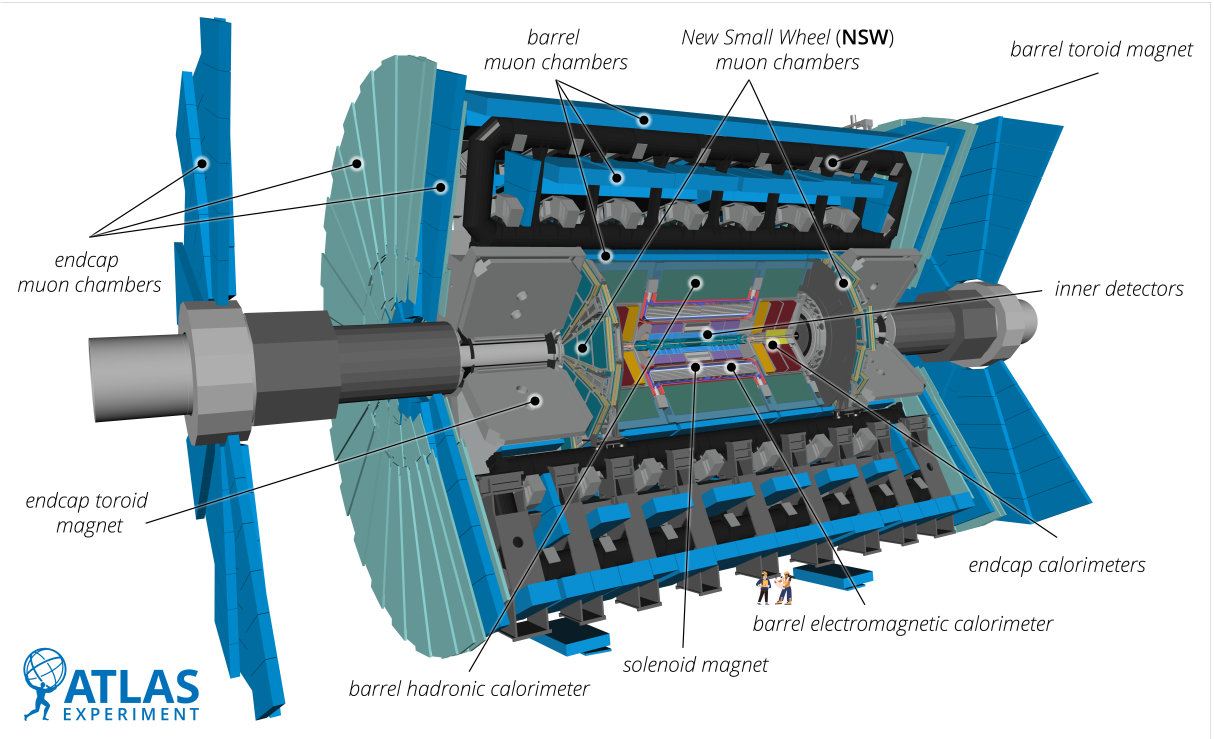

Figure 2 – Schematic view of the ATLAS detector, illustrating its primary subsystems: the Inner Detector, Calorimeters, Magnet System, and Muon Spectrometer [11].

2.2 ATLAS Subsystems

Figure 3 presents a cross-sectional view of the ATLAS detector, illustrating how various particles interact with its subsystems. The ATLAS detector consists of several concentric subsystems, each optimized for a specific role in particle detection and measurement.

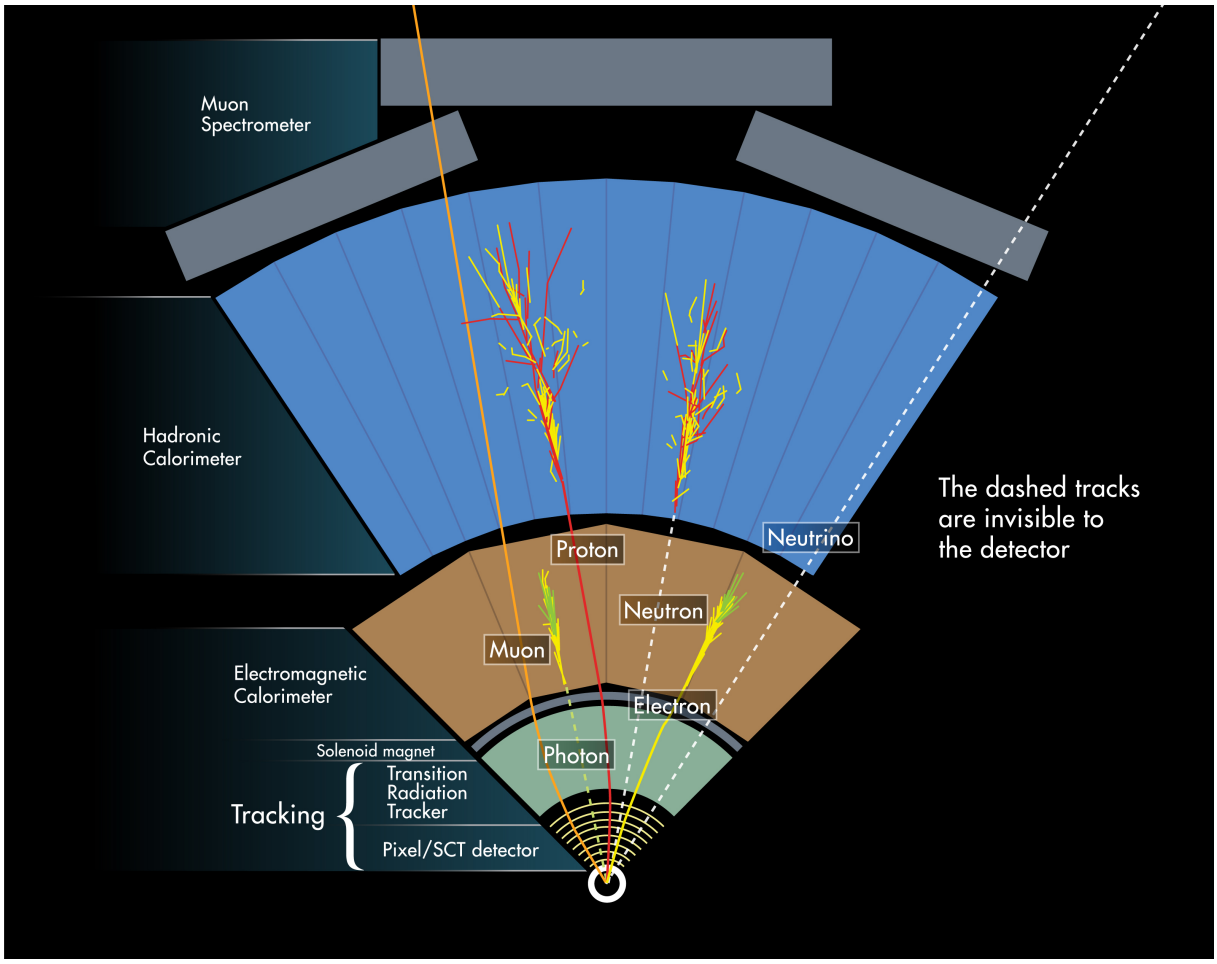

Figure 3 – Transversal slice of the ATLAS detector, illustrating its primary subsystems and the typical paths of different particle types as they interact with the Inner Detector, Calorimeters, and Muon Spectrometer.

Closest to the collision point, the Inner Detector (ID) tracks charged particles with high precision, measuring their direction, momentum, and charge [12].

The Magnet System, shown in gray in Figure 3, consists of a central solenoid providing the magnetic field for the Inner Detector, and large toroidal magnets positioned around the calorimeters and muon spectrometer to enable momentum measurements of muons [13].

The Muon Spectrometer, located in the outermost layers, detects and measures the momentum of muons, which can penetrate the entire detector [14].

The ATLAS calorimeter system, shown in orange and blue in Figure 3 and rendered in Figure 4, is positioned between the Inner Detector and the Muon Spectrometer. It measures the energy deposited by particles as they pass through, producing ionization signals proportional to that energy. The calorimeter system consists of the Liquid Argon Calorimeter (LAr), designed to measure the energy of particles that interact electromagnetically, such as electrons and photons, and the Tile Calorimeter (TileCal), which measures the energy of particles undergoing hadronic interactions, such as protons, neutrons, and pions [15].

In addition to its role in offline physics analysis, the calorimeter system provides critical input to the Level-1 (L1) trigger — a hardware-based system responsible for rapidly selecting potentially interesting collision events in real time. This capability relies on accurate and fast energy measurements from the calorimeters. The next section discusses how this functionality is implemented within the Trigger and Data Acquisition (TDAQ) system.

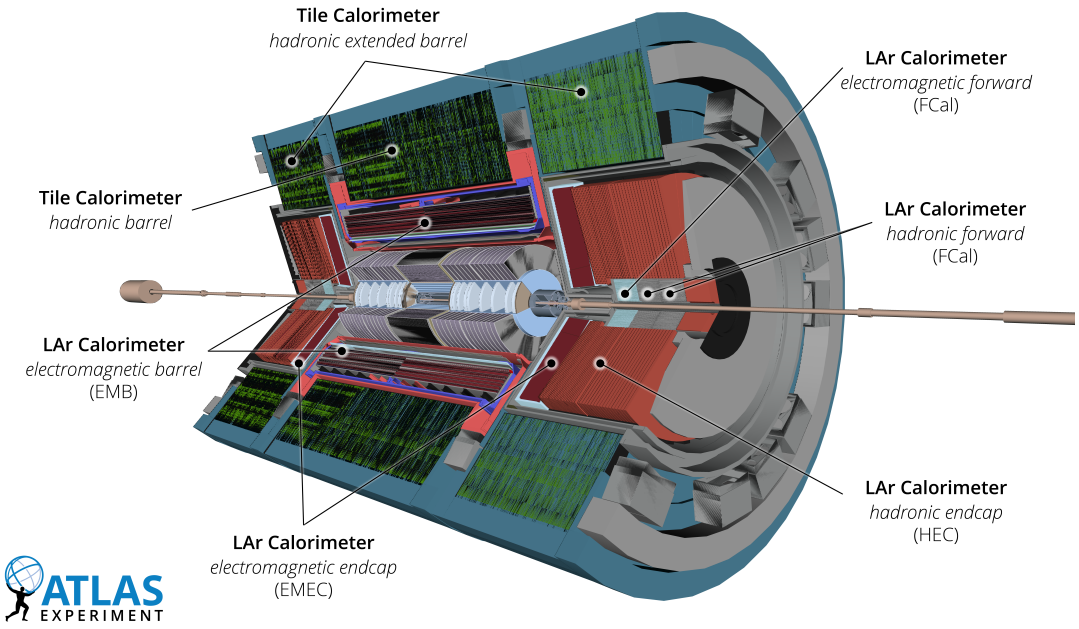

Figure 4 – Cut-away view of the ATLAS calorimeter system, comprising the Liquid Argon electromagnetic calorimeters and the Tile hadronic calorimeter.

2.3 ATLAS Trigger and Data Acquisition System (TDAQ)

Figure 5 provides an overview of the ATLAS Trigger and Data Acquisition (TDAQ) system, which handles the large data volume produced by LHC collisions. Proton bunches cross at a frequency of 40 MHz, known as Bunch Crossings (BC), where each crossing can potentially produce one or more proton-proton collisions. Each bunch crossing occurs

every 25 ns, aligned with the LHC's 40 MHz collision frequency. The resulting data volume far exceeds the storage and processing capabilities of conventional systems [16].

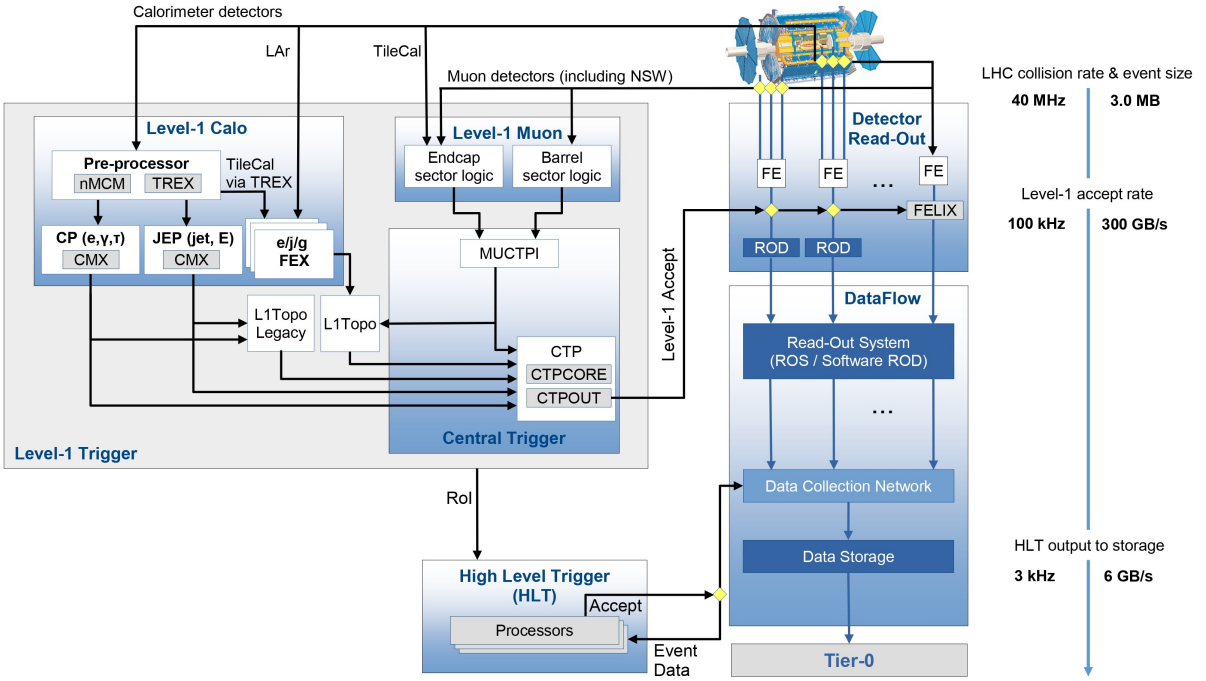

Figure 5 – Overview of the ATLAS Trigger and Data Acquisition (TDAQ) system, showing the data flow from detector readout through the Level-1 hardware trigger and the High-Level software trigger to offline storage [16].

To manage this challenge, ATLAS employs a sophisticated filtering architecture designed to identify and retain only the most relevant events for physics analysis. The data processing is divided into two stages: *online* and *offline*. Online processing occurs in real time and is responsible for selecting events during collisions, while offline processing occurs after data recording and involves detailed analysis using advanced reconstruction algorithms. Offline processing includes full event reconstruction, calibration, alignment corrections, and detailed physics analysis, performed after data recording.

The TDAQ system reduces data through a multi-level filtering strategy that progressively decreases the event rate while preserving essential physics information. It is structured into two main levels:

- **Level-1 (L1) Trigger**: A hardware-based filtering stage that processes data from the calorimeters and muon detectors. Operating with a fixed latency of 2.5 μs, it reduces the event rate from 40 MHz to approximately 100 kHz. A key component is the **Level-1 Calorimeter Trigger (L1Calo)**, which receives data from the electromagnetic and hadronic calorimeters and identifies high-energy electromagnetic showers, hadronic jets, and global event quantities for early event selection [16].

- **High-Level Trigger (HLT)**: A software-based stage that receives events selected by L1. It applies full event reconstruction and more sophisticated selection algorithms to further reduce the event rate to about 3 kHz, storing only the most relevant events for offline analysis [16].

A key input to the L1Calo system comes from the upgraded Liquid Argon (LAr) calorimeter. Signals from the calorimeter are digitized by the front-end electronics and processed in real time by the back-end trigger readout system, including the LATOME boards, which prepare the energy information used by the Level-1 trigger. The processed data is then transmitted to the L1Calo Feature Extractors—eFEX, jFEX, and gFEX—which identify electromagnetic clusters, jets, and global event properties for triggering purposes.

This hierarchical filtering system, combining both real-time and offline analysis, is essential for the success of the ATLAS experiment, allowing researchers to select rare physics signals while discarding the majority of non-relevant collision events.

2.4 The ATLAS Detector Upgrade Program

The ATLAS Phase-I upgrade was designed to prepare the detector for the higher luminosity conditions of Run 3 and beyond. This program involved upgrades to several subsystems, including the muon detectors, the forward proton system, and the trigger and data acquisition (TDAQ) infrastructure. A key component of this effort was the upgrade of the Liquid Argon (LAr) calorimeter trigger readout, which is the focus of this thesis.

The LHC follows a structured schedule alternating between Runs and long shutdowns, as shown in Figure 6. Each Run corresponds to an operational phase of the accelerator, delivering proton-proton collisions for data collection by experiments. In contrast, a long shutdown is a planned maintenance and upgrade period, during which major technological improvements are implemented to prepare the LHC and its detectors for the next operational cycle.

The LHC began its first operational cycle, Run 1, from 2010 to 2013, reaching a center-of-mass energy of up to 8 TeV and leading to the discovery of the Higgs boson. After Long Shutdown 1, extensive hardware upgrades allowed Run 2, from 2015 to 2018, to operate at a center-of-mass energy of 13 TeV, significantly increasing the dataset available for precision measurements and new physics searches.

Following the conclusion of Run 2, the LHC and its experiments, including the ATLAS detector, entered Long Shutdown 2 in December 2018. Over the next years, critical upgrades were implemented as part of the Phase-I upgrade program. These improvements aimed to maintain trigger efficiency and detector performance under the higher pile-up conditions expected in Run 3 and the future High-Luminosity LHC (HL-LHC) era.

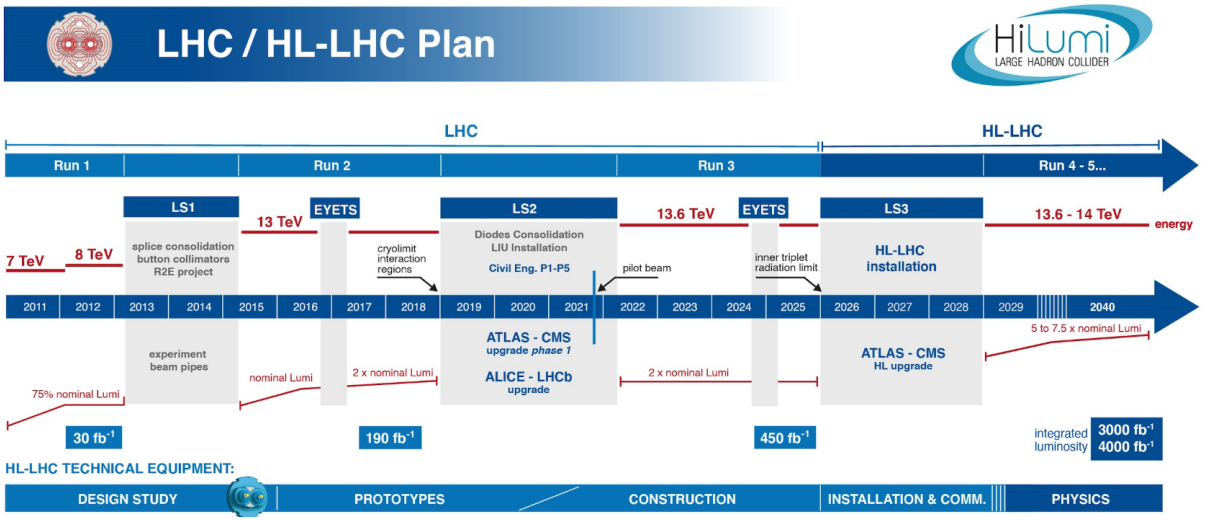

Figure 6 – Timeline of the LHC operational schedule, including Runs, Long Shutdowns, and the transition to the High-Luminosity LHC (HL-LHC) [21].

Among these, the LAr calorimeter received a major trigger readout upgrade. This enhancement replaces the previous analog summing approach with a fully digitized readout, providing higher spatial granularity in the calorimeter data for the Level-1 trigger system. This change significantly improves event selection capabilities, allowing the trigger to operate reliably at higher luminosities while maintaining the precision of energy measurements at the trigger level.

A detailed examination of the Liquid Argon calorimeter, the architecture of its upgraded trigger readout system, and the motivation for the firmware development presented in this thesis is provided in the following chapter.

3 The Liquid Argon Calorimeter Trigger Readout Architecture

The Liquid Argon (LAr) calorimeter system is a critical component of the ATLAS detector, providing high-resolution energy measurements that are essential for both offline physics analysis and real-time trigger decisions. This chapter presents the architecture of the LAr trigger readout chain, with a focus on the back-end electronics where the firmware development and validation described in this thesis are implemented.

The chapter begins with an overview of the physical structure and segmentation of the LAr calorimeter, which is essential for understanding the data organization, channel mapping, and firmware configuration. This is followed by a description of the data flow from the calorimeter to the Level-1 Calorimeter Trigger (L1Calo) and the Front-End Link eXchange (FELIX) systems. The Phase-I upgrade and the introduction of Super Cells are then discussed, providing the context for the increased data granularity that drives the architecture of the upgraded readout chain. These descriptions are intended to contextualize the operation and design of the back-end processing system—the Liquid Argon Digital Processing Blade (LDPB) and the LATOME boards—which are responsible for real-time energy reconstruction, data formatting, and delivering the processed information to the trigger and data acquisition systems. The chapter concludes with descriptions of the front-end electronics, the timing and synchronization infrastructure, and the monitoring paths that ensure coherent operation and enable real-time validation of the system.

Together, these sections provide the operational and architectural background necessary to understand the firmware development, the High-Level Synthesis (HLS) upgrades, and the validation methodologies presented in this thesis.

3.1 Structure of the LAr Calorimeters

The ATLAS Liquid Argon (LAr) calorimeter is a sampling detector designed to measure the energy of electromagnetic particles—such as electrons and photons—and hadronic particles in the forward region. Its geometry and segmentation directly influence the design of the trigger readout and the firmware processing chain described in this work. It operates by alternating layers of passive absorber material with active liquid argon gaps. Particles traversing the detector produce ionization in the liquid argon, which is then collected by electrodes and converted into electrical signals proportional to the deposited energy.

The LAr calorimeter is divided into four distinct detector regions, each optimized for different pseudorapidity ranges and physics goals, as illustrated in Figure 7:

- **Electromagnetic Barrel (EMB)** — Covers the central region of the detector ($|\eta| < 1.475$). It is responsible for precise energy measurements of electrons and

photons produced near the interaction point.

- **Electromagnetic Endcap (EMEC)** — Extends the coverage to the forward regions ($1.375 < |\eta| < 3.2$), maintaining high resolution for electromagnetic showers at larger pseudorapidities.
- **Hadronic Endcap Calorimeter (HEC)** — Positioned behind the EMEC, it measures hadronic showers from strongly interacting particles, providing complementary coverage to the TileCal in the central region.
- **Forward Calorimeter (FCAL)** — Covers the extreme forward region ($3.2 < |\eta| < 4.9$), where particle fluxes are highest. It is designed to withstand high radiation and measure both electromagnetic and hadronic energy.

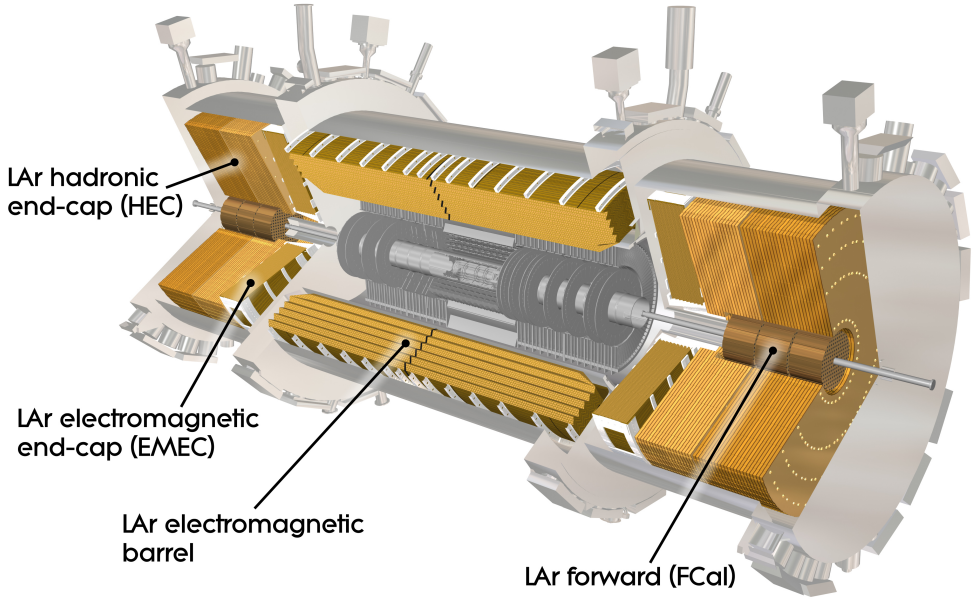

Figure 7 – Diagram of the ATLAS Liquid Argon Calorimeter, showing its four main regions: EMB, EMEC, HEC, and FCAL.

The geometry of the calorimeter features fine segmentation in both lateral (η-ϕ) and longitudinal layers, optimized for particle identification, energy resolution, and pile-up mitigation. The electromagnetic calorimeters (EMB and EMEC) consist of multiple longitudinal layers with different granularity: a finely segmented front layer for shower position determination, a middle layer for the bulk of energy measurement, and a coarse back layer to capture shower leakage.

This segmentation is not only crucial for offline physics reconstruction but also directly impacts the design of the trigger readout system. The mapping of calorimeter cells into Super Cells (SCs), the channel distribution into front-end and back-end electronics, and the configuration of firmware processing blocks are all intrinsically tied to the calorimeter geometry. Different detector regions have different numbers of Super Cells, data rates, and processing requirements, which are reflected in both the hardware and firmware architecture of the trigger readout chain.

The following sections present the architecture of the trigger readout system, providing the operational context for the back-end processing and firmware development described in this thesis.

3.2 Overview of the LAr Trigger Readout Path

The Liquid Argon (LAr) calorimeter provides real-time transverse energy measurements and digitized Super Cell (SC) data to the ATLAS Level-1 (L1) trigger system and data acquisition. This information is essential for event selection, allowing the trigger to identify electromagnetic clusters, hadronic jets, and global event properties such as missing transverse energy (E_T^{miss}), while simultaneously ensuring that the raw digitized signals are available for detector monitoring and full event reconstruction.

As illustrated in Figure 8, the LAr trigger readout chain consists of multiple stages. Signals generated in the calorimeter are digitized by the front-end electronics, the Liquid Argon Trigger Digitizer Boards (LTDBs), operating at 40 MHz. The digitized Super Cell (SC) data is transmitted optically to the back-end system — the Liquid Argon Digital Processing Blade (LDPB) — which hosts four LATOME boards.

The LATOME boards process the SC data in real time, performing energy calculation, basic filtering, and data formatting. The output is simultaneously transmitted to two destinations:

- The Level-1 Calorimeter Trigger (L1Calo) Feature Extractors (FEX): eFEX, jFEX, and gFEX, where the trigger decision is formed.
- The Front-End Link eXchange (FELIX) system, which handles data acquisition, detector monitoring, and validation streams.

The following sections describe the Phase-I upgrade that introduced this trigger readout architecture, as well as the implementation of its back-end and front-end electronics, the Super Cell mapping, the timing and synchronization infrastructure, and the data validation paths.

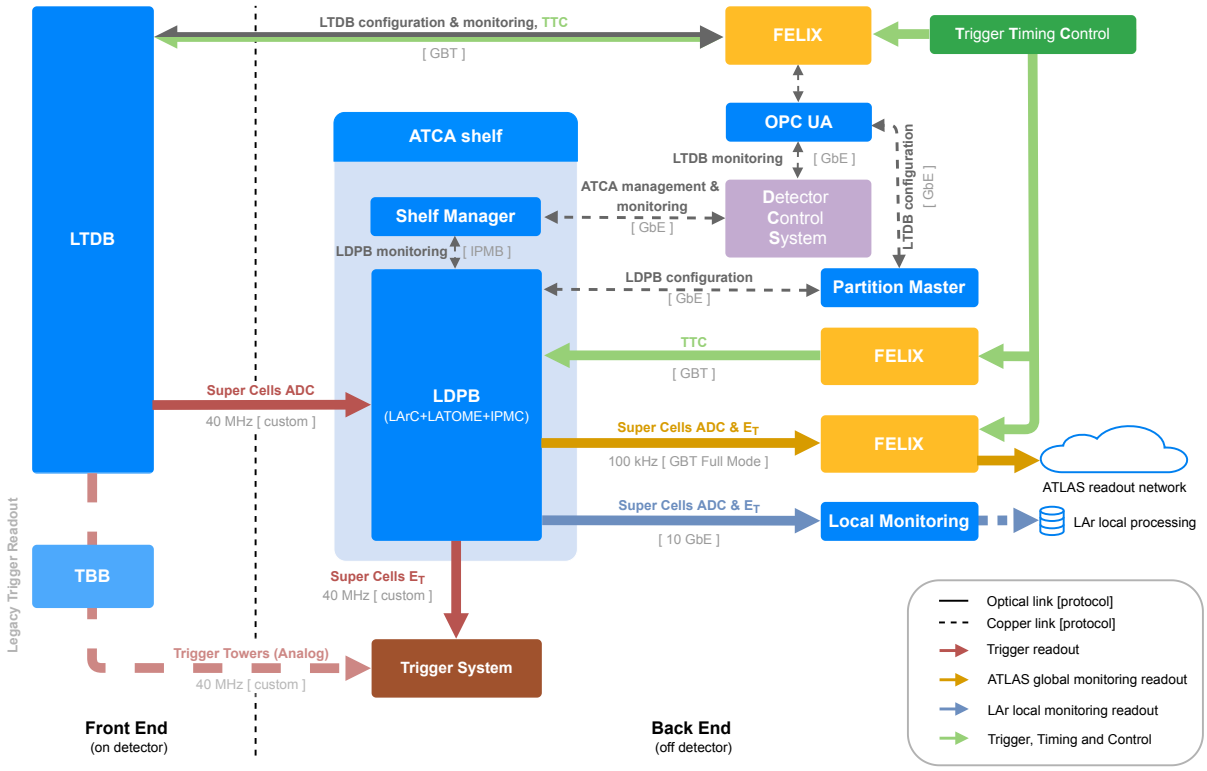

Figure 8 – The ATLAS LAr calorimeter trigger readout system. Signals are digitized by the LTDBs, processed by the LATOMEs in the back-end, and transmitted to both the L1Calo Feature Extractors (eFEX, jFEX, gFEX) for triggering and the FELIX system for data acquisition and monitoring.

3.3 Phase-I Upgrade and the Introduction of Super Cells

The Phase-I upgrade of the ATLAS Liquid Argon (LAr) calorimeter introduced substantial improvements to the Level-1 (L1) trigger system, most notably by replacing the legacy Trigger Tower-based readout with a fully digital path based on high-granularity Super Cells (SCs). This enhancement increased the spatial resolution by up to a factor of ten in key calorimeter regions, enabling finer energy measurements and more selective event triggering.

In the legacy system, Trigger Towers (TTs) covered an area of $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$ and summed energy from all longitudinal layers into a single value. In contrast, SCs retain layer-specific information, with the middle layer achieving a finer granularity of $\Delta\eta \times \Delta\phi = 0.025 \times 0.1$. This segmentation allows for more precise reconstruction of electromagnetic showers and improves the identification of electrons, photons, and tau leptons [6].

Figure 9 illustrates this improvement. The left side shows a single Trigger Tower summing transverse energy across all layers, whereas the right highlights the enhanced segmentation of Super Cells. By maintaining separate energy measurements per layer,

SCs enable better discrimination of overlapping energy deposits and provide more detailed input to the trigger algorithms.

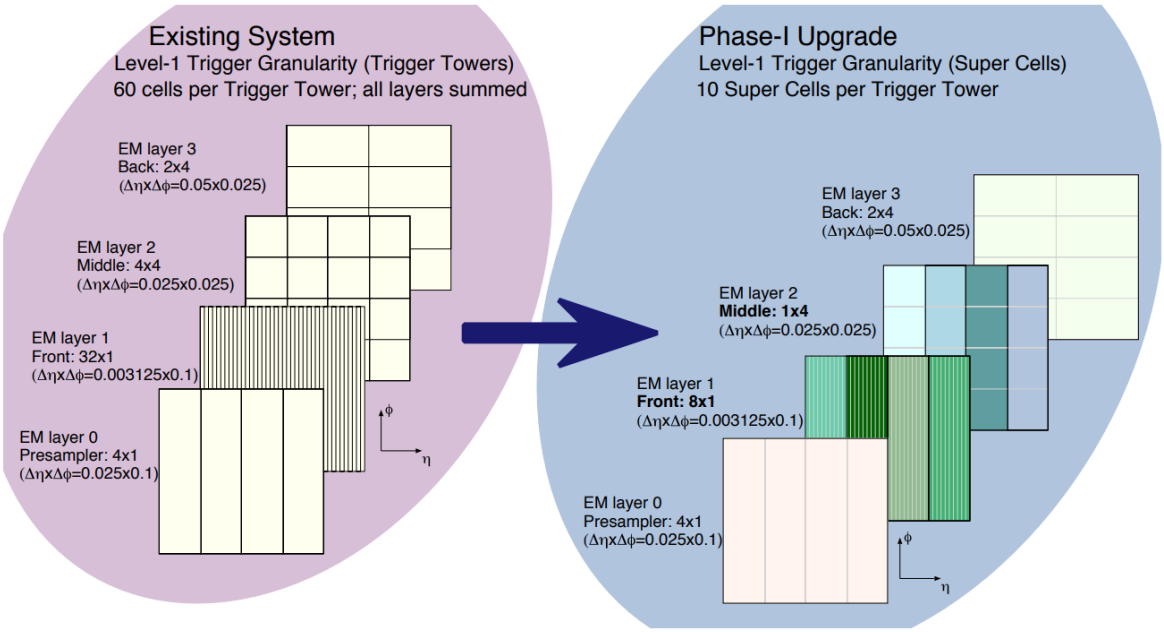

Figure 9 – Comparison between the legacy Trigger Tower segmentation and the Phase-I Super Cell segmentation. Super Cells provide finer granularity, preserve longitudinal layer information, and improve trigger performance.

The increase in granularity results from both longitudinal and lateral segmentation. Each Trigger Tower is typically divided into four longitudinal layers, and for certain layers — particularly the middle layer — additional lateral segmentation reduces the $\Delta\eta$ granularity from 0.1 to 0.025 while keeping $\Delta\phi = 0.1$. This finer segmentation directly impacts the design of the trigger readout system by significantly increasing the volume of data that must be processed in real time.

To handle this increase in data, the readout architecture was completely redesigned. The front-end Liquid Argon Trigger Digitizer Boards (LTDBs) digitize the SC signals at 40 MHz, while the back-end processors — the LATOME boards — perform real-time energy calculation and data formatting for the Level-1 trigger system and the FELIX data acquisition path. The next sections describe the implementation of these electronics and their roles in the trigger readout chain.

3.4 Back-End Electronics

The back-end electronics of the LAr trigger readout system are responsible for receiving the digitized Super Cell (SC) data from the front-end, performing real-time energy reconstruction, and delivering the processed information to both the Level-1 Calorimeter

Trigger (L1Calo) and the Front-End Link eXchange (FELIX) for trigger decision-making and data acquisition.

At the core of the back-end system is the Liquid Argon Digital Processing Blade (LDPB), an ATCA carrier board that hosts four Advanced Mezzanine Cards (AMCs) known as LATOMEs, as shown in Figure 10. Each LATOME is equipped with an Intel Arria 10 FPGA, which implements the full processing chain required for the trigger path.

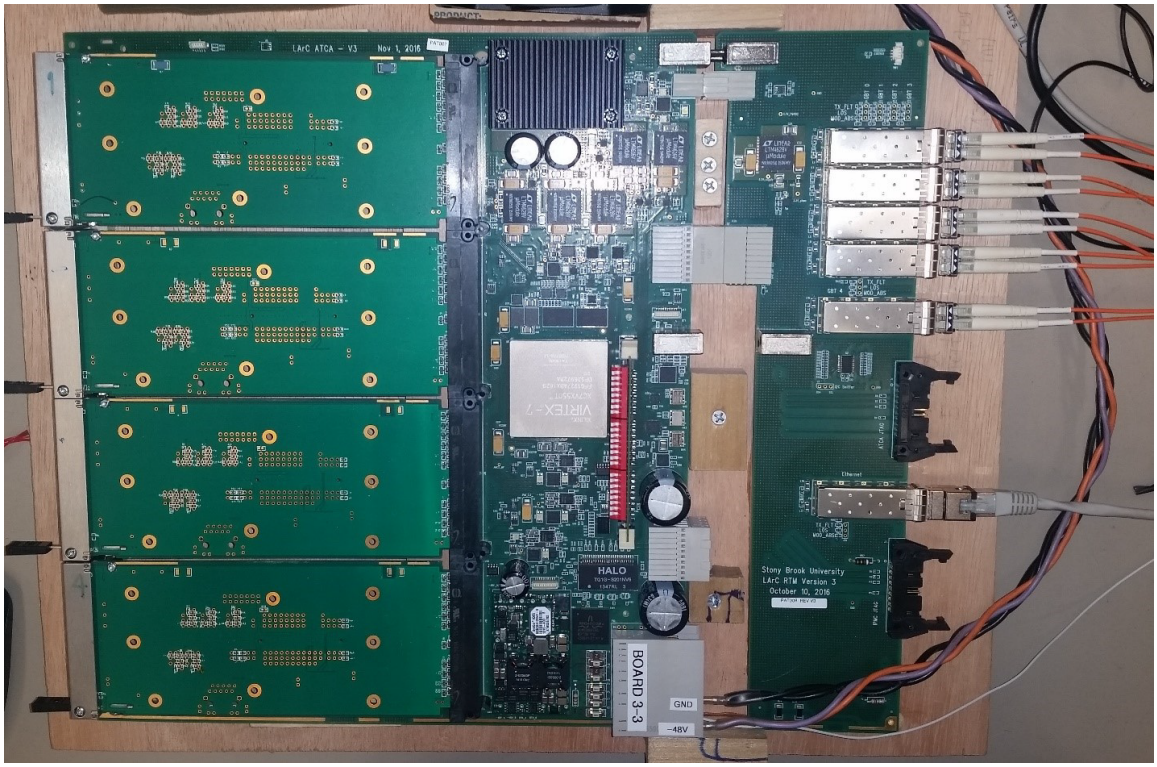

Figure 10 – LAr Carrier (LArC) ATCA board hosting four LATOME boards.

Each LATOME receives data from 48 optical input links operating at 320 MHz. These links, connected via four MTP-12 connectors visible on the right side of the LATOME board in Figure 11, carry serialized SC data from the Liquid Argon Trigger Digitizer Boards (LTDBs). The incoming data is deserialized, realigned, and grouped by bunch crossing in the LATOME firmware to ensure proper timing and data integrity.

The real-time processing performed in the LATOME includes pedestal subtraction, basic filtering, noise suppression, and the computation of the Super Cell transverse energy (E_T). The processed output is then formatted according to the interface specifications of the downstream systems. Data is transmitted simultaneously to the L1Calo Feature Extractors (FEX) via a MTP-48 connector located on the right side of the board, and to the FELIX system for data acquisition, monitoring, and validation.

The L1Calo FEX processors are divided into three systems:

- **eFEX (Electromagnetic Feature Extractor)** — responsible for electron and photon identification;
- **jFEX (Jet Feature Extractor)** — responsible for jet and tau detection;
- **gFEX (Global Feature Extractor)** — responsible for computing global event properties, such as missing transverse energy (E_T^{miss}).

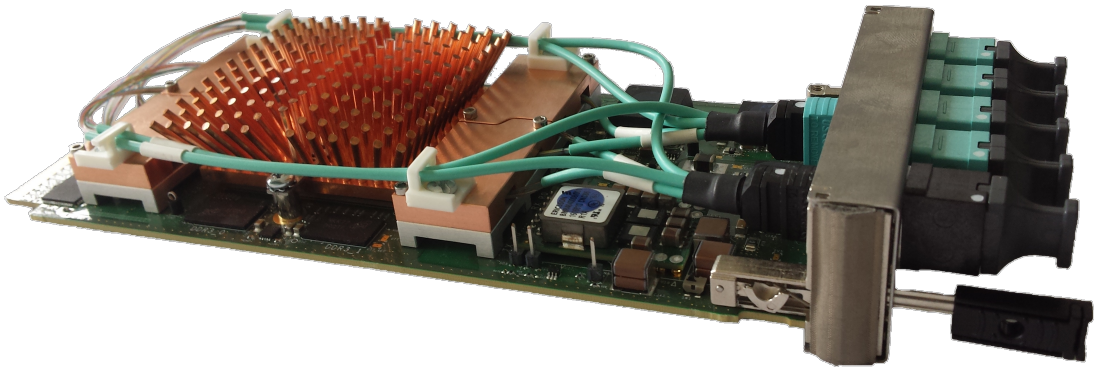

Figure 11 – A close-up view of a LATOME board, showing the heat sink, the four MTP-12 optical input connectors (green), and the MTP-48 output connector (black).

The LATOME firmware must meet latency constraints imposed by the ATLAS trigger system. All data alignment, energy computation, and output delivery occur within the fixed latency window required for the L1 trigger decision.

The next section describes the front-end electronics, which digitize the analog signals from the calorimeter and deliver the Super Cell data to the LATOME boards for processing.

3.5 Front-End Electronics

The front-end electronics are responsible for digitizing the analog signals generated by the LAr calorimeter and transmitting the digitized Super Cell (SC) data to the back-end system. This task is performed by the Liquid Argon Trigger Digitizer Boards (LTDBs), which are located on-detector in the radiation environment.

Each LTDB receives up to 320 analog SC signals sampled at 40 MHz. The signals are digitized with 12-bit resolution using custom radiation-hard ADCs. After digitization, the SC data is serialized and transmitted optically to the back-end electronics via 48 optical output links operating at 320 MHz.

In addition to the digital output for the upgraded trigger system, the LTDBs maintain backward compatibility by providing summed analog signals to the legacy Trigger

Tower-based system. This dual-output architecture allows for a smooth transition between the old and new trigger paths during commissioning and initial data-taking phases.

The LTDBs are designed with radiation-tolerant components to ensure stable operation in the high-radiation environment of the ATLAS calorimeter. Each board interfaces directly with the back-end LATOME boards hosted in the LAr Digital Processing Blade (LDPB), delivering the digitized SC data for real-time processing in the Level-1 trigger system.

The next section describes the timing and synchronization infrastructure that ensures coherent data transfer between the front-end and back-end electronics.

3.6 Timing and Synchronization

Precise timing and synchronization are essential for the operation of the LAr trigger readout system, ensuring that Super Cell (SC) data is correctly aligned to the corresponding bunch crossing (BC) and processed within the strict latency constraints of the Level-1 (L1) trigger.

The timing distribution is managed by the Trigger, Timing, and Control (TTC) system, which provides a global 40 MHz clock derived from the LHC machine clock. This clock synchronizes both the front-end and back-end electronics, aligning data sampling, digitization, and processing across the entire readout chain.

In addition to the clock signal, the TTC system distributes several critical timing commands:

- **Bunch Counter Reset (BCR)** — resets the bunch counter at the start of each LHC orbit to ensure consistent bunch crossing identification.
- **Event Counter Reset (ECR)** — resets the event counter, maintaining unique event identifiers within a run.
- **Level-1 Accept (L1A)** — signals the acceptance of an event, initiating the transfer of selected data to the readout system.

These signals are distributed optically to both front-end and back-end electronics using Gigabit Transceiver (GBT) links. The LTDBs use the 40 MHz clock to sample and digitize analog SC signals synchronously with the LHC bunch crossings. The back-end LATOME boards receive the same TTC-derived clock and commands, using them to correctly deserialize, realign, and process the incoming SC data with deterministic latency.

This timing infrastructure ensures that data from the same bunch crossing is consistently associated throughout the readout chain, from the calorimeter front-end to the Level-1 Calorimeter Trigger (L1Calo) and the FELIX data acquisition system.

The next section describes the monitoring paths that enable real-time validation of the trigger data and ensure system integrity during operation.

3.7 Monitoring and Data Validation

In addition to its primary role in providing real-time energy information to the Level-1 (L1) trigger, the LAr trigger readout system incorporates dedicated monitoring paths. These paths are essential for verifying data integrity, validating energy reconstruction, and ensuring the overall performance of the system during operation.

Two main monitoring streams are implemented in the back-end electronics:

- **Global Monitoring Path** — A Full Mode data stream from the LATOME to the Front-End Link eXchange (FELIX). This stream includes both the raw digitized ADC values and the computed transverse energy (E_T) for each Super Cell. It allows real-time validation of the energy reconstruction performed in the firmware by comparing the raw inputs with the processed outputs.
- **Ethernet-Based Monitoring Path** — A separate path connected via a 10 Gigabit Ethernet (GbE) link from the LATOME to a dedicated monitoring server. This path provides flexible, user-configurable access to internal signals, firmware debug information, and status registers, offering a complementary tool for online debugging and validation.

The Global Monitoring Path is crucial for ensuring consistency between the online trigger data and the offline reconstructed data. By capturing both the ADC inputs and the processed E_T values, it allows the system to detect potential discrepancies caused by firmware errors, hardware faults, or data corruption.

The Ethernet-Based Monitoring Path enhances the system's operability by offering access to diagnostic information without interfering with the main trigger or data acquisition streams. This includes features such as firmware counters, data alignment status, error flags, and live snapshots of internal processing blocks.

These monitoring infrastructures are critical not only for detector commissioning and firmware development but also for long-term operations, providing the tools necessary to maintain data quality and system reliability under the demanding conditions of high-luminosity LHC running.

This chapter has detailed the architecture of the Liquid Argon trigger readout system, from signal generation in the calorimeter to real-time data processing in the back-end LATOME boards. While this overview focused on the hardware and data flow, the focus of this thesis resides in the LATOME firmware itself.

The next chapter presents the internal architecture of this firmware, describing how it handles data de-serialization, alignment, energy calculation, and output formatting. It also discusses the evolution of the firmware from version 6.0 to 6.3, highlighting the design changes and upgrades that directly motivated the validation work described in this thesis.

4 LATOME Firmware

This chapter describes the internal architecture of the LATOME firmware, focusing on its functional blocks and how data is received, processed, and transmitted. Each block is responsible for a specific stage in the data path, from input processing to output formatting.

The discussion begins with an overview of the firmware's functional responsibilities and data flow organization, followed by a detailed description of its clock domains and processing pipeline. The firmware is structured into four functional blocks: the Input Stage (IS), Remap (ISM), User Code (UC), and Output Summing (OSUM), each responsible for a distinct step in the data path. In addition, support components manage monitoring, slow control, and timing synchronization.

The chapter also details the characteristics of the legacy firmware (Version 5), providing essential context for understanding the subsequent High-Level Synthesis (HLS)-based upgrades. This technical foundation is essential for understanding the firmware and the validation strategies discussed in the following chapters, particularly the comprehensive evolution and validation of firmware versions 6.0 to 6.3 which are presented in Chapter 6.

4.1 Firmware Functional Overview

The LATOME firmware performs real-time processing of Super Cell data in the ATLAS LAr calorimeter trigger readout chain. It receives digitized input from 48 high-speed optical links connected to the Liquid Argon Trigger Digitizer Boards (LTDBs), processes the data for each LHC bunch crossing, and delivers transverse energy (E_T) values to the Level-1 Calorimeter Trigger (L1Calo) Feature Extractors (FEXes) and the Trigger and Data Acquisition (TDAQ) system.

The main data path is implemented through four sequential processing stages:

1. **Deserialization and De-scrambling:** The firmware converts serialized 12-bit ADC data from each optical input into parallel data streams, aligns them temporally, and prepares them for processing.

2. **Geometry-Based Remapping:** The incoming SC data is reorganized into logical groupings that reflect the detector geometry. This remapping step simplifies downstream processing and aligns the data layout with the expectations of the FEX algorithms.

3. **Transverse Energy Calculation and Bunch Crossing Tagging:** Digital filtering techniques are applied to each SC to compute its transverse energy (E_T), while

associating each measurement with the correct bunch crossing identifier (BCID), synchronized to the LHC's 25 ns cycle.

4. **Output Summing and Formatting:** The processed E_T values are optionally summed (for jFEX and gFEX paths), encoded, and formatted into structured data frames. These are then transmitted at fixed latency to the L1Calo and FELIX systems.

In addition to the main trigger path, which transmits real-time E_T data to the L1Calo, the LATOME firmware also supports a parallel monitoring and data acquisition (TDAQ) path. This path provides raw and processed SC data for event readout, diagnostics, and validation.

To ensure reliable operation, the firmware includes additional support modules for:

- **Clock Management and TTC Decoding:** Ensures alignment with the LHC timing structure by decoding the 40 MHz clock and control signals (BCR, ECR, L1A) distributed by the ATLAS TTC system.
- **Slow Control Interface:** Enables remote configuration, register access, and status monitoring via the IPbus protocol.
- **Monitoring Infrastructure:** Allows continuous access to internal signals and data streams through dedicated Ethernet and FELIX-based links, facilitating real-time validation and debugging.

These auxiliary systems are integrated alongside the main data path and are critical for maintaining synchronization, enabling configuration and control, and supporting continuous performance validation across all operating conditions.

4.2 Firmware Architecture

The LATOME firmware is organized into four functional blocks, each corresponding to a specific stage in the processing pipeline. These blocks operate under a pipelined architecture to meet the fixed latency and high-throughput constraints of the ATLAS Level-1 trigger system.

Figure 12 shows the high-level organization of the firmware, with the main data path spanning from the LTDBs to the Feature Extractor (FEX) output links. The main data path is structured as follows:

- **Input Stage (IS):** Deserializes and de-scrambles 12-bit ADC samples from 48 optical input links at 320 MHz. It aligns the SC data temporally and ensures synchronization across channels.

- **Configurable Remapping:** Reorganizes the data received from the Input Stage (IS) at 320 MHz, mapping Super Cell information into Trigger Towers according to the geometry of the detector for each bunch crossing. The reorganized data is then transferred to the User Code block operating at 240 MHz. Given that the spatial arrangement of Super Cells varies across the detector, the remapping block is configured individually for each of the 116 LATOMEs during initialization.
- **User Code (UC):** This block processes the Super Cell data provided by the REMAP stage at 240 MHz. It computes the transverse energy (E_T) for each Super Cell using digital filters and appends the corresponding Bunch Crossing ID (BCID). The results, together with the associated data quality bits, are forwarded to the OSUM block with fixed latency for each bunch crossing.
- **Output Summing (OSUM):** The final aggregation stage of the LATOME processing chain. It receives, at 240 MHz, the transverse energy values and quality bits produced by the User Code, and performs regional summations according to the detector granularity required by the trigger system. These sums are then encoded, formatted, and transferred to the Feature Extractor (FEX) output fibers at 280 MHz. Similar to the REMAP, its configuration is defined during initialization for each of the 116 LATOMEs based on the detector geometry.

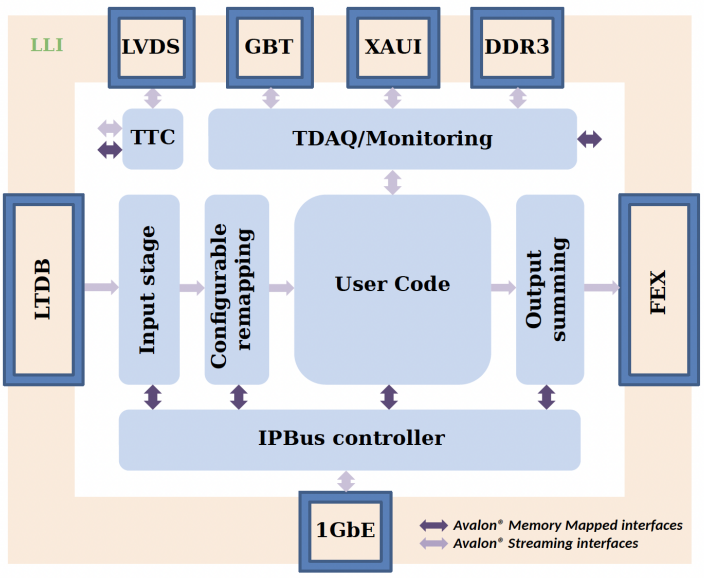

Figure 12 – LATOME firmware block diagram. The pale brown frame corresponds to the hardware interface, while the blue boxes represent higher-level functional blocks [6].

Figure 13 illustrates the three clock domains used in the LATOME firmware, which separate the high-speed input stage, the processing logic, and the output serialization. This organization reflects the functional requirements of the trigger path and ensures proper synchronization between stages.

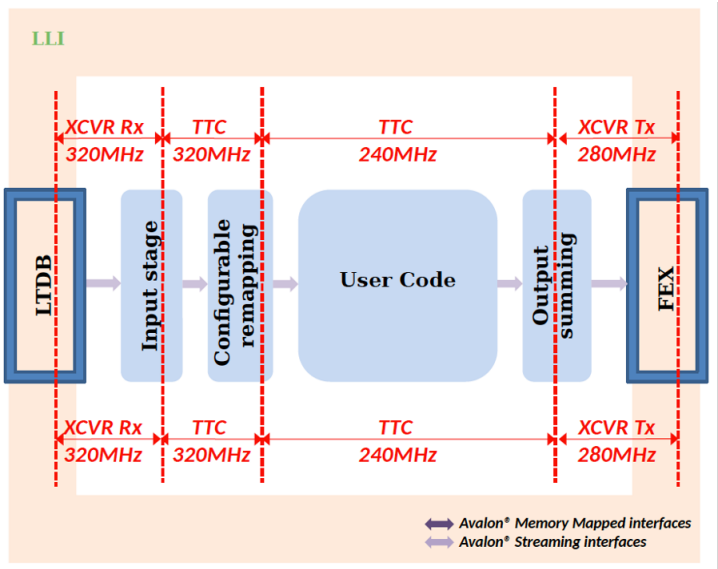

Figure 13 – LATOME firmware clock domains block diagram [6].

- **Input Stage at 320 MHz:** This domain handles the deserialization and descrambling of 12-bit ADC data from the 48 optical input links. Each Super Cell sample, produced at 40 MHz, is serialized across eight 320 MHz cycles to accommodate the required input throughput.
- **Processing Stage at 240 MHz:** Once deserialized and realigned, data enters the 240 MHz domain for energy reconstruction and bunch crossing tagging. Operating at a lower frequency simplifies routing within the FPGA fabric and reduces logic congestion while maintaining throughput. Internally, the number of streams expands from 48 to 62 to reflect the remapping of Super Cells into Trigger Towers.
- **Output Stage at 280 MHz:** The final output is transmitted to the Feature Extractor (FEX) subsystems using an 11.2 Gbps protocol. This requires data to be encoded and sent over seven LHC clock cycles (40 MHz) using seven 32-bit words per frame, resulting in a 280 MHz transmission clock.

The boundaries between these domains are managed using dedicated clock domain crossing (CDC) mechanisms to preserve timing determinism and data integrity. The entire pipeline is fixed-latency and fully synchronized with the 40 MHz LHC bunch crossing clock, ensuring alignment with the correct event window.

In parallel to the main path, the firmware includes auxiliary modules essential for control and validation:

- **Monitoring (MON):** Provides access to internal data via FELIX or Ethernet for diagnostics and validation.

- **Slow Control (IPCTRL):** Manages configuration and control using the IPbus protocol.
- **Timing, Trigger, and Control (TTC):** Decodes LHC clock and control signals (BCR, ECR, L1A) for system-wide synchronization.
- **Low-Level Interface (LLI):** Serves as the integration layer between hardware logic and the configuration/monitoring infrastructure.

This architectural foundation establishes the basis for the firmware evolution and validation efforts discussed in the following sections.

4.3 Legacy Design: Version 5

The current operational LATOME firmware, referred to as Version 5 (v5), was developed entirely using VHDL. In this design, data remains serialized throughout its processing path, with clock domain transitions occurring within the logic blocks. This time-multiplexed architecture contributes to the complexity of the firmware. For instance, the remapping stage exemplifies these challenges, spanning three distinct steps and multiple clock domains to align, reorder, and process the data before delivering it to the User Code block.

As shown in Figure 14, the remapping stage begins with input synchronization at 320 MHz, followed by data reordering and multiplexing. The data enters the remapping stage through 48 physical input links, each delivering 8 consecutive words per bunch crossing, totaling 384 input words. In the final step of the remapping stage, the clock frequency transitions from 320 MHz to 240 MHz, reducing the number of words per frame from 8 to 6. This results in 62 output streams, totaling 372 words. The reduction from 384 to 372 accounts for the suppression of inactive or empty SC inputs, which are filtered out during the remapping process.

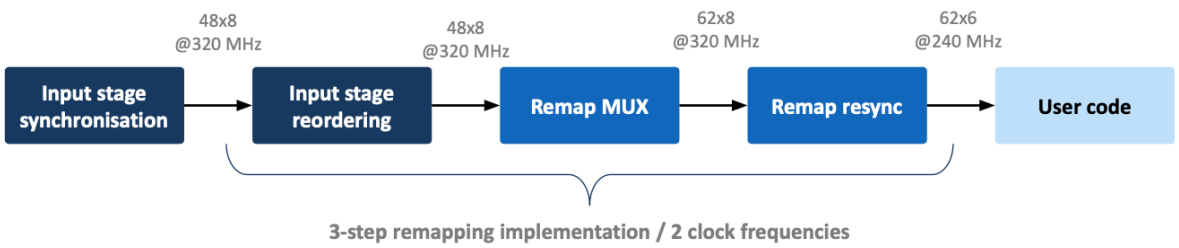

Figure 14 – Three-step remapping implementation in the LATOME v5 firmware.

The LATOME firmware must operate its internal blocks at multiples of 40 MHz to maintain synchronization with the LHC rate of collisions. At the initial remapping stage,

each data frame consists of 8 words processed at 320 MHz. As the data progresses through the design, the number of words per frame reduces to six, corresponding to a frequency of 240 MHz. The FEX systems, however, expect frames consisting of seven 32-bit words, which requires a frequency of 280 MHz for the output data protocol.

The timing diagram in Figure 15 shows these operational frequencies, where eight, six, and seven words are processed at 320, 240, and 280 MHz, respectively.

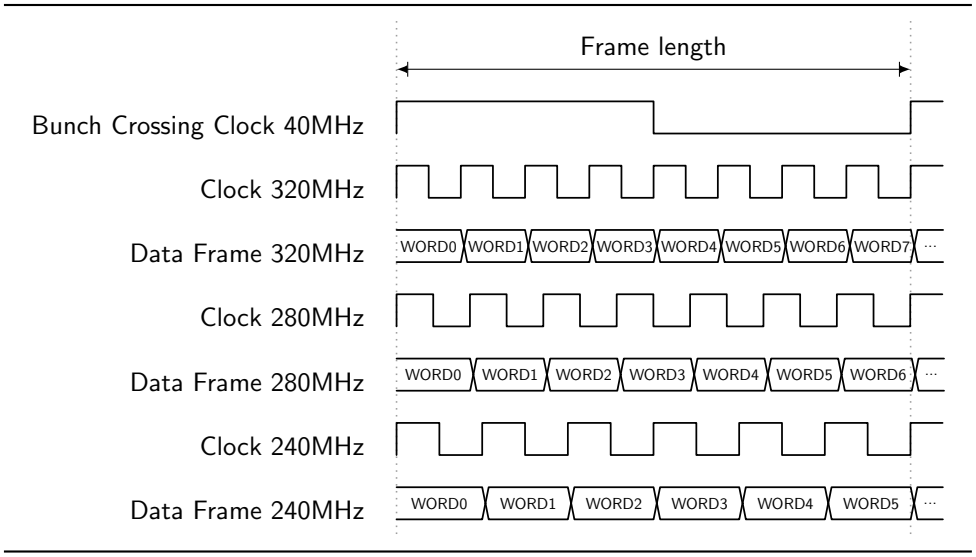

Figure 15 – Timing diagram illustrating the clock frequencies and word configurations in the v5 LATOME firmware [23].

While being functional, the v5 firmware has timing violations that became evident after eight years of development. These violations are a common challenge in FPGA designs and occur when the design fails to meet timing constraints, such as setup and hold times for flip-flops. Setup time refers to the period required for an input to stabilize before the clock edge, while hold time refers to the period needed to remain stable after the clock edge. Failing to meet these constraints can lead to metastability, causing unpredictable behavior, as documented in [23], which references the metastability model introduced in [24].

In addition to timing challenges, the v5 firmware exhibits high FPGA resource utilization. For instance, maintaining 62 Processing Units (PUs) in the User Code block to process the 62 data streams output from the remapping stage significantly increases resource consumption. The serialized nature of the design and the need for multiple clock domain transitions further exacerbate the complexity of maintaining timing closure. These architectural choices also hinder maintainability and increase the effort required for debugging and upgrading the firmware.

To address these challenges, a new LATOME firmware architecture based on High-Level Synthesis (HLS) was proposed by Dr. Marcos Vinicius Silva Oliveira. The

redesigned firmware replaces the time-multiplexed processing architecture with a parallel logic structure, making use of switch matrices to manage configurable routing. HLS offers a higher level of abstraction, reducing development time and simplifying design iterations. By optimizing resource usage and addressing timing performance, the new architecture sets the foundation for future firmware improvements.

5 Methodology and Tools

This chapter is dedicated to outlining the essential tools and methodologies that underpinned the implementation, optimization, and verification of the LATOME firmware upgrade. It commences by introducing Field-Programmable Gate Arrays (FPGAs), which serve as the fundamental reconfigurable hardware platform, highlighting their benefits for rapid prototyping and iterative improvements due to their reprogrammability and massive parallelism.

Following this, High-Level Synthesis (HLS) is explored, detailing its role in enabling hardware functionality description using high-level languages like C++ to reduce development effort and enhance design flexibility, with the Siemens Catapult HLS tool central to generating synthesizable RTL code from these descriptions. Additionally, the chapter then presents a HLS Wrapper Generator for VHDL Integration, a custom Python-based tool developed to streamline the integration of top-level HLS modules by restructuring flattened Catapult HLS ports into more modular, array-based VHDL interfaces for improved simulation and ease of use.

A novel switch matrix architecture is subsequently introduced as a foundational design pattern, vital for enhancing data routing and synchronization within the firmware by replacing sequential operations with a single-step, unified clock domain approach, thereby simplifying timing closure.

The discussion then moves to the chosen verification approach, elaborating on Coroutine-Based Co-Simulation with Cocotb and QuestaSim for comprehensive functional testing, enabling concurrent and asynchronous tasks, internal signal probing, and integration with Python libraries for accelerated debugging. Finally, the chapter concludes by detailing the overarching multi-layered simulation and validation strategy, which employs both firmware-agnostic and firmware-aware models across unit-level, block-level, and full-system simulations to ensure the correctness and robustness of the upgraded design across various levels of integration.

5.1 Field-Programmable Gate Arrays (FPGAs)

Field-Programmable Gate Arrays (FPGAs) are reconfigurable integrated circuits that allow digital logic circuits to be defined and modified by the user after manufacturing. Unlike Application-Specific Integrated Circuits (ASICs), which have fixed functionality, FPGAs offer reprogrammability and flexibility, making them ideal for evolving requirements, rapid prototyping, and iterative improvements.

FPGAs are composed of a matrix of programmable logic blocks interconnected through configurable routing channels. These blocks implement combinational and sequential logic, while modern FPGAs also include embedded memory, digital signal processing

(DSP) units, and high-speed transceivers.

In the ATLAS experiment, FPGAs are central to the Level-1 trigger system and the LAr calorimeter readout, where real-time processing of detector signals is essential. Their massive parallelism enables operations on hundreds of data channels simultaneously, while their low-latency response supports the stringent 25 ns bunch-crossing cycle of the LHC.

Crucially, their reconfigurability allows firmware upgrades, enabling performance improvements without replacing hardware. This is particularly important for systems such as LATOME, which must evolve in parallel with detector upgrades and higher luminosities. To fully leverage this potential, the LATOME upgrade employs High-Level Synthesis to streamline design and validation.

5.2 High-Level Synthesis (HLS)

As FPGA systems grow in complexity, traditional Register Transfer Level (RTL) design using VHDL or Verilog becomes increasingly difficult to manage. HLS addresses this challenge by allowing developers to describe functionality in high-level languages such as C, C++, or SystemC. The synthesis tool then converts this high-level algorithmic description into RTL, applying optimizations for area, latency, and throughput.

With HLS, the designer focuses on functionality rather than hardware-level timing, clocking, and signal control. The tool's ability to automate the generation of control logic, pipelining structures, and datapath elements typically leads to performance levels comparable to hand-written RTL while significantly reducing development time.

One of the main advantages of HLS is rapid design iteration. Architectural explorations, trade-offs, and optimizations can be evaluated by modifying high-level code, instead of rewriting low-level RTL. The LATOME firmware benefits from this flexibility, especially in the transition to fully parallel processing architectures.

In the LATOME upgrade, the tool used for HLS is Catapult, developed by Siemens EDA. Catapult supports fine-grained control of synthesis through directives (pragmas) that influence loop unrolling, pipelining, array partitioning, and resource sharing.

Among these directives, loop unrolling is particularly relevant to the LATOME firmware. Unrolling duplicates the hardware for each iteration of a loop, enabling multiple iterations to be executed in parallel instead of sequentially. This optimization increases throughput at the cost of higher resource utilization, and is especially well suited for workloads that process independent data streams, such as the parallel treatment of Super Cell inputs. When applied with a full unroll, every iteration of the loop is mapped to a distinct hardware structure, producing a fully parallel architecture.

Beyond its architectural benefits, full unrolling also has a practical advantage in

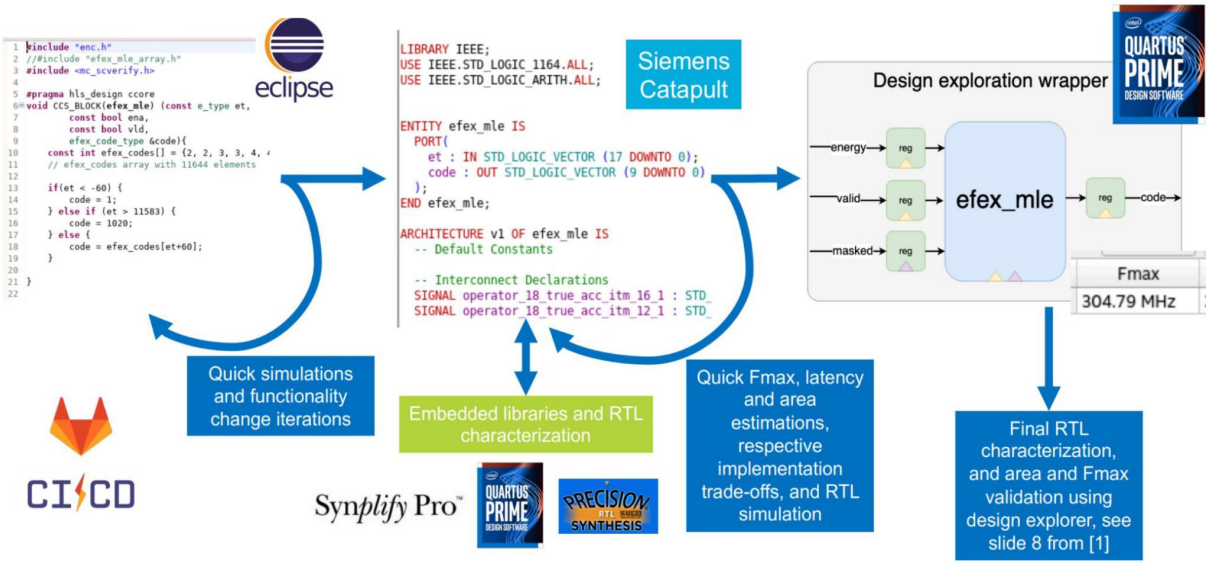

Figure 16 – Design flow from C++ to VHDL using Siemens Catapult.

simulation. Since each loop iteration corresponds to an explicitly generated signal in the synthesized RTL, all signals become directly visible and traceable in waveform inspection tools. This significantly improves observability during debugging and validation campaigns, allowing engineers to monitor each data path individually and verify that the remapping and summing logic behave as intended. In the LATOME HLS workflow, this feature was instrumental in correlating C++ models with the generated VHDL code and RTL-level simulations in Cocotb.

The tool translates C++ into synthesizable VHDL, abstracting control logic while preserving timing behavior. As shown in Figure 16, this process simplifies the transition from algorithm to hardware, with accurate latency modeling and efficient resource mapping.

In the context of LATOME, Catapult facilitates rapid prototyping of components such as the switch matrices used in the REMAP and OSUM blocks. These blocks benefit from HLS's ability to manage multiple data streams concurrently, enforce latency constraints, and minimize logic utilization. Coupled with dedicated verification, the Catapult workflow ensures that generated RTL is both performant and reliable.

5.3 Wrapper Generator for HLS–VHDL Integration

To facilitate the integration of top-level HLS modules into the LATOME firmware simulation environment, a custom graphical tool was developed using Python and the `tkinter` library. This utility generates VHDL wrapper files that restructures flattened Catapult HLS ports into array-based interfaces, making them more modular, readable, and easier to integrate into existing simulation infrastructure.

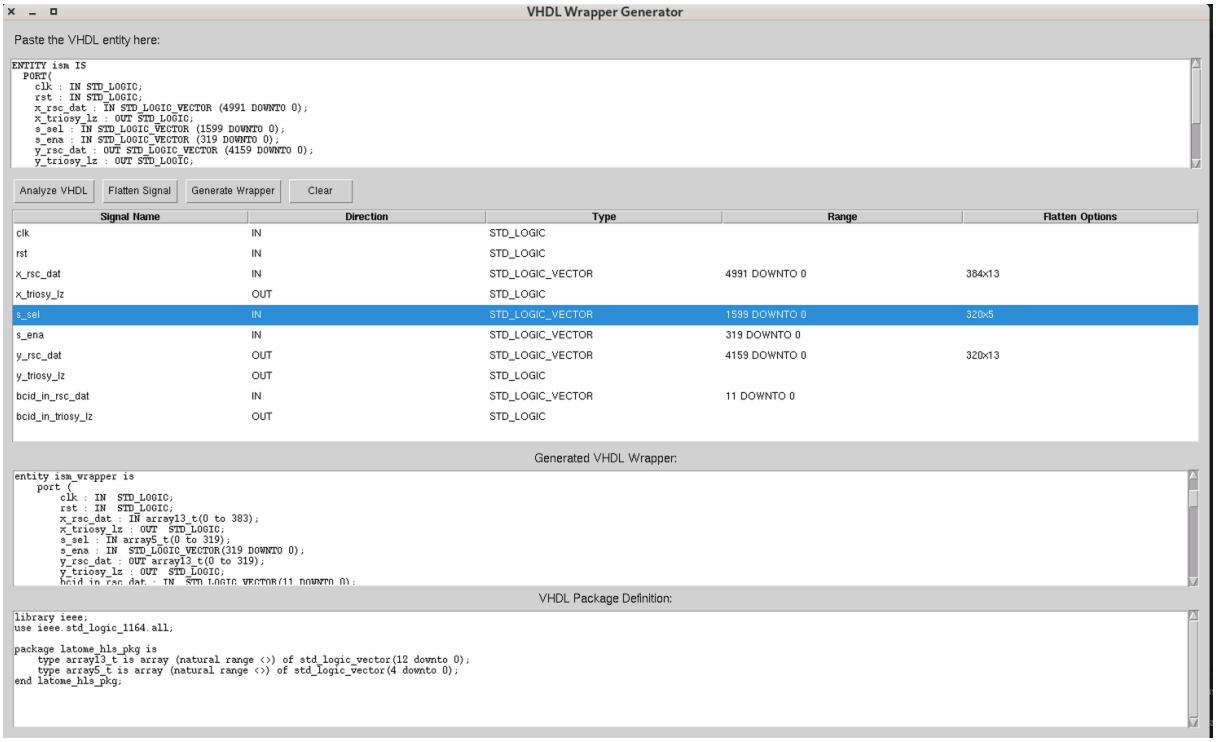

Figure 17 – VHDL Wrapper Generator GUI. The user pastes the Catapult-generated VHDL entity (top), selects flattened ports (middle), and receives a new wrapper entity and package (bottom) for easier signal handling.

Catapult HLS, when translating C++ code to VHDL, flattens arrayed ports into a single wide `std_logic_vector`. For example, a C++ input defined as an array of N elements, each with B bits, such as `input[N]`, is synthesized into the following VHDL signal:

$$\texttt{flat\_signal} : \texttt{std\_logic\_vector}(N \cdot B - 1 \text{ downto } 0) \tag{5.1}$$

While this format is functionally correct, it complicates testbench development and debugging, as the logical structure of the original array is lost. To restore modularity, the tool generates wrapper entities that reinterpret these wide vectors as VHDL arrays of `std_logic_vector`, following the inverse mapping:

$$\texttt{array\_signal} : \texttt{array}(0 \text{ to } N - 1) \text{ of } \texttt{std\_logic\_vector}(B - 1 \text{ downto } 0) \tag{5.2}$$

To support this structure, the tool also produces a reusable VHDL package with custom type definitions:

$$\texttt{type arrayB\_t is array}(\texttt{natural range } <>) \texttt{ of std\_logic\_vector}(B - 1 \text{ downto } 0) \tag{5.3}$$

For instance, `array13_t` defines an array of 13 elements, while `array5_t` defines an array of 5 elements.

The flattening and unflattening logic is implemented automatically using `generate` statements in VHDL. The assignment rule for each array index i is:

$$\texttt{flat\_signal}[(i+1)\cdot B-1 : i\cdot B] \leftarrow \texttt{array\_signal}(i) \tag{5.4}$$

The GUI abstracts this process: the user pastes the Catapult-generated entity, selects ports and their bit-widths, and receives a fully generated wrapper entity with a supporting package, ready for integration into the LATOME simulation framework.

This methodology was used for the top-level HLS blocks ISM and OSUM, where flattened ports needed to be reconstructed to interact with standard VHDL structures. Internal HLS modules, which operate directly on scalar or small vector signals, do not require this conversion.

By automating this process, the wrapper generator eliminates repetitive manual coding, reduces the risk of indexing errors, and improves the readability of simulation waveforms, all without introducing any additional logic or resource overhead. The full Python source code and representative VHDL output are provided in Appendix A.

5.4 Switch Matrix Architecture

A major innovation of the HLS-based LATOME firmware is the introduction of configurable switch matrices, which replace the sequential remapping logic of the legacy design with a fully parallel interconnection fabric. A switch matrix is a programmable structure that connects multiple inputs to multiple outputs in parallel, according to a user-defined mapping that reflects the geometry of the calorimeter region.

In the legacy firmware, the remapping process relied on sequential transfers across multiple clock domains. In contrast, the switch matrix introduced in version 6 executes all remapping in a single cycle within a unified clock domain, avoiding time-multiplexed operations and reducing synchronization complexity.

Figure 18 illustrates the principle. Each black dot denotes a possible programmable connection between an input and an output stream. Only the subset of connections required by the detector mapping is instantiated, resulting in a sparse implementation that conserves FPGA resources while maintaining full flexibility.

This architecture delivers input data to the correct output in a single cycle, simplifying timing closure and minimizing latency. This architecture delivers input data to the correct output in a single cycle, simplifying timing closure and minimizing latency. It provides a uniform interface that facilitates testing, reuse, and potential partial reconfiguration. Beyond these immediate benefits, the switch matrix abstraction makes the design scalable to different calorimeter regions and adaptable to evolving firmware requirements.

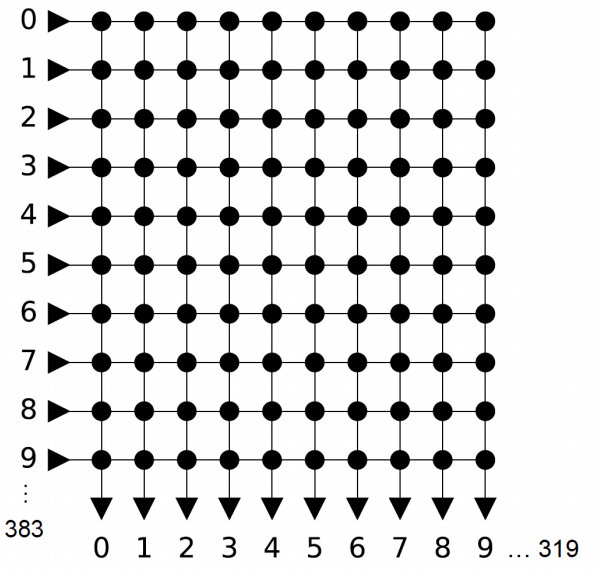

Figure 18 – Diagram of the switch matrix architecture in the REMAP block. Black dots indicate programmable interconnections between input and output streams.

5.5 Coroutine-Based Co-Simulation with Cocotb and QuestaSim

Simulation plays a dual role in the LATOME workflow. On the one hand, it ensures that the VHDL generated by HLS remains consistent with its C++ source. More importantly, it validates that the firmware implementation as a whole complies with the functional specifications of the trigger system, such as correct remapping, summing behavior, latency constraints, and error handling. To achieve this, the LATOME project adopts a coroutine-based co-simulation strategy combining Cocotb and QuestaSim.

Cocotb is a Python framework that allows testbenches to be written in Python rather than VHDL or Verilog. Its coroutine-based execution model enables test code to behave like concurrent hardware processes, while remaining readable and concise. Multiple asynchronous tasks—such as input stimulation, output monitoring, and internal probing—can be executed in parallel, improving observability and accelerating debugging.

The interaction between Cocotb and QuestaSim is illustrated in Figure 19, where the Python scheduler coordinates multiple coroutines that probe and drive the VHDL design through the simulator interface.

This setup supports:

- Functional testing under nominal and edge-case scenarios.
- Probing of internal signals and registers during runtime.
- Comparison against reference models for bit-accurate validation.
- Result analysis and visualization through Python tooling.

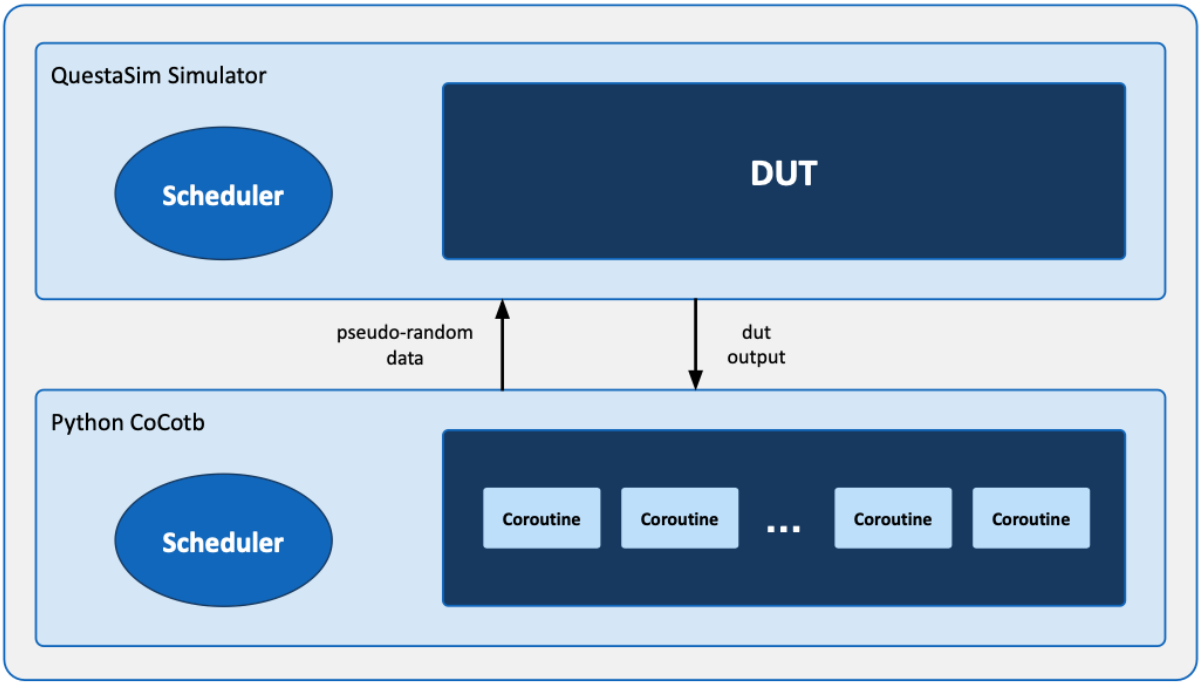

Figure 19 – Interaction between Cocotb and QuestaSim, showcasing coroutine-based simulation and probing of multiple circuit points.

Although QuestaSim is limited to functional verification and does not provide post-synthesis metrics such as timing closure or FPGA resource utilization, it is essential for establishing behavioral correctness. Low-level implementation insights are obtained later through synthesis and place-and-route tools such as Vivado.

5.6 Simulation and Validation Strategy

Given the complexity of the upgraded firmware and the reliance on HLS-generated RTL, a structured simulation and validation strategy is necessary to confirm that the implementation behaves as intended. The LATOME project therefore adopts a modular and hierarchical methodology that balances early debugging efficiency with full-system coverage. While early firmware versions were validated mainly through direct stimulus and waveform inspection, later versions—starting from v6.2—introduced a layered approach based on firmware-aware and firmware-agnostic models to enable more systematic verification.

- **Modular DUT Design:** Each firmware component (e.g., REMAP, ISM, OSUM) is encapsulated and simulated independently. This helps isolate design errors, simplifies debugging, and allows reuse of targeted testbenches.
- **Layered Testing:** Verification progresses through multiple stages: unit-level tests for isolated functionality, block-level integration for interface verification, and full-

system simulations for end-to-end behavior. This stepwise approach makes it possible to build confidence progressively.

- **Dual Modeling Strategy:** Two complementary reference models support validation:
  - The *firmware-agnostic model* checks high-level functionality based on input-output relationships.
  - The *firmware-aware model* reproduces the bit-level transformations of the HLS-generated design, allowing detailed comparisons.

  Cross-checking results between the models and the DUT improves coverage and reduces the likelihood of undetected discrepancies.

- **Full-System Validation:** After component-level checks, the entire firmware is simulated in system mode with realistic inputs and timing. These runs confirm synchronization, latency alignment, and compliance with output formatting.

This methodology verifies the firmware implementation against its specifications before hardware deployment and provides the basis for the validation results discussed in the following chapter.

6 Firmware Evolution and Validation: Versions 6.0 to 6.3

This chapter describes the evolution of the LATOME firmware from Version 6.0 to Version 6.3, outlining the architectural modifications introduced in each release together with the corresponding simulation and validation strategies. The shift from a time-multiplexed to a fully parallelized architecture required a redesign of key components, most notably through the adoption of switch matrices and the modularization of functional blocks.

To assess behavior and stability during this transition, a layered simulation methodology was progressively introduced. Beginning with Version 6.2, this approach combined firmware-aware and firmware-agnostic models, enabling validation at both bit-level and behavioral levels.

Each version is examined both in terms of architectural intent and validation methodology, providing a chronological narrative of design improvements and compliance with the performance and timing requirements of the ATLAS LAr trigger system.

6.1 Overview of the Firmware Evolution

The transition from the legacy LATOME firmware (v5) to the upgraded series of High-Level Synthesis (HLS)-based designs marked a substantial architectural and methodological shift. While Version 5 remained operational throughout Run 3, it relied on time-multiplexed logic distributed across three distinct clock domains, which introduced persistent challenges in timing closure, maintainability, and scalability.

The HLS initiative addressed these limitations by adopting a fully parallel architecture centered on configurable switch matrices. Implemented in C++ and synthesized with the Siemens Catapult HLS tool, this redesign enabled a more modular and resource-efficient firmware structure, while simplifying timing integration. The upgrade unfolded through a series of incremental firmware versions, each introducing and validating essential elements of the new design.

A major advancement was the replacement of time-multiplexed processing stages in the REMAP block with a parallel Input Switch Matrix (ISM). The ISM consolidates the routing logic into a single processing step, where Super Cell signals are directed through a sparse switch matrix that eliminates sequential multiplexing. This approach removes the overhead of serial data transfers, reduces synchronization complexity, and better aligns with the constraints and strengths of HLS-based synthesis. The new architecture also preserved compatibility with the broader LATOME firmware ecosystem, including downstream modules and monitoring tools.

In parallel, the OSUM block was redesigned to adopt the same parallel processing

paradigm. While the internal structure of OSUM is examined in later sections, it is important to note that this transformation was central to achieving an HLS-based, high-throughput data pipeline. Together, these changes established the foundation for a production-ready architecture and enabled the introduction of structured simulation and validation frameworks, refined progressively in subsequent versions.

The HLS-based implementation began with a proof-of-concept known as the demonstration firmware (versions 6.0.0 and 6.1.0). These releases introduced the ISM block and maintained compatibility with the legacy monitoring infrastructure using a transitional Readout Switch Matrix (RSM). The RSM reversed the effects of the new mapping, enabling test flows without modifying the byte-stream decoder used by the ATLAS Athena software. In version 6.1.0, the RSM was removed once the software mapping was updated to recognize the HLS output format. These early versions focused exclusively on the monitoring path and were not yet intended for full trigger path validation. Their main purpose was to demonstrate that HLS-generated components could be integrated and simulated within the LATOME ecosystem.

Version 6.2.0 marked the transition from validating the monitoring path to exercising the full trigger path, connecting the REMAP and OSUM blocks into a cohesive processing chain. This release introduced support for the eFEX data path and was the first to adopt a layered simulation methodology. Validation was carried out using a combination of firmware-aware and firmware-agnostic models, enabling detailed verification of bit-level behavior, timing alignment, and bunch crossing synchronization through cross-comparison at multiple abstraction levels.

In version 6.3.0, this architecture was extended to support the jFEX path. The internal design remained consistent, allowing reuse of the same simulation infrastructure with only minor adaptations. This release demonstrated the scalability of the architecture and confirmed that the processing pipeline could accommodate different data sources without loss of timing alignment or functional behavior.

Although not covered in this thesis, version 6.4.0 is expected to finalize the 6.x series by integrating the gFEX path. Future developments—beginning with version 7.x—will focus on the HLS reimplementation of the User Code block, completing the migration to a fully synthesizable HLS architecture across the entire LATOME trigger path.

6.2 Demonstration Firmware Integration and HLS Validation (v6.0.0 and v6.1.0)

The first step in the HLS-based upgrade of the LATOME firmware was the demonstration firmware (versions 6.0.0 and 6.1.0), designed to test the feasibility of integrating HLS blocks while preserving compatibility with the existing monitoring infrastructure. These releases focused exclusively on the `MON` (monitoring) path, with a complete redesign of the configurable remapping logic as their central architectural change. Figure 20

illustrates the architecture of the legacy monitoring path prior to HLS integration.

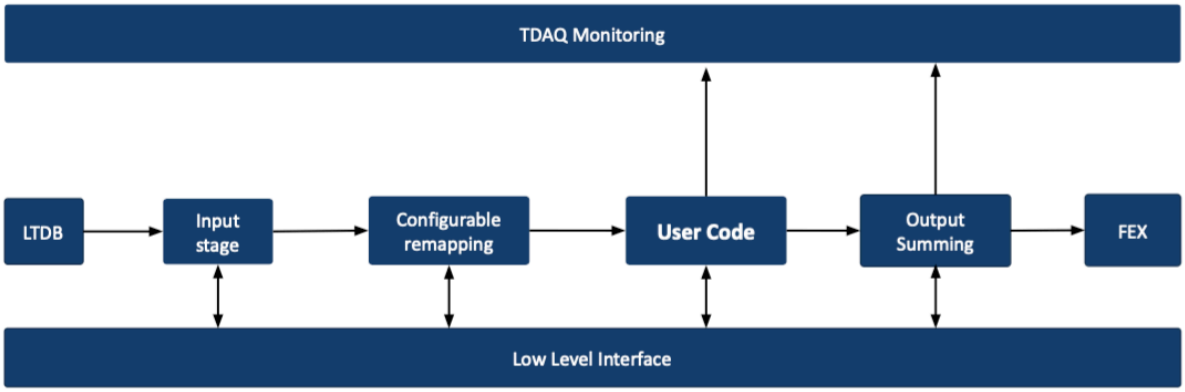

Figure 20 – Monitoring path dataflow in the legacy LATOME firmware (v5).

This design, though stable in operation, was based entirely on hand-written VHDL and implemented using time-multiplexed logic across multiple clock domains. These constraints posed recurring challenges in timing closure and limited flexibility for scaling or refactoring the design as requirements evolved.

To evaluate the feasibility of HLS within the LATOME firmware, version 6.0.0 introduced a modified monitoring path architecture, shown in Figure 21. The legacy Configurable Remapping block was replaced by the ISM HLS IP, designed and synthesized with the Siemens Catapult HLS tool. To maintain compatibility with the legacy monitoring infrastructure—particularly the byte-stream decoding routines in Athena—a transitional block called the *Readout Switch Matrix* (RSM) was added. The RSM reverted the ISM output format back to the structure expected by downstream software, allowing validation and monitoring to proceed without modifications to the software stack.

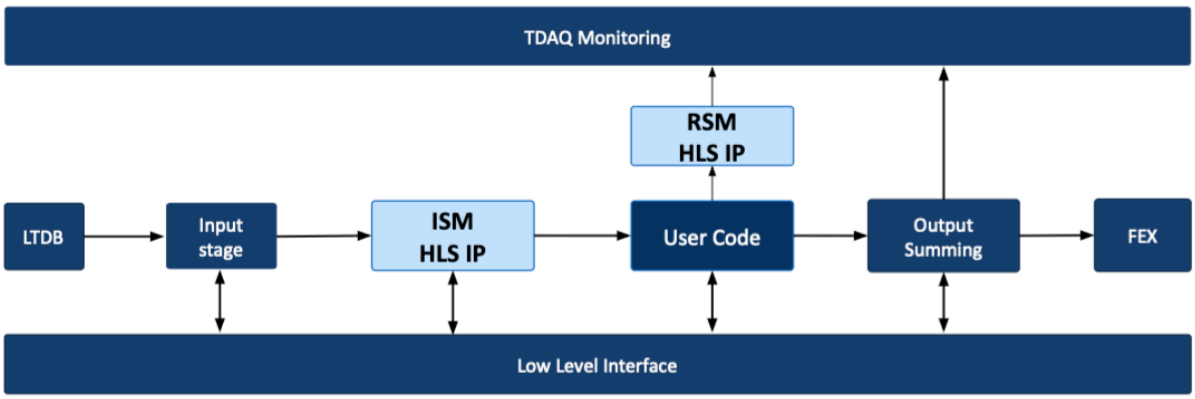

Figure 21 – Demonstration firmware architecture (v6.0.0), with the legacy remapping replaced by an HLS-based Input Switch Matrix (ISM).

6.2.1 S2P Converter and Clock Domain Crossing

Figure 22 shows the high-level architecture of the ISM HLS IP. A total of 48 identical S2P converters operate in parallel—one per input stream—enabling deserialization and alignment of the incoming data. Once transferred into the 240 MHz domain, the ISM block performs the remapping using static multiplexers, and the P2S block reorders the output to match the User Code format.

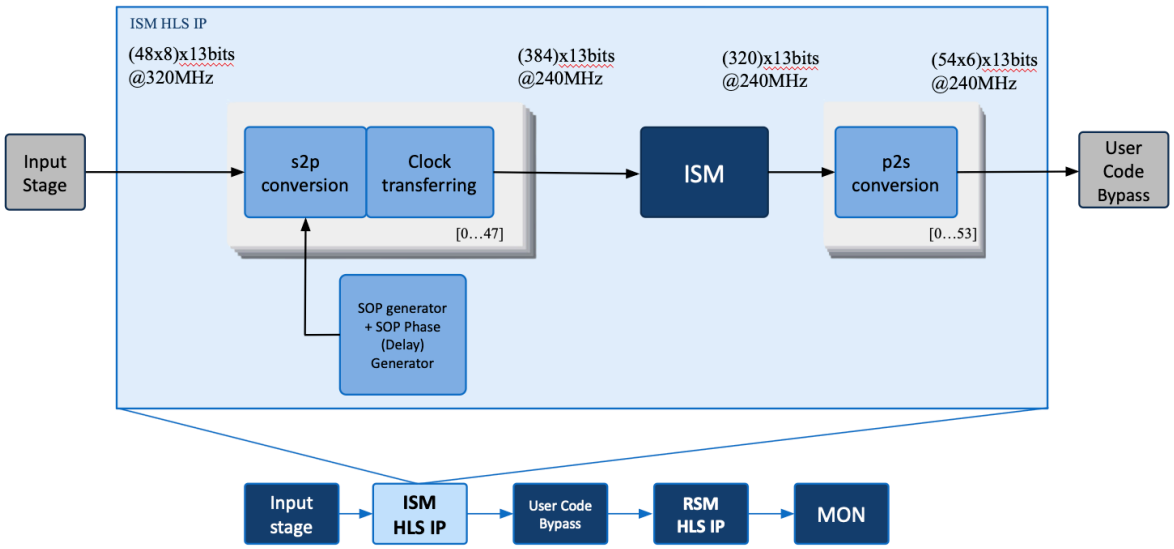

Figure 22 – Block diagram of the ISM HLS IP. Each input passes through an S2P converter with clock transfer, followed by the parallel switch matrix (ISM) and the P2S converter.

Each S2P module receives a continuous stream of 13-bit serialized Super Cell data at 320 MHz. Over eight input cycles (25 ns), these words are buffered and presented as a parallel frame of eight values synchronized to the 240 MHz domain. In effect, each S2P delivers one complete parallel frame per 240 MHz cycle, as illustrated in Figure 23.

The clock transfer relies on a dedicated enable signal, denoted as `valid_240`, which acts as the latch command for capturing the buffered 320 MHz data into the 240 MHz shadow registers. Rather than representing data validity, this signal serves as the strobe that guarantees the safe handover of one complete frame per bunch crossing. The precise position of this strobe is defined by a phase selection circuit, avoiding sampling near metastability regions.

The alignment of `valid_240` is governed by the SOP generator and SOP phase generator, shown in Figures 24–26. The SOP generator produces a periodic reference pulse marking the start of each bunch crossing. The SOP phase generator then introduces a programmable delay, allowing the strobe to be shifted within the transfer window until a stable capture point is reached. Extensive validation campaigns demonstrated the

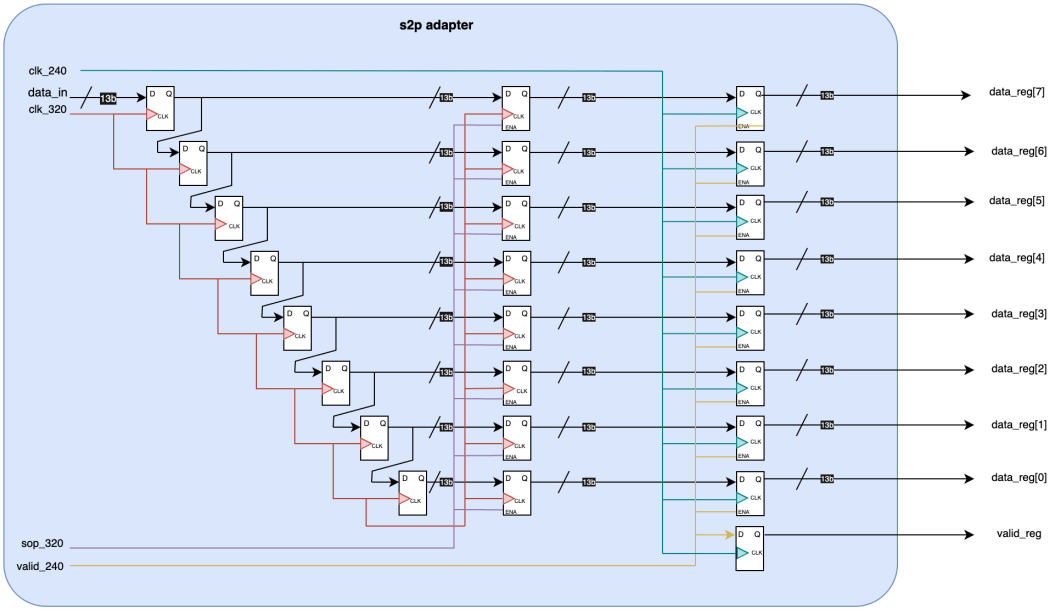

Figure 23 – Internal structure of a Serial-to-Parallel (S2P) converter with clock transfer, bridging 320 MHz serialized data to the 240 MHz domain.

criticality of selecting an optimal delay setting for reliable operation.

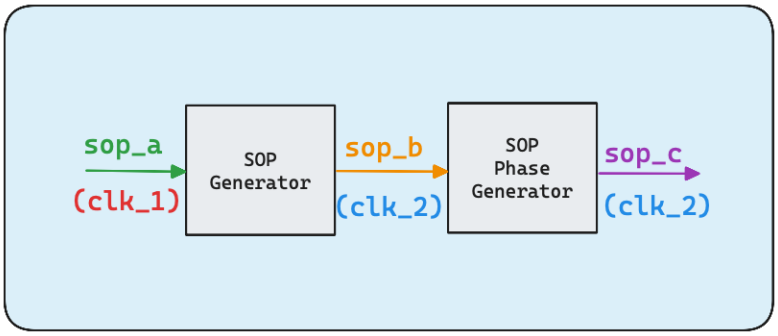

Figure 24 – Connection between the SOP generator and the SOP phase generator, providing synchronization and phase selection for the S2P modules.

During initialization, an additional control known as `sop_refresh` can be enabled to periodically realign `valid_240` with every incoming SOP. This guarantees that any phase errors or jitter at startup are corrected. Once the optimal delay configuration has been identified, `sop_refresh` is disabled, leaving the internal counter to generate `valid_240` autonomously and ensuring stable operation without disturbances from external SOP fluctuations.

Together, the S2P converter, `valid_240` strobe, SOP generator, SOP phase generator, and optional `sop_refresh` form the backbone of the data synchronization strategy. This design reliably transfers all 48 input streams into the 240 MHz processing domain, enabling the ISM to perform remapping with consistent timing. Validation confirmed

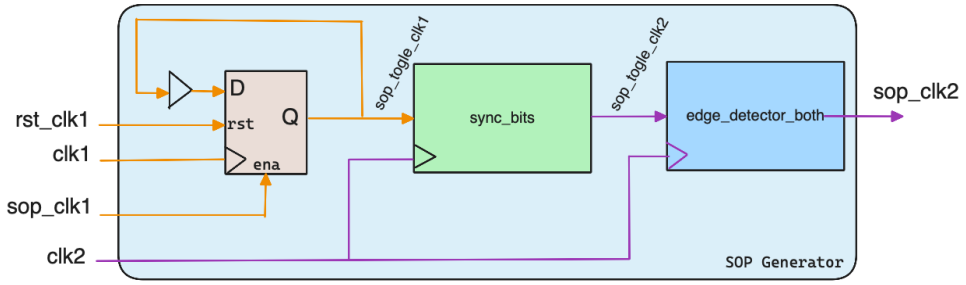

Figure 25 – SOP generator: produces a reference pulse that marks the start of each bunch crossing and distributes it to the S2P converters.

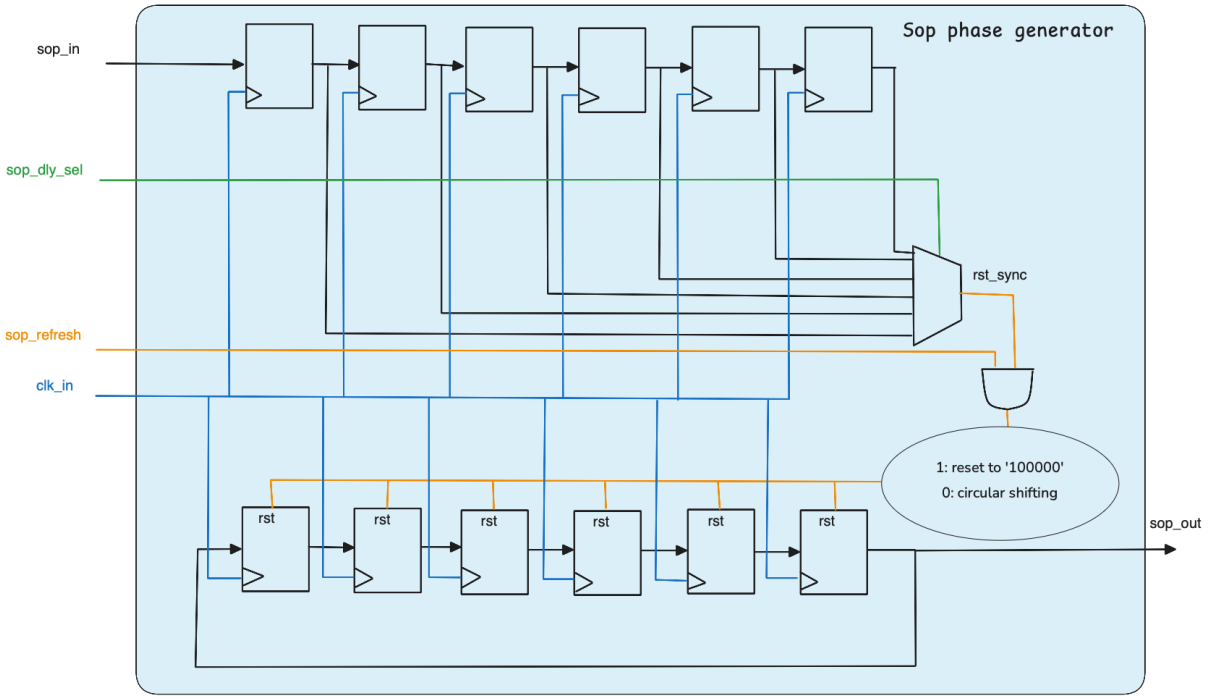

Figure 26 – SOP phase generator: introduces a programmable delay to the reference pulse, ensuring that data capture in the 240 MHz domain occurs at a stable and controlled instant.

stable operation across all tested phase settings and refresh strategies.

6.2.2 The Input Switch Matrix (ISM)

The ISM is responsible for routing the 13-bit Super Cell data across predefined output positions according to a detector-specific mapping. In the demonstration firmware, the ISM receives a total of 384 parallel SC inputs per bunch crossing while working in the 240 MHz domain.

Internally, the ISM is structured as an array of 320 instances of two cascaded multiplexers. Each instance corresponds to one output stream and selects its input based on a configurable index and an optional `enable` bit. The first multiplexer selects which of

the 384 deserialized input positions will be routed to the output, while the second acts as a conditional gate: if the `enable` signal is deasserted, the selected value is masked to zero. The mapping between the LATOME input streams and the first-stage multiplexer inputs is fixed and identical for all configurations; only the selection indices and enable flags are updated to reflect each detector region's layout. This flexible architecture allows the ISM to implement region-specific mappings while maintaining full parallelism across the data path. The precise mapping configurations were defined during the LATOME HLS system-level studies, which fall outside the scope of this thesis.

Figure 27 shows the internal composition of the ISM multiplexer structure, with the two cascaded stages allowing flexible and configurable routing.

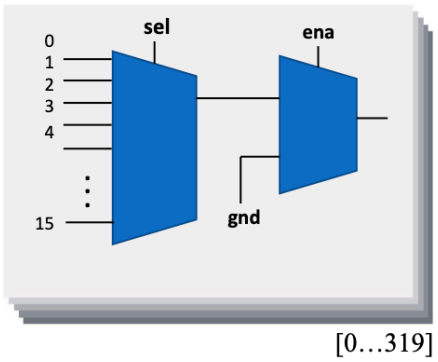

Figure 27 – Structure of the ISM highlighting the two cascaded multiplexers.

The choice of having exactly 320 outputs is not arbitrary—it reflects the interface expectations of the downstream `User Code` block, which processes the remapped Super Cell data. This expectation, in turn, arises from a comprehensive analysis of official ATLAS detector mappings. For any valid region, the number of active (non-zero) Super Cell inputs routed through the LTDBs into the LATOME never exceeds 320. This bound results from the physical partitioning of the detector and the topology of the LATOME input fibers. The ISM was therefore dimensioned with 320 output channels to match the processing capacity of the `User Code`, ensuring compatibility across all detector regions and eliminating the need for dynamic resizing or complex routing strategies.

Table 1 summarizes the number of enabled and disabled Super Cell inputs for several representative detector mappings. This analysis demonstrates that, even in the densest configurations, 320 parallel outputs suffice to capture all relevant information without omission.

As an illustrative case, the full mapping for the EMBA_1 region is included in Appendix B, showing the complete correspondence between input fibers and Super Cell identifiers. This concrete example confirms the rationale behind the 320-output ISM design and reinforces its universality across the LAr calorimeter front-end.

Table 1 – Active Super Cell inputs per region according to the official LATOME input mapping files.

Region	**Enabled SC Inputs**	**Disabled SC Inputs**
EMBA_1	320	64
EMBA_EMECA_1	304	80
EMECA_1	296	88
EMECA_HECA_1	288	96
FCAL1A	192	192
FCAL2A	176	208

By supporting static remapping for all valid configurations, the ISM enables deterministic routing and significantly simplifies validation, synthesis, and integration workflows within the HLS-based LATOME firmware.

6.2.3 Parallel-to-Serial (P2S) Adapter

To interface the output from the ISM at 240 MHz with the input of User Code block — which expects serialized, time-sliced input streams — the P2S adapter performs a controlled serialization of the data.

Figure 28 shows the internal architecture of one of the 54 P2S modules. Each module receives six 13-bit parallel words from the `User Code` block, updated once per bunch crossing. These inputs are fed directly into a 6-to-1 multiplexer, which selects one word per clock cycle, synchronized to the 240 MHz processing clock. This controlled selection produces a serialized 13-bit output stream across six consecutive cycles.

At the heart of the adapter is a counter that increments every clock cycle and drives a multiplexer, selecting one word at a time from data_reg[i]. The counter is reset at the start of each transmission cycle using the valid_reg signal, which marks the beginning of a new valid packet from the User Code.

This structure also includes a 5-cycle delay register (z^5) that aligns the valid_reg signal with the sixth and final word of the group. This delayed signal is used to assert the final valid output, which indicates to downstream modules that the serialized packet has been completed and can be latched or forwarded.

Two pipeline registers are placed at the output of the multiplexer: one for the 13-bit data path and another for the corresponding valid flag. These help balance combinational delays and ensure synchronous data propagation across the design.

6.2.4 RSM HLS IP

To validate the integration of HLS-generated modules in the LATOME firmware without modifying the existing software infrastructure, the Readout Switch Matrix (RSM)

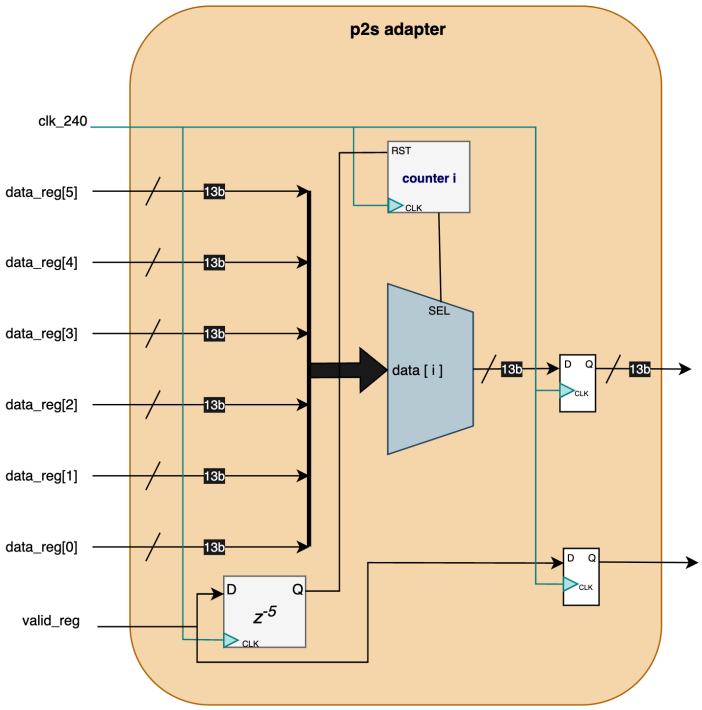

Figure 28 – Architecture of the P2S adapter used to serialize six 13-bit words from the User Code output.

was introduced as a transitional block. Synthesized entirely using the Catapult HLS tool, the RSM reverts the remapped Super Cell data back to the format expected by the existing byte-stream decoder in Athena. This makes it possible to verify the correct behavior of the ISM HLS IP and the full data flow through the firmware, without requiring any changes on the software side.

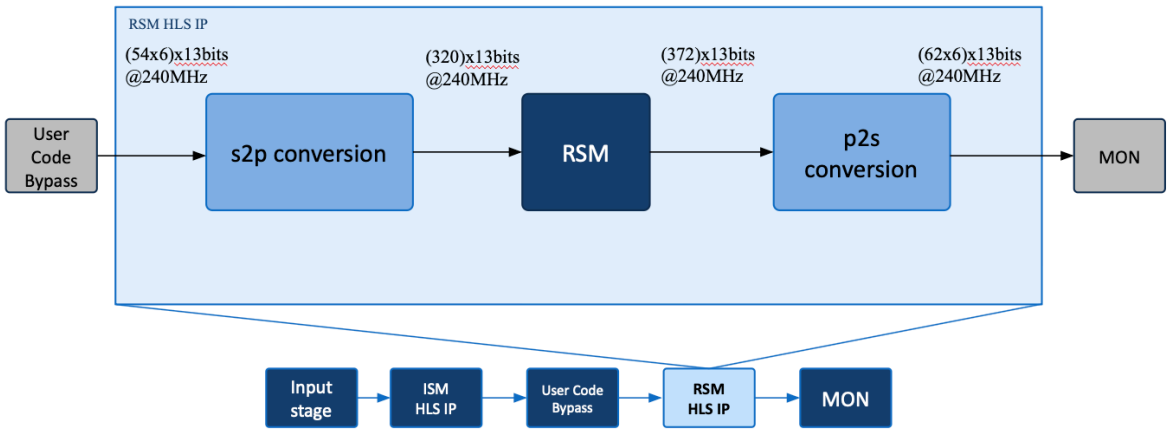

Figure 29 – Architecture of the RSM HLS IP. The S2P and P2S modules match those in the ISM, but operate within a single clock domain.

As shown in Figure 29, the RSM replicates a structure similar to the ISM HLS

IP, with a parallel data processing pipeline built around **Serial-to-Parallel (S2P)** and **Parallel-to-Serial (P2S)** modules. However, unlike the ISM, there is no need for clock domain crossing since both the input and output data operate at the same 240 MHz frequency.

Each RSM output stream is reconstructed using only the 13-bit transverse energy information produced by the User Code. While the full Super Cell format includes 75 bits (e.g., for saturation, quality, and metadata), the demonstration firmware intentionally limits the scope to 13 bits per Super Cell. This decision avoids the additional complexity and area cost of serializing full-width outputs—particularly since the RSM is meant only for transitional testing and will be removed in version 6.1.0

The RSM thus serves as a proof-of-concept module to demonstrate that the HLS framework can be successfully embedded in the LATOME firmware, enabling validation workflows while keeping downstream tools unchanged.

In summary, the demonstration firmware (v6.0.0) successfully replaced the legacy remapping infrastructure with an HLS-based implementation. The ISM HLS IP introduced a fully-parallel architecture for region-specific remapping, while the RSM HLS IP enabled seamless integration with existing software tools by reverting the processed data to its original format. This transitional architecture proved that HLS modules could be embedded in the LATOME firmware without disrupting downstream workflows.

With the RSM removed in version 6.1.0, and the Athena decoder updated to decode the native output of the new HLS pipeline, the system became fully HLS-integrated on the monitoring path. The next step was to validate its functional correctness and timing behavior through targeted simulation campaigns. These validation strategies and results are detailed in the next subsection.

6.2.5 Validation Strategy and Test Campaigns

The validation of the demonstration firmware (v6.0.0 and v6.1.0) was carried out through a comprehensive simulation campaign combining C-level simulations, Python-based testbenches, and hardware tests performed at the laboratory. These tests aimed to validate the intended functionality of the newly integrated HLS components.

Initially, the ISM logic was verified at the functional level using C++ simulations. As illustrated in Figure 30, a 384-input array of 13-bit Super Cell values was passed to a pure C++ model of the ISM algorithm. The output was then checked against the expected result derived from the LATOME mapping configuration files. These tests provided a rapid functional check of the logic and were later referred to as **Layer 0** of the simulation framework. However, this setup did not test the generated RTL code, nor did it simulate any timing behavior or clock domain intricacies.

To address this limitation, the validation workflow transitioned to a Python-based

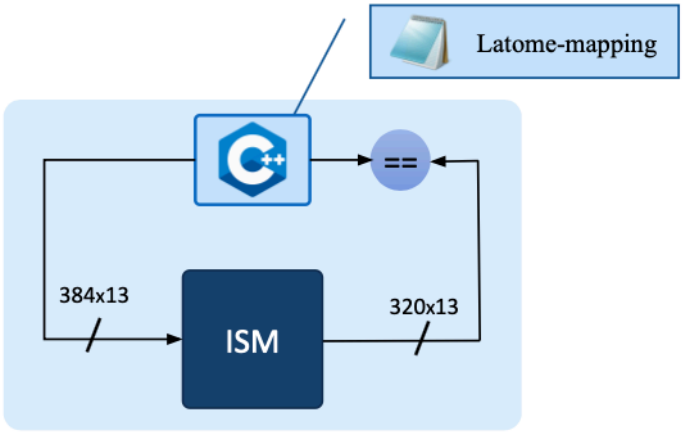

Figure 30 – Layer 0: Functional verification of the ISM logic using a C++ implementation and mapping files.

testbench built with the Cocotb framework. This second environment, shown in Figure 31, allowed the direct simulation of the RTL generated by the Siemens Catapult HLS tool. Specifically, the ISM VHDL description was automatically synthesized from the same C++ source code used in the initial functional tests. Unlike the C++ simulation—which only evaluated logic correctness at the algorithmic level—these Cocotb-based tests operate at the VHDL, enabling full verification of the synthesized hardware under a realistic clocked environment.

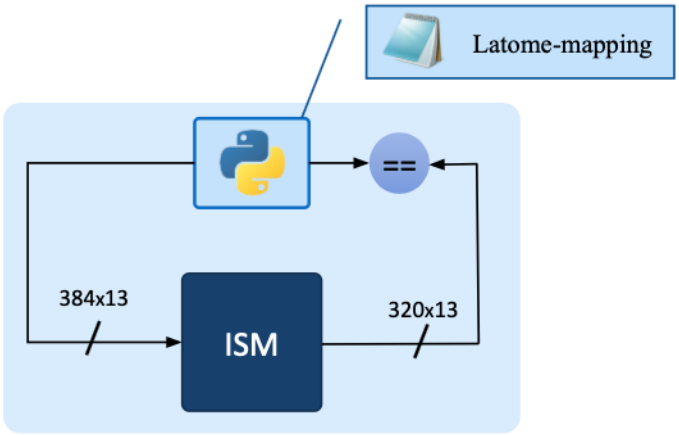

Figure 31 – Python-based RTL validation of the ISM block using Cocotb. Tests include signal-level monitoring of the VHDL output.

A key motivation for adopting Cocotb was the inability of the Siemens SCVerify framework to simulate modules that perform **clock domain crossing**, such as the S2P converter. Consequently, while C++ tests remained useful for initial functional validation, all deeper verification activities moved to Cocotb.

The transition to Cocotb-based simulation leveraged the VHDL output generated by the Catapult HLS tool, with the LATOME VHDL wrapper generator playing a key role in facilitating the integration. As discussed in Section 5.3, this wrapper was particularly useful for adapting the flattened array interfaces produced by Catapult into more accessible signal structures for simulation.

Specifically, for the ISM block, Catapult generated a single flattened vector of $384 \times 13 = 4992$ bits to represent the entire input array. The wrapper unpacks this structure into 384 individual 13-bit signals, matching the expected interface for the ISM's input ports. While this transformation was not strictly required for functional validation, it greatly simplified the connection between testbench signals and the HLS-generated RTL, streamlining the simulation workflow.

The simulation environment was designed to support all 116 distinct LATOME mapping configurations required to cover the full geometric coverage of the ATLAS Liquid Argon Calorimeter. Each mapping defines a specific configuration of the `select` and `enable` signals that control the ISM's 320 output multiplexers, ensuring correct routing for the corresponding detector region.

For each mapping, randomized 13-bit values were assigned to all 384 input channels of the ISM and propagated over a large number of bunch crossings. To ensure reproducibility, the pseudo-random generator was seeded with a fixed value for each test run. The resulting outputs were automatically compared to the expected remapped values, derived from the corresponding LATOME mapping files. This systematic and high-coverage testing strategy provided strong assurance of the ISM block's correctness and stability, establishing a solid foundation for system integration and subsequent hardware validation.

6.2.5.1 Cocotb Simulation

To analyze the timing behavior of the ISM block, a waveform-based validation approach was implemented using the Cocotb simulation framework in combination with QuestaSim. Cocotb supports asynchronous test routines, where checking logic must explicitly wait for the appropriate simulation condition before verifying signal values. In our testbenches, the `bcid` (Bunch Crossing ID) signal was used as a synchronization hook. Since the ISM block has a fixed latency, the expected output timing can be inferred directly from the input cycle and the known propagation delay.

Figure 32 illustrates a typical waveform trace used in these simulations. It is possible to observe the synchronization between the input data and the `bcid_in`, as well as the alignment between the output `y` signal and the propagated `bcid_out` of the ISM block. This precise timing relationship enables robust validation routines that check signal correctness only when the output is valid.

A representative Cocotb synchronization routine is shown below:

```
await ReadOnly()
while dut.bcid_out.value != 0:
    await RisingEdge(dut.clk)
await RisingEdge(dut.clk)
```

Listing 6.1 – BCID-based output synchronization in Cocotb.

This structure guarantees that the output is checked exactly at the aligned latency point, preventing premature or invalid assertions. It also ensures reproducibility and stability in waveform-based validation workflows.

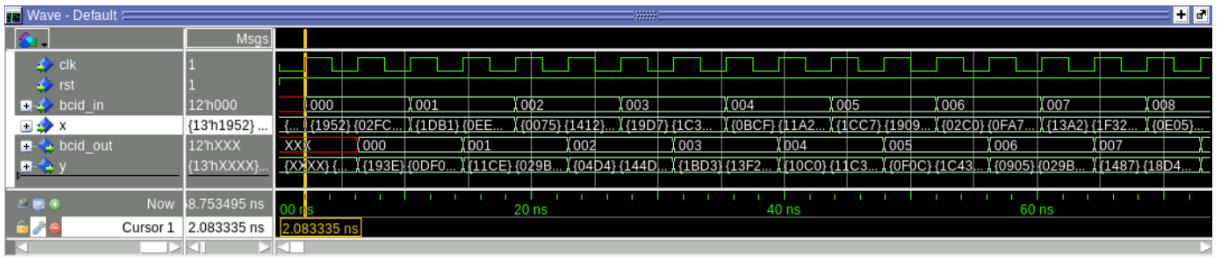

Figure 32 – Example waveform from Cocotb/QuestaSim showing synchronization using the `bcid` signal to validate output after the fixed latency of the ISM block.

6.2.5.2 Full Demonstration Firmware Simulation

After verifying the correctness of the ISM block in isolation, the validation campaign advanced to encompass the complete data processing pipeline shown in Figure 33. This testbench integrates all major functional blocks in the monitoring path: the Serial-to-Parallel (S2P) converter with clock transferring, the ISM HLS IP, the RSM HLS IP, and the final Parallel-to-Serial (P2S) adapter. Each of these components was instantiated using the RTL generated by the Siemens Catapult HLS tool and verified using the Cocotb simulation framework.

The configuration shown in Figure 33 implements the end-to-end dataflow from 48 input streams at 320 MHz to 62 output streams at 240 MHz, mirroring the actual operational setup of the LATOME firmware. This final test ensured that all interfaces and synchronization boundaries were respected, including the use of the intermediate 320×13 format from the ISM to the RSM and the final 62×6 stream serialization.

To further mimic the real firmware scenario, the full testbench also included a placeholder block representing the `User Code`. However, for the purpose of validating the dataflow between the ISM and RSM blocks, a simplified bypass mode was introduced. This bypass configuration—consisting solely of S2P and P2S adapters connected by a single-cycle delay—enabled direct comparison of ISM and RSM behavior without introducing

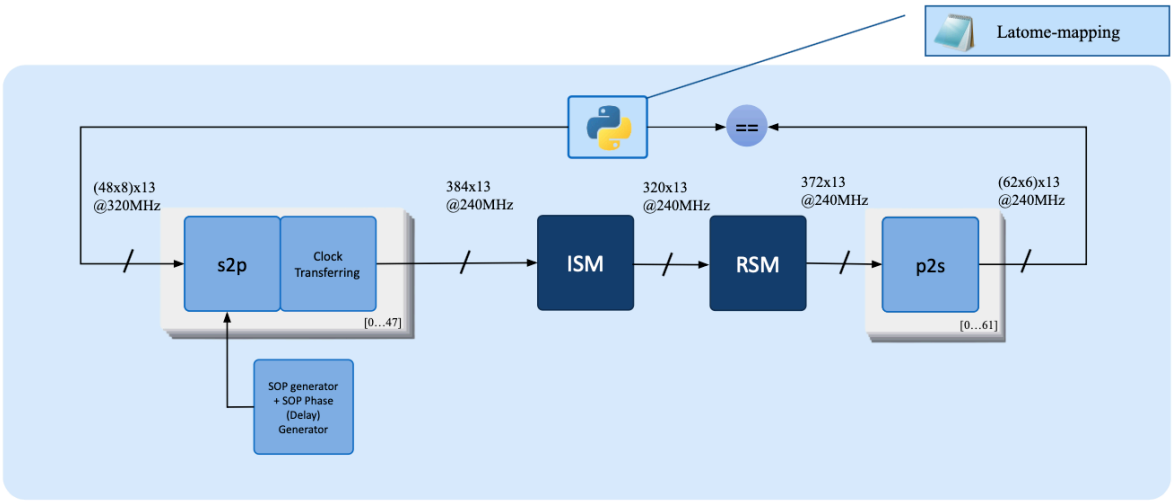

Figure 33 – Cocotb testbench for the full monitoring chain of firmware version 6.0.0. Includes S2P with clock transferring, ISM, RSM, and P2S blocks.

the algorithmic complexity of the actual processing logic. As shown in Figure 34, this setup allowed precise verification of the remapping and serialization pipeline in isolation.

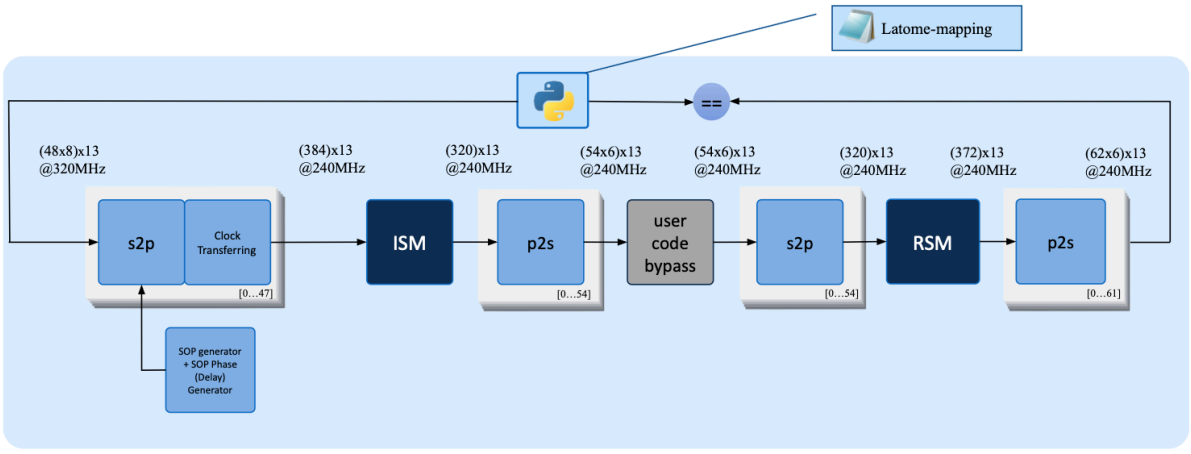

Figure 34 – Extended testbench including User Code bypass to directly validate ISM–RSM compatibility.

In simulation, the `sop_delay` parameter, which governs the clock transferring stage, exhibited no practical impact on data alignment. All `sop_delay` configurations successfully transferred data without timing violations, as expected in an ideal simulation environment devoid of clock jitter, routing delays, and metastability. However, in hardware, the precise setting of `sop_delay` is critical, and the corresponding tests are detailed in the next subsection.

6.2.5.3 Transition to Firmware Version 6.1.0

Following the successful validation of the full demonstration chain in version 6.0.0, the RSM block was removed in firmware version 6.1.0. This transition marked the final step toward full HLS integration in the monitoring path. The Athena decoder was updated to interpret the ISM output format directly, eliminating the need for reformatting and confirming the system's readiness for native HLS-based data handling.

Importantly, the simulation and validation framework remained unchanged, allowing seamless reuse of the testbenches and test scenarios. The verification strategies continued to operate directly on the ISM output, now decoded natively by the software stack. The next set of simulations thus focused on timing-critical behavior, in particular the validation of the `sop_delay` parameter within the clock domain transfer logic.

6.2.6 Clock Domain Transfer Validation: SOP Delay Tests

To ensure robust and stable clock domain transfer between the 320 MHz and 240 MHz domains, a comprehensive set of validation tests was performed using the target hardware. These tests focused on verifying the reliability of the `sop_delay` parameter, which defines the precise instant at which data is latched into the 240 MHz domain. This timing parameter is critical to avoid metastability and ensure deterministic data capture in the Serial-to-Parallel (S2P) converters.

The laboratory tests served two main objectives:

- Validate the correct functionality of the upgraded firmware on real hardware using the LATOME stand-alone test setup.
- Characterize the operational margin of the `sop_delay` parameter for multiple detector mappings under stress conditions.

6.2.6.1 Hardware Test Environment

All tests were performed using the LATOME standalone readout infrastructure, composed of the following components:

- **LATOME 40:** Configured as a pattern generator to emit predefined Super Cell signals.
- **LATOME 108:** The Device Under Test (DUT), running the upgraded firmware.
- **FELIX system:** Hosted on `pcemf-felix-03`, connected to both LATOMEs.
- **Test server:** `pc-emf-fw-03`, acting as the main interface for controlling test sequences.

- **IPbus access:** Managed through a control hub hosted on `pcemf-pm-02`, allowing remote register access and configuration.

6.2.6.2 Firmware v6.0.0 Test Results

For version 6.0.0, stress testing was performed using the EMBA_1 detector region mapping. Table 2 shows the total of 50 configuration cycles that were executed for each of the six `sop_delay` values (0 through 5) over a continuous period of 10 hours. Failures were observed only for `sop_delay` 0. The remaining delay values yielded stable and error-free operation.

`sop_delay_select`	EMBA_1 (%)
0	46.00
1	100.00
2	100.00
3	100.00
4	100.00
5	100.00

Table 2 – Percentage of success on complete tests for firmware version v6.0.0 using the EMBA_1 mapping (50 completed tests).

All other LATOME mapping families were tested over a 24-hour period using 10 configuration cycles per delay value. The results confirmed the same safe operating margin, as shown in Table 3.

`sop_delay_select`	EMBA_1	EMBC_1	EMBA_EMECA_1	EMBC_EMECC_1	EMECA_1	EMECC_1
0	55.56%	47.62%	50.00%	71.43%	58.82%	45.45%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
`sop_delay_select`	EMECA_HECA_1	EMECC_HECC_1	FCAL1A	FCAL1C	FCAL2A	FCAL2C
0	76.92%	50.00%	50.00%	52.63%	40.00%	40.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 3 – Percentage of success on complete tests for firmware version v6.0.0 across LATOME mappings.

6.2.6.3 Firmware v6.1.0 Test Results

Firmware version 6.1.0 removed the RSM HLS IP block, simplifying the data path to connect the User Code directly to the monitoring interface. This architectural change was enabled by updated Athena software mappings, which aligned the User Code outputs

with the expected readout format. The same validation tests were repeated to ensure that this simplification preserved functional correctness and stable behavior across bunch crossings.

The results for version 6.1.0 mirrored those obtained with version 6.0.0. For the EMBA_1 mapping, 50 configuration cycles per delay value were executed over a continuous period of 10 hours, as shown in Table 4. The remaining LATOME mappings were tested over 24 hours, with 10 cycles per delay value, and the results are summarized in Table 5. Identical behavior was observed across all tests, confirming the robustness and stability of the clock transfer mechanism.

sop_delay_select	EMBA_1 (%)
0	56.00
1	100.00
2	100.00
3	100.00
4	100.00
5	100.00

Table 4 – Percentage of success on complete tests for firmware version v6.1.0 using the EMBA_1 mapping (50 completed tests).

sop_delay_select	EMBA_1	EMBC_1	EMBA_EMECA_1	EMBC_EMECC_1	EMECA_1	EMECC_1
0	50.00%	60.00%	70.00%	30.00%	30.00%	40.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
sop_delay_select	EMECA_HECA_1	EMECC_HECC_1	FCAL1A	FCAL1C	FCAL2A	FCAL2C
0	40.00%	50.00%	30.00%	30.00%	20.00%	60.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 5 – Percentage of success on complete tests for firmware version v6.1.0 across LATOME mappings.

6.2.7 LATOME HLS Software Infrastructure

The validation campaigns for firmware versions 6.0.0 and 6.1.0 were supported not only by simulation and hardware tests, but also by a dedicated LATOME HLS software infrastructure. Developed in parallel with the firmware itself, this infrastructure provided a unified foundation for general configuration across all systems that use/interface with the LATOME firmware.

At its core, the system is based on structured input and output specification files that define the mapping between LATOME inputs and the corresponding Feature Extractor (FEX) outputs. From these specifications, the infrastructure produces all necessary configuration files required by the LDPB software — the production-level toolchain used to initialize and configure LATOMEs in the Phase-I system. It ensures alignment between firmware output and the decoding conventions adopted by the ATHENA software framework.

Figure 35 illustrates the full pipeline. The process begins with the IN and OUT mapping specifications, which are used in the LATOME HLS system level studies to produce files such as the configurations of the ISM `ism_mapping.csv`, `ism_enable.csv`, and `ism_settings.csv`. These drive the internal behavior of the ISM HLS IP. The same input mappings are used by the ISM & RSM mapping generator to create `pu_list_latome.txt` files — artifacts which describe the association between Super Cells and processing units at the monitoring level and is also used by other tools within the LAr software ecosystem, such as LAr Soup and LAr Skill.

This mapping is then converted into the firmware-readable `mon_mapping.txt` format, which defines how Super Cell data is organized and presented at the readout interface (the Mon block). To ensure software-level compatibility, the information is further translated into the `mon_mapping_athena.txt` format using Super Cell Online IDs extracted from the LArID database. This allows the same HLS-based data format to be fully compatible with the ATHENA decoder, ensuring identical treatment of data in both test benches and the operational data acquisition system.

With the validation of the initial firmware stages and the LATOME HLS software infrastructure firmly established, development proceeded toward firmware version 6.2.0. This version marked the beginning of the OSUM block implementation using High-Level Synthesis, with initial focus placed exclusively on the eFEX output path. Version 6.2.0 thus served as a transition point between foundational architectural validation and the integration of new HLS-based output logic. The following section presents the design strategy, integration details, and validation results associated with this first HLS implementation of the OSUM block.

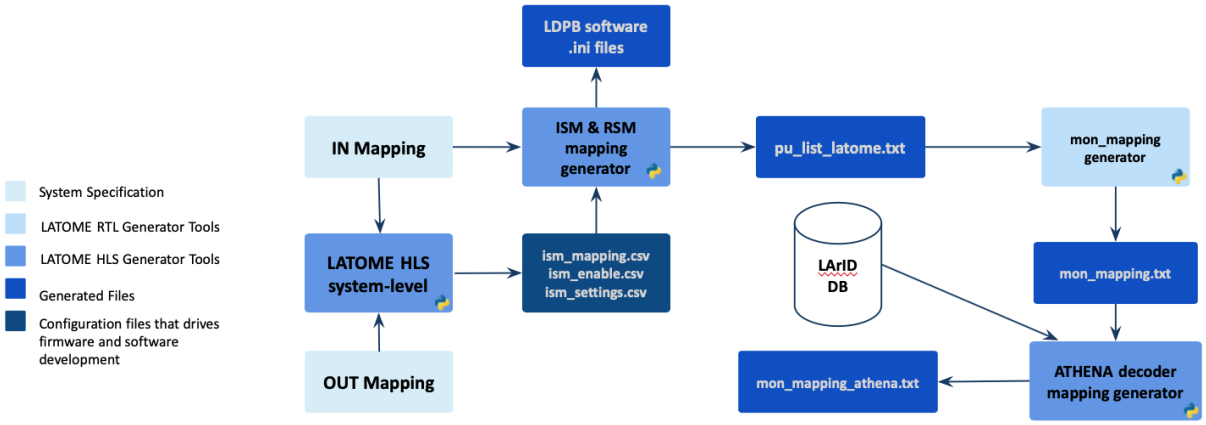

Figure 35 – LATOME HLS software infrastructure

6.3 Firmware Version 6.2

With the monitoring path fully validated in firmware versions 6.0.0 and 6.1.0, development efforts shifted toward the trigger path, aiming to progressively replace the legacy Output Summing (OSUM) block with a High-Level Synthesis implementation. The architectural scope of this transition is shown in Figure 36, where the previously designated ISM HLS IP is now labeled HLS REMAP, and a new block named HLS OSUM is introduced to handle FEX-oriented outputs. This marks the beginning of a broader effort to implement the LATOME FEX path entirely in HLS.

The primary goal of firmware version 6.2.0 was to implement the eFEX-specific output path of the OSUM block, which operates at full SC granularity. Unlike jFEX and gFEX paths, which aggregate energy deposits over coarser η–ϕ areas to form Trigger Towers, the eFEX system transmits the transverse energy E_T of each SC individually. This design is essential for achieving high-resolution selection of electrons, photons, and tau leptons through topological discrimination and hadronic background rejection. As such, the eFEX path bypasses any summation logic in η–ϕ space, focusing instead on the direct transmission of fine-grained SC transverse energy (E_T) toward the eFEX output fibers.

The HLS OSUM implementation for eFEX in version 6.2.0 therefore consisted of a dedicated data path designed to preserve the original spatial resolution of Super Cells while ensuring correct timing, formatting, and integrity for readout. This version also introduced a new clock domain crossing (from 240 MHz to 280 MHz) as part of the output interface — a detail explored in future sections.

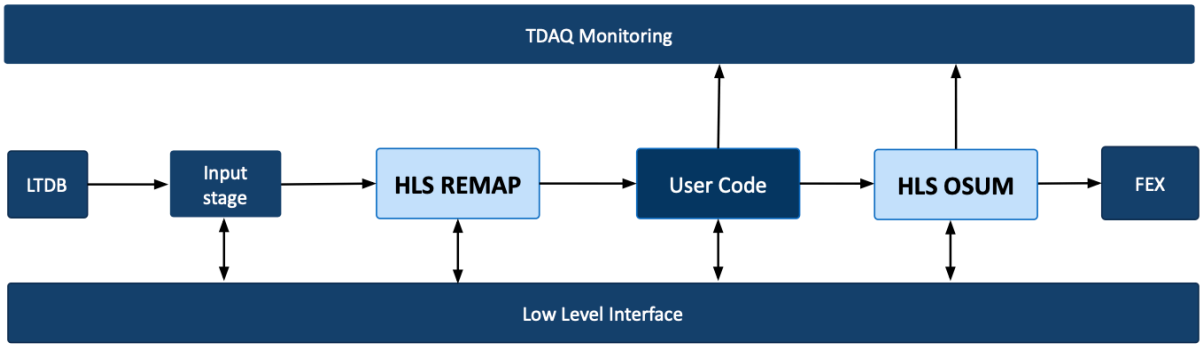

Figure 36 – Simplified trigger path architecture from firmware version 6.2 onward highlighting both HLS designed blocks.

6.3.1 OSUM HLS Architecture

The OSUM HLS architecture designed in firmware version 6.2.0 follows a structured pipeline to process Super Cell data from the User Code and deliver it to the FEXes output fibers with full granularity. This pipeline is illustrated in Figure 37, and comprises several interconnected processing blocks operating across distinct clock domains.

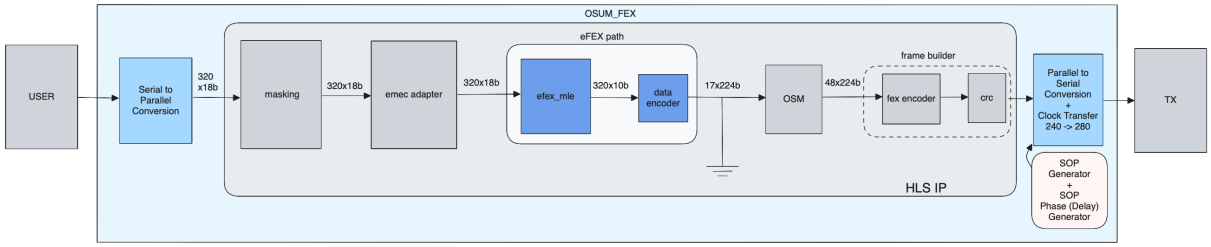

Figure 37 – Block diagram of the OSUM HLS IP showing the eFEX output path. The figure highlights the sequential pipeline stages from the User Code to the final transmission to the FEX system, including key clock domain boundaries and data formatting blocks.

6.3.1.1 Masking

The first block in the OSUM pipeline is the *masking* stage. Its primary function is to allow the selective disabling of individual Super Cell inputs based on a per-channel enable flag. This feature is particularly useful during calibration runs, error isolation procedures, or debugging campaigns where certain channels need to be suppressed without altering the firmware structure.

This block receives the Super Cell energy value along with its associated metadata and a control bit. Specifically, it takes as input the 18-bit signed energy value (`x`), a validity flag (`vld_i`), a saturation flag (`sat_i`), and an enable bit (`ena`) that determines whether the data should be forwarded or masked.

When `ena` is high, the input value and its validity flag are passed through unchanged. When `ena` is low, the output value is replaced with a predefined constant `mask_value`, and the validity flag is forcibly set to high (`vld_o` = 1) to ensure downstream stages receive a consistent data frame structure. The saturation flag (`sat_o`) is forwarded transparently.

This masking logic is applied independently to all 320 channels in parallel at the beginning of the OSUM HLS architecture. It operates within the same 240 MHz clock domain as the incoming User Code data and introduces no latency, preserving deterministic timing for downstream encoding stages.

6.3.1.2 EMEC Adapter

The *EMEC Adapter* block performs dedicated preprocessing for Super Cell data originating from the Endcap Electromagnetic Calorimeter (EMEC) region. Due to specific geometrical and readout characteristics in this region, each adapter processes six input channels corresponding to spatially related Super Cells.

Internally, the adapter implements a weighted recombination of inputs to produce four effective energy values. The calculation splits one of the shared inputs across two outputs to preserve spatial consistency, while the remaining two outputs are zeroed. This transformation can be expressed as:

$$
\begin{aligned}
y_0 &= x_0 + \frac{x_1}{2}, \\
y_1 &= x_2 + \left(x_1 - \frac{x_1}{2}\right), \\
y_2 &= x_3 + \frac{x_4}{2}, \\
y_3 &= x_5 + \left(x_4 - \frac{x_4}{2}\right), \\
y_4 &= 0, \quad y_5 = 0.
\end{aligned}
$$

Validity and saturation flags are propagated using logical combinations of the original inputs, ensuring reliable status tracking through the processing pipeline. All computations employ 18-bit signed saturated arithmetic to guarantee reliable operation without overflow.

A total of 16 EMEC adapter instances are instantiated in parallel in the firmware — 8 for the EMEC-A side and 8 for the EMEC-C side — ensuring full coverage of the EMEC region and standardized formatting before the encoded data enters the next pipeline stage.

6.3.1.3 eFEX MLE: Multi-Linear Encoder

The `efex_mle` block is responsible for converting 18-bit signed transverse energy (E_T) values, provided in ADC counts, into 10-bit encoded values compatible with the eFEX transmission format. This transformation is crucial for reducing the data bandwidth

while preserving precision over a large dynamic range. The encoded values are later packed into the eFEX frame and transmitted to the trigger system.

In this context, **ADC (Analog-to-Digital Converter)** units represent digitized values of the analog energy deposited in the calorimeter Super Cells. While the LATOME firmware processes these values entirely in integer ADC counts, ATLAS documentation and downstream physics analysis workflows conventionally express energy in physical units of MeV. This convention is adopted because MeV is more interpretable in the context of high-energy physics, allowing physicists to relate trigger decisions and reconstructed energy values to known physical processes. To bridge these representations, a fixed conversion factor is used:

$$1 \text{ ADC count} = 12.5 \text{ MeV} \tag{6.1}$$

This implies, for example, that an internal threshold of `et = -60` corresponds to -750 MeV, and an upper bound of `et = 11584` represents 144 800 MeV, matching precisely the endpoints of the eFEX encoding table described in the LATOME firmware documentation.

The efex_mle block uses this correspondence to assign a specific code to each energy range. The conversion is implemented through a piecewise-linear encoding scheme divided into four dynamic linear regions:

- Region 1: from -750 MeV to 5600 MeV, encoded in 25 MeV steps,
- Region 2: from 5600 MeV to 18 400 MeV, encoded in 50 MeV steps,
- Region 3: from 18 400 MeV to 44 000 MeV, encoded in 100 MeV steps,
- Region 4: from 44 000 MeV to 144 800 MeV, encoded in 400 MeV steps.

In addition to these dynamic ranges, the encoding logic reserves specific values to represent special conditions:

- Code `1023` (0x3FF): indicates saturation during energy calculation,
- Code `1022` (0x3FE): indicates invalid or missing input data,
- Code `0` (0x000): used when encoding is disabled.

This encoding allows for a compressed yet accurate representation of calorimeter E_T, with a resolution matched to physics needs across energy scales. The implementation ensures that all outputs are compliant with the expected interpretation downstream in the eFEX processing chain.

6.3.1.4 Data Encoder: Construction of the eFEX Data Frame

After the MLE encoding step, the 10-bit energy values corresponding to 20 Super Cells must be organized into a structured 224-bit frame, compatible with the eFEX format. This function is performed by the data encoder block.

This block packs the 20 codes into a fixed structure that includes:

- **20 10-bit slots** (`DATA0` to `DATA19`): These contain the MLE-encoded Super Cell E_T values,
- **BCID (Bunch Crossing ID)**: The 7 least significant bits of the BCID are spread across three locations in the frame (`BCID[6:5]`, `BCID[4:3]`, and `BCID[2:0]`),
- **K28.5 field**: A fixed comma symbol used by the GBTx receiver for frame boundary synchronization,
- **CRC placeholder**: Although a 9-bit slot for the CRC is defined in the last word of the frame, the CRC is not yet calculated at this stage. The actual checksum is computed only after the OSM step.

W #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	BCID[4:3]		DATA1[9:0]										DATA0[9:0]										BCID[6:5]		K28.5							
1	DATA19[9:8]		DATA4[9:0]										DATA3[9:0]										DATA2[9:0]									
2	DATA19[7:6]		DATA7[9:0]										DATA6[9:0]										DATA5[9:0]									
3	DATA19[5:4]		DATA10[9:0]										DATA9[9:0]										DATA8[9:0]									
4	DATA19[3:2]		DATA13[9:0]										DATA12[9:0]										DATA11[9:0]									
5	DATA19[1:0]		DATA16[9:0]										DATA15[9:0]										DATA14[9:0]									
6	CRC[8:0]									BCID[2:0]			DATA18[9:0]										DATA17[9:0]									

Figure 38 – Bit layout of the eFEX data frame constructed by `efex_data`. The CRC field is not yet valid at this stage.

The frame is divided into seven 32-bit words, totaling 224 bits. All the seven words are used to encode the 20 data fields and embed selected BCID bits for event alignment and integrity. The final word includes a reserved space for the CRC checksum, which remains zero until calculated by the `crc9` block later in the pipeline.

This encoding step concludes the preparation of eFEX data frames. At this point, each of the 16 encoders has generated a valid 224-bit word containing physics information, ready to be multiplexed and routed by the Output Switch Matrix (OSM).

6.3.1.5 Output Switch Matrix (OSM)

After the data encoder builds 16 eFEX frames in parallel, these must be routed to up to 48 physical output fibers in the LATOME architecture. This task is handled by the `osm` block, which implements the Output Switch Matrix (OSM) using a bank of 48 configurable 4-to-1 multiplexers.

Each multiplexer selects one of four input candidates—statically defined by a configuration matrix and drawn from the 16 encoded frames—based on a 2-bit runtime control field. This flexible routing mechanism enables dynamic remapping of logical inputs to physical outputs without requiring re-synthesis of the firmware.

The interface of the OSM is summarized below:

- `x[16]`: Sixteen parallel 224-bit input frames from the data encoder,
- `s[48]`: Forty-eight 2-bit selector fields, one per output,
- `y[48]`: Forty-eight output frames routed to the LATOME frame builder.

Each output fiber of the OSM is statically mapped to a trigger path destination corresponding to one of the three L1Calo feature extractors: eFEX, jFEX, or gFEX. This association depends on the LATOME's position in the detector and remains fixed for a given deployment. For example, a configuration for the EMBA_1 assigns the first 36 outputs to eFEX, the next 10 to jFEX, and the final 2 to gFEX.

The 224-bit frame width is preserved throughout the switching operation, ensuring that frame contents remain unaltered. Additionally, a 17th grounded input stream is instantiated at the top level.

The OSM thus provides a reconfigurable and scalable bridge between the data encoding and frame construction stages, supporting multiple LATOME deployment scenarios with region-specific mappings.

6.3.1.6 Frame Select Block

After the 48 output streams are routed by the Output Switch Matrix (OSM), each must be finalized into either a standard data frame or an alignment frame, depending on synchronization requirements. This decision is handled by the *Frame Select* block.

The primary role of this block is to examine the current Bunch Crossing ID (BCID) and compare it to a predefined reference value. If the current BCID matches this synchronization value—typically set to `3500` in the LATOME firmware—the block overrides the incoming data and constructs a special *alignment frame*. Otherwise, it passes the data frame through unchanged.

The alignment frame serves as a synchronization marker across the system. Instead of Super Cell data, it carries metadata fields that identify the LATOME's position and output stream configuration. These include:

- A fixed comma character for framing (`K28.5`),
- A control character (`K28.0`),

- The fiber ID, which uniquely identifies the physical output link,
- The FEX ID, indicating the target processing block (eFEX, jFEX, or gFEX),
- The LATOME ID, representing the board instance within the system,
- The LATOME source ID, a 32-bit identifier encoding higher-level routing metadata,
- The 12-bit Bunch Crossing ID (BCID), and
- Reserved fields to ensure fixed-length formatting and alignment.

Depending on the FEX type, the frame format slightly varies: for eFEX outputs (`fex_id = 0`), the control metadata follows a specific eFEX convention, while jFEX and gFEX outputs use a different encoding, as defined by their control field assignments.

When the frame is not substituted (i.e., for typical data transmission), the original 224-bit data frame is passed unchanged to the next stage. The control metadata is updated to reflect the nature of the frame — either `data` or `align`, and the corresponding FEX type — ensuring downstream modules can decode the information appropriately.

The alignment frame structure is illustrated in Figure 39, showing the placement of each metadata field. This frame type is inserted periodically based on the BCID and plays a key role in downstream synchronization and diagnostic procedures.

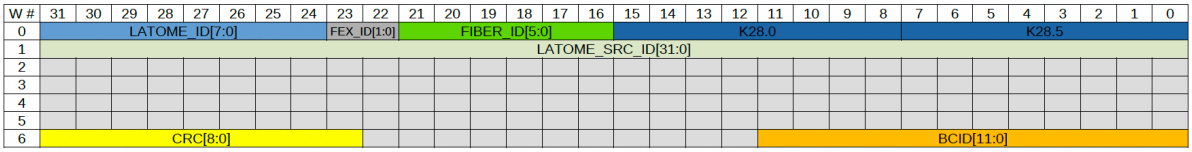

Figure 39 – Structure of the eFEX align frame.

6.3.1.7 CRC-9 Calculation

Once the data or alignment frame has been selected, a 9-bit Cyclic Redundancy Check (CRC) code is computed and appended to the frame to ensure data integrity. This operation is performed by the `crc9` block, which implements the encoding algorithm defined by the LATOME specification.

The CRC is calculated over the first six 32-bit words and the lower 23 bits of the seventh word of the frame. These constitute the first 215 bits of the 224-bit data frame. Before computation, the serialized bitstream is formed by concatenating the words in little-endian order, starting from the least significant bit (LSB) of the first word to bit 23 of the last word.

The polynomial used for CRC generation is:

$$P(x) = x^9 + x^7 + x^6 + x^5 + x^4 + x^3 + x^1 + 1$$

Special handling is applied to the synchronization characters:

- Any occurrence of the `K28.5` comma character in the frame is replaced with the byte `0x00` before CRC computation.
- The `K28.0` character, if present, is included in the CRC calculation without modification.

The resulting 9-bit checksum is inserted into the upper bits of the final word:

- Bit 23 receives the most significant bit (MSB) of the CRC,
- Bit 31 receives the least significant bit (LSB).

This reverse ordering ensures compatibility with the deserialization logic on the receiver side. When the CRC-9 is correctly calculated and embedded, a subsequent computation of the CRC over the full 224-bit frame at the receiver will yield a zero result, confirming error-free transmission.

Once the CRC-9 is computed, it must be appended to the outgoing frame. This is accomplished by a final step in the pipeline, where the 9-bit checksum is inserted into bits 215 to 223 of the 224-bit frame as shown in Figure 38 and Figure 39. This operation is purely structural: it concatenates the existing 215-bit payload with the calculated CRC, producing the final word-aligned output to be serialized and transmitted over the optical fibers.

6.3.2 Clock Transfer and SerDes

The Output Summing (OSUM) HLS IP communicates with the surrounding LATOME firmware through parallel inputs. However, the data from the User Code arrives in serial streams, while the HLS IP blocks expects fully parallel inputs. To bridge these two interfaces, dedicated Serial-to-Parallel (S2P) and Parallel-to-Serial (P2S) adapters are instantiated at the input and output boundaries of the OSUM logic. These adapters also manage the necessary clock domain crossing, ensuring a seamless transfer of data between the 240 MHz and 280 MHz domains.

6.3.2.1 Serial-to-Parallel Conversion

Before data enters the OSUM HLS IP arithmetic blocks, each input stream undergoes de-serialization via a series to parallel converter. This logic transforms serial 20-bit words into parallel blocks over six clock cycles, matching the 320 outputs of the ISM.

The deserialization structure consists of:

- A top-level wrapper that instantiates 54 independent deserializers in parallel.
- Each deserializer accumulates six consecutive serial samples from its stream into a cascaded shift register structure, which converts the serial input into six parallel words.
- Upon reception of the start-of-packet flag, the contents of the shift register are captured into shadow registers, ensuring temporal alignment across streams.
- The outputs are forwarded as a flattened array of 320 parallel values.

All logic runs synchronously with the 240 MHz clock. The `sop_out` signal marks the beginning of each new frame, ensuring that the summing logic receives properly aligned groups of data. This interface preserves both temporal coherence and synchronization across the 54 input channels to the OSUM.

6.3.2.2 Parallel-to-Serial Conversion and Clock Domain Crossing

After the processing steps inside the OSUM HLS IP, the output frames are still organized as 224-bit parallel words, grouped in 48 independent output channels. However, to comply with the FEXes output requirements, these frames must be serialized into 32-bit words and transferred to the 280 MHz domain.

This transition is managed by a dedicated Parallel-to-Serial (P2S) converter, which performs two key tasks:

- It serializes each 224-bit data frame into seven consecutive 32-bit chunks. This ensures compatibility with downstream transmission hardware.
- The conversion into seven 32-bit words aligns with the 280 MHz output domain, which spans seven clock cycles per 40 MHz frame. This transfer is coordinated through a valid-flag driven handover mechanism, with synchronization signals such as `sop_240` and `sop_280` managed by auxiliary modules like the `valid_flag_gen` and `sop_generator`.

The serialization logic converts each 224-bit data frame into seven 32-bit words, delivering one word per clock cycle at 280 MHz. An internal counter coordinates this slicing operation, and a circular valid flag mechanism ensures that each word is emitted in the correct sequence.

This P2S interface guarantees that the 32-bit data and control words are correctly transmitted to the downstream FEX system, maintaining both temporal alignment and the integrity of the frame structure established during the OSUM processing.

6.3.2.3 Mini-FEX Monitoring Logic

To enhance observability and facilitate debugging during simulation campaigns, laboratory validation, and Point-1 system integration, a lightweight diagnostic structure—referred to as the *Mini-FEX*—was instantiated at the top level of the `osum_enc.vhd` wrapper. Although its complete implementation is only present in firmware version 6.4 (which lies outside the formal scope of this thesis), its architecture builds upon mechanisms developed and tested in earlier versions.

The Mini-FEX provides real-time, non-intrusive monitoring of internal signals related to the OSUM-to-LLI transmission chain, with particular emphasis on the 240 MHz to 280 MHz clock domain crossing. Its diagnostic outputs are exposed through IPbus-readable counters and flags.

- **Valid Output Flag:** This counter increments on every transmission of a valid 32-bit data word from the P2S block. Under continuous operation, it should saturate at its maximum value, confirming that the OSUM HLS block is producing output consistently.
- **Start-of-Packet Markers:** Three independent counters track synchronization events: one for SOP pulses generated in the 240 MHz domain, one for the resynchronized SOP used in the 280 MHz domain, and one for a non-delayed SOP path. These channels help diagnose potential synchronization issues introduced during clock crossing.
- **SOP Period Errors:** To ensure that SOP pulses maintain a fixed cadence (e.g., every 6 or 7 clock cycles depending on domain), the Mini-FEX accumulates error flags triggered when a deviation from the expected SOP periodicity is detected. All such counters should remain zero during stable operation.
- **CRC Errors:** For each of the 48 output streams, the integrity of the 224-bit OSUM output frame is verified using a 9-bit CRC. If the calculated CRC differs from the expected value, a corresponding error counter is incremented. Nominally, these counters should remain at zero across all channels.
- **Control Word Activity:** A set of counters also tracks the emission of control words used by the serializer. These counters reflect the presence of control markers (e.g., packet delimiters or padding), and their saturation can be indicative of consistent framing logic.

By consolidating these observability features in a centralized structure, the Mini-FEX offers a robust toolset for firmware validation. It enables rapid identification of subtle

issues—such as SOP misalignment, CRC mismatches, or control flow irregularities—across both simulation and hardware contexts.

6.3.3 Simulation Strategy and Models

To validate the functional behavior of the firmware logic for remapping and output summing, two complementary models were introduced starting with firmware version 6.2: the *Firmware Agnostic Model* and the *Firmware Aware Model*. These models implement independent verification strategies and are integral to a layered simulation structure designed to support systematic validation of the LATOME HLS Device Under Test (DUT).

6.3.3.1 Firmware Agnostic and Firmware Aware Models

The Firmware Agnostic Model simulates expected behavior using only the input and output mapping specifications. It does not rely on knowledge of the internal firmware structure. This makes it a powerful tool for verifying the logical correctness of remapping and summing operations against known configurations.

In contrast, the Firmware Aware Model incorporates detailed knowledge of the firmware, using as inputs the exact configuration files used by the ISM and OSUM blocks. This model mimics the internal structure of the firmware and supports simulation of individual blocks with high fidelity. It allows inspection of intermediate outputs within the firmware path—capabilities not available in the agnostic model.

Both models are mutually exclusive and are cross-validated using the same seeded random input to generate comparable input data as shown in Figure 40. This design ensures consistency and enables cross-validation between models, minimizing the risk of propagating systematic errors.

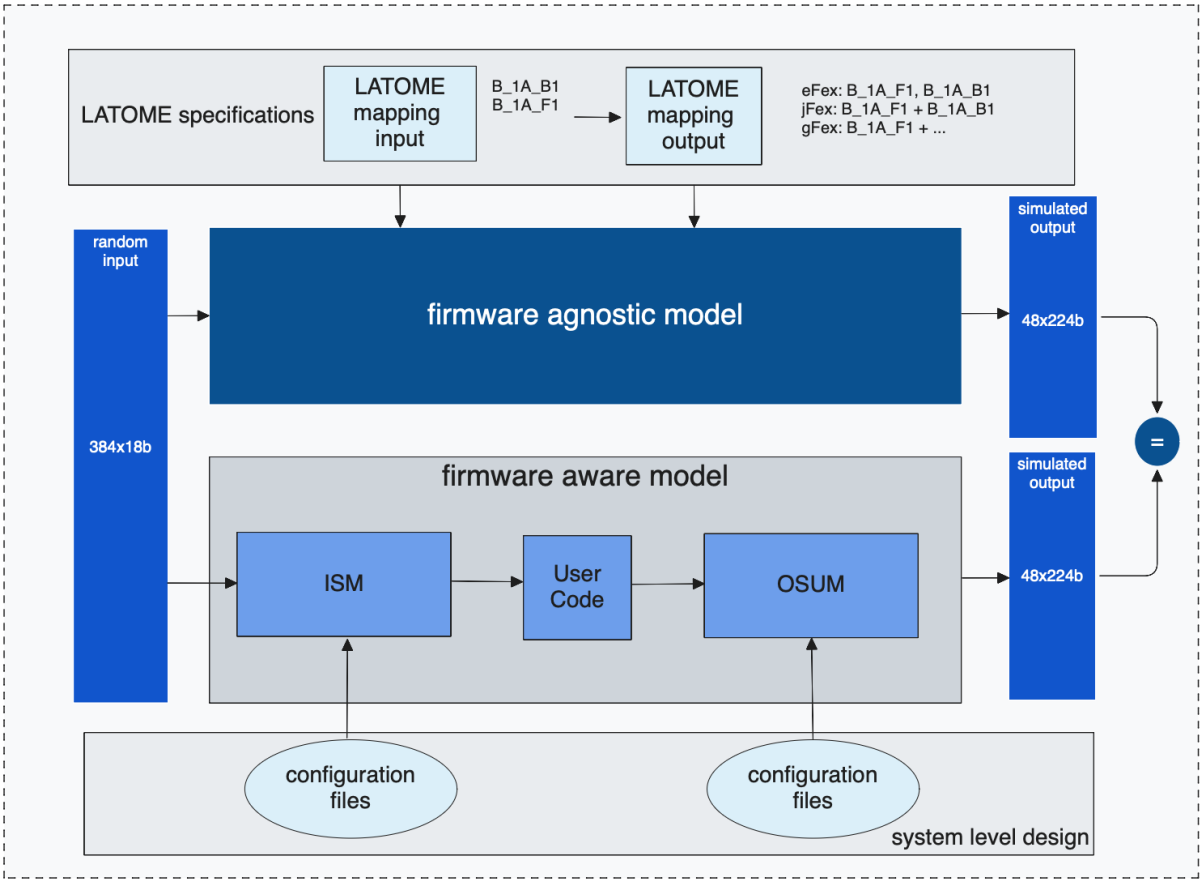

Figure 40 – Firmware Aware and Agnostic Models

6.3.3.2 Layered Simulation Strategy

To progressively test and integrate simulation complexity, the validation framework is structured into four layers:

6.3.3.2.1 Layer 0

This layer simulates each one of the new blocks, individually, using a C++ testbench. This stage is divided in two types of simulations:

- *C++ simulation:* Executed in software and compiled with gcc, it compares the outputs of the HLS synthesizable code against the output from the testbench code (non-synthesizable).
- *RTL co-simulation:* Performed using Siemens Questa Advanced Simulator and the SCVerify flow. The C++ testbench interfaces with VHDL code generated by Catapult via an automatically generated SystemC adaptor. This ensures that both simulation modes use the same testbench and allows detailed register-transfer-level

(RTL) validation. However, the C++/RTL co-simulations are significantly slower due to the generated circuit at the register transfer level.

6.3.3.2.2 Layer 1

The goal of this simulation layer is to validate the functionality of the High-Level Synthesis (HLS) blocks synthesized from C++ code, focusing on the Input Switch Matrix (ISM) and Output Summing (OSUM) modules. These two blocks, implemented in C++ and translated to RTL using the Catapult HLS tool, are tested with parallel interfaces operating in a single 240 MHz clock domain. The aim is to identify and resolve any functional issues in the HLS logic before introducing additional complexities such as clock domain crossings and serial to parallel and parallel to serial conversions. The Device Under Test (DUT) used in this layer is shown in Figure 41, has the ISM on the left connected directly to the OSUM with no User Code Bypass or Serializers in between. This initial verification phase ensures the correctness of internal data paths and control logic, providing a stable foundation for more integrated simulations in subsequent layers.

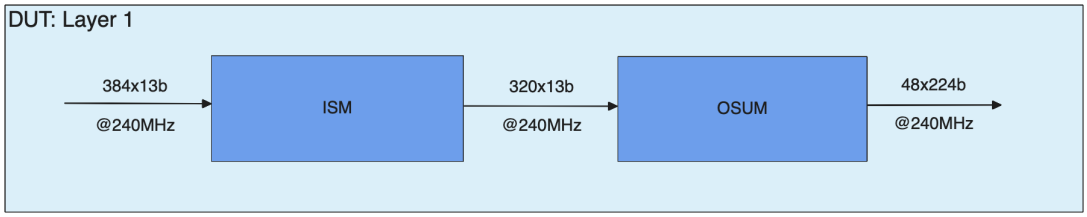

Figure 41 – Simulation strategy for Layer 1: isolated HLS block verification in a single clock domain.

6.3.3.2.3 Layer 2

Layer 2 builds upon the foundation established in Layer 1 by introducing serialization and clock domain crossing, which are required to interface the newly developed HLS blocks with existing firmware structures. As illustrated in Figure 42, the DUT for this layer includes serializers and deserializers surrounding the ISM and OSUM blocks. These additional components enable time-division multiplexing by converting data from a 320 MHz serial format to a 240 MHz parallel format before the ISM, and from 240 MHz parallel back to 280 MHz serial after the OSUM.

The primary objective of this simulation layer is to validate the behavior of the SerDes blocks and their interaction with the HLS-generated modules, ensuring proper clock synchronization and data integrity across three different clock domains. This architecture allows integration with legacy components while maintaining the performance of the new firmware. To ensure consistency, this layer also verifies that serialization and deserialization

do not introduce any data corruption or timing issues as signals traverse the firmware pipeline.

In addition, Layer 2 incorporates a *user code bypass* block. This module mimics the user code behavior when it applies unity FIR coefficients, effectively routing input signals directly to their corresponding outputs. This allows thorough testing of the entire data path while isolating the influence of the user code logic itself.

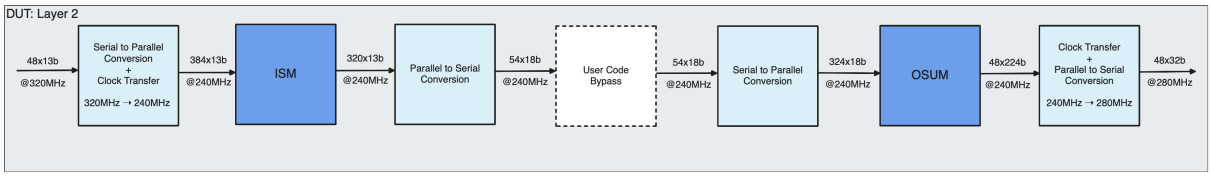

Figure 42 – Simulation strategy for Layer 2: verification of blocks across clock domain boundaries using SerDes.

6.3.3.2.4 Layer 3

Integrates the full LATOME top-level firmware, including User Code and Python simulations of the low-level software stack. This layer exercises the entire chain from serialized input to final output and is used to validate the connectivity, configuration, and synchronization of the real-time trigger path. Figure 43 illustrates the Layer 3 setup.

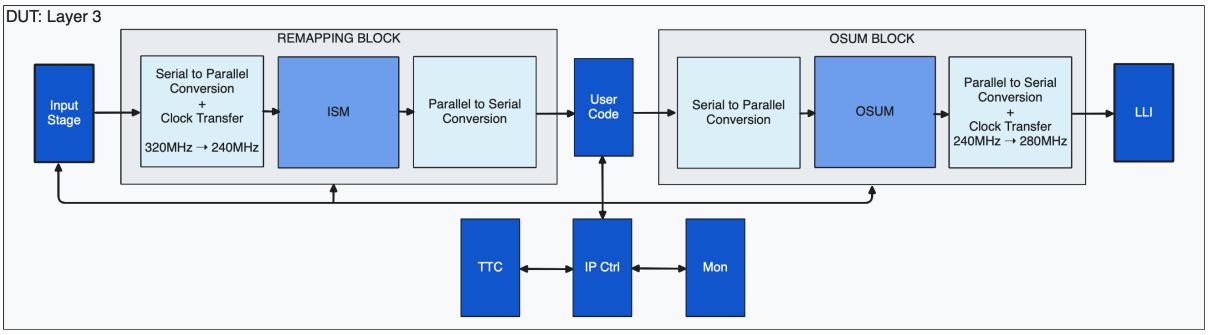

Figure 43 – Simulation strategy for Layer 3: full-firmware validation including User Code and output logic.

This simulation methodology has been used to validate all 116 mapping configurations from firmware versions v6.2.0 and onward, ensuring alignment between the firmware outputs and the expected outputs from both models.

6.3.4 Validation Results

The validation campaign for firmware version 6.2 focused on the finalized integration of the `remap` and `osum` blocks regarding the eFEX path, following the architectural

simplifications introduced after the removal of the `rsm` logic. This version successfully passed all stages of the layered simulation strategy. It began with Layer 0, where each block was tested in isolation through stand-alone simulations using C++. Layer 1 introduced the functional simulation of the generated RTL in a single clock domain, validating the behavior of the blocks with parallel interfaces. In Layer 2, the serializers, deserializers, and clock domain transfer logic were introduced, enabling the verification of data transfers between the 320 MHz, 240 MHz, and 280 MHz domains. Finally, Layer 3 integrated all components into the full LATOME firmware and confirmed the system's functional behavior using the top-level user code and the Python-based low-level software.

6.3.4.1 REMAP Validation

After the successful completion of the simulation campaign, it was time to proceed with hardware validation. The main objective of this phase was to verify the firmware's clock domain transfers, particularly within the REMAP block, as done for firmware versions 6.0.0 and 6.1.0, and now extended to include the clock domain crossing present in the OSUM block. The first step was to determine the optimal `sop_delay_select` value for `remap`. To achieve this, the same `mon_tdaq` testing infrastructure used in earlier versions was employed in the laboratory.

The procedure included a stress test on the `EMBA_1` mapping, in which each of the six possible delay configurations was exercised over 50 consecutive configuration cycles, running continuously for 10 hours. In addition, a broader test campaign was carried out for all representative mapping families, with 10 configuration cycles per delay value. Both tests revealed the same operating margin, and the full testing campaign lasted approximately 24 hours.

Tables 6 and 7 summarize the observed success rates for each `sop_delay_select` configuration. The best results were consistently obtained for delay values 1, 2, and 5, which achieved full or near-full success across all regions. Delay 2 emerged as the most robust and stable choice, achieving 100% success in both the stress test and the full mapping sweep.

It is important to note that, version 6.2 was compiled with minor timing violations due to constraints during implementation. These violations did not prevent functional operation but are likely the cause of isolated failures seen in otherwise valid configurations, such as delays 0, 3, and marginally 5. Delay 4, in particular, showed complete failure across all mappings and was therefore discarded as a viable configuration.

Despite the slightly lower success rates in some configurations, the overall outcome confirms the firmware's stability in real-world conditions. The test results validate the correct behavior of the clock domain crossing logic in the `remap` block under diverse mapping scenarios and long-duration operation.

sop_delay_select	EMBA_1 (%)
0	98.00
1	98.00
2	100.00
3	96.00
4	0.00
5	98.00

Table 6 – Percentage of success on complete tests for firmware version v6.2.0 using the EMBA_1 mapping (50 completed tests).

sop_delay_select	EMBA_1	EMBC_1	EMBA_EMECA_1	EMBC_EMECC_1	EMECA_1	EMECC_1
0	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	90.00%	100.00%	100.00%	90.00%
4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
5	100.00%	100.00%	100.00%	100.00%	90.00%	100.00%
sop_delay_select	EMECA_HECA_1	EMECC_HECC_1	FCAL1A	FCAL1C	FCAL2A	FCAL2C
0	100.00%	90.00%	100.00%	90.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
5	100.00%	100.00%	100.00%	100.00%	90.00%	100.00%

Table 7 – Percentage of success on complete tests for firmware version v6.2.0 across LATOME mappings.

The validation of the `osum` block, which also introduced a new clock domain boundary, required a different strategy, as the infrastructure for SOP delay stress testing was not yet available for the OSUM block at the time of testing.

6.3.4.2 OSUM Validation

Because the required infrastructure was not yet available, the `osum` block could not be validated via laboratory SOP-delay stress testing, unlike `remap`. While `remap` already had a working monitoring path that enabled SOP-delay scans in the `MON/TDAQ` environment, validating `osum` required the full trigger readout path, which had not been commissioned during the `osum` validation phase. The adopted strategy therefore combined (i) laboratory diagnostics using the Mini-FEX module and (ii) operational monitoring at Point 1 (P1), the ATLAS detector site and control room.

The Mini-FEX module, integrated into the `osum` top-level wrapper, exposes internal counters and error flags usable in simulation and hardware. In laboratory tests it confirmed that the expected counters—tracking SOP-aligned streams in the input (240 MHz) and output (280 MHz) domains—incremented consistently. At the same time, the critical error

indicators, including `sop_period_errors` and the CRC error counters, remained at zero, indicating stable behavior across the `osum` clock domains.

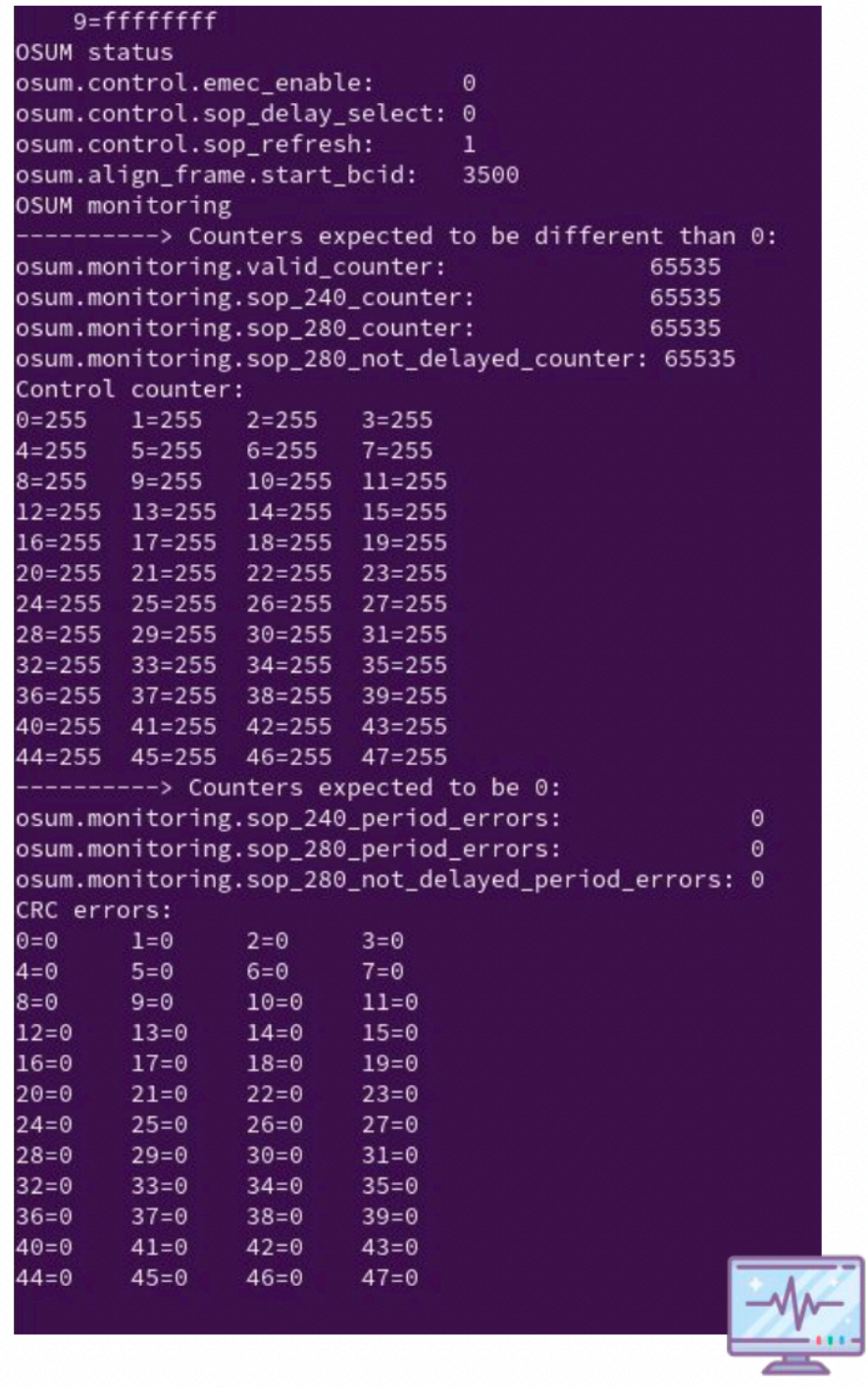

Figure 44 – Mini-FEX diagnostic output during laboratory testing. All counters incremented as expected and no SOP-related or CRC errors were observed.

To assess behavior under realistic conditions, Point-1 tests were performed and the results from Grafana dashboards were used to correlate the configured SOP-delay values with CRC error rates. Figure 45 shows that, once the delay values were correctly set, CRC

errors dropped markedly, consistent with correct alignment and timing across all active links.

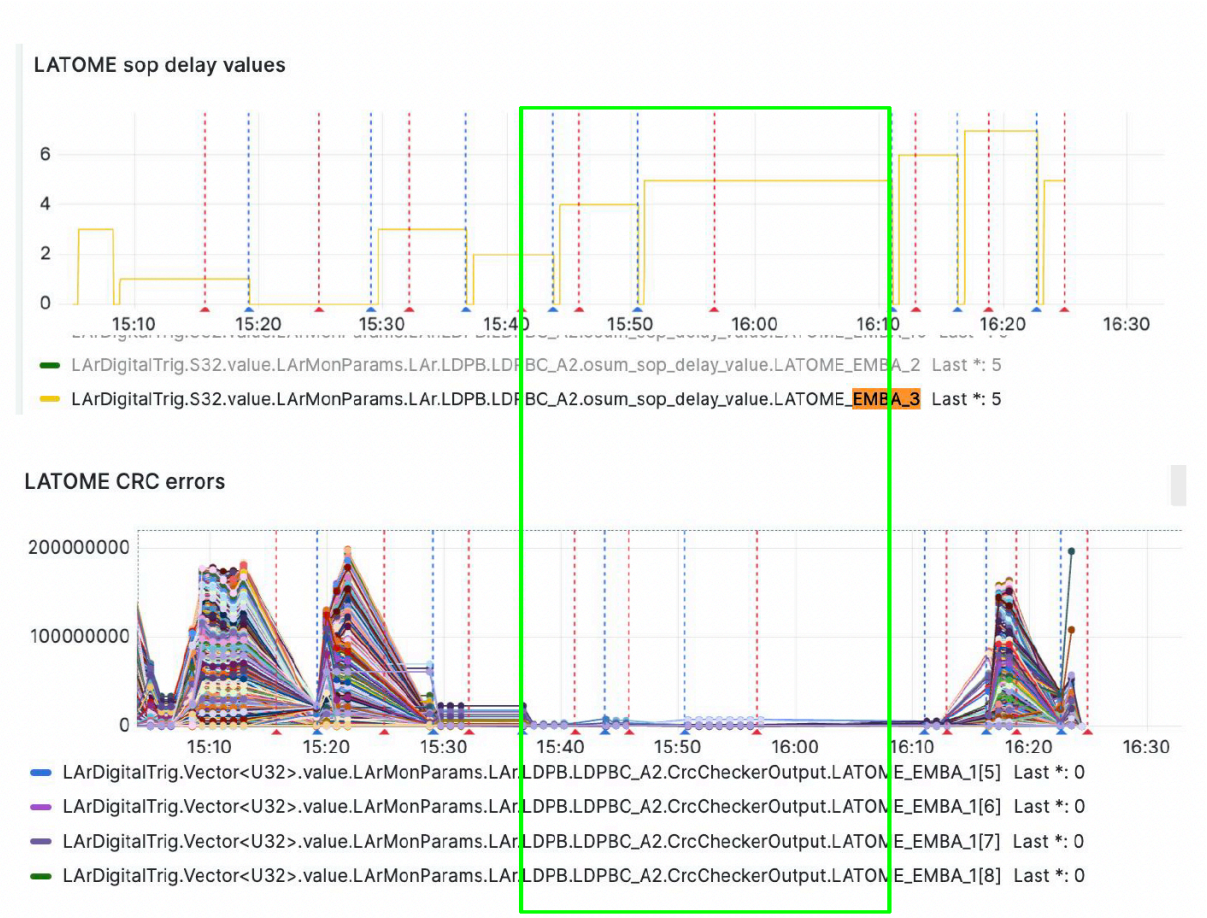

Figure 45 – Grafana dashboard from a P1 run showing OSUM SOP-delay values (top) and corresponding CRC error counters (bottom). The highlighted period shows stable operation after configuration.

Finally, a targeted SOP-delay scan for `osum` was carried out in the P1 environment with the `remap` delay fixed at the optimal value of 2. Table 8 summarizes the results for `osum` delay settings 0–7: CRC errors appeared for 0–2 and 7, while 3–6 produced stable, error-free operation. The nominal configuration for firmware version 6.2 was therefore set to `osum_sop_delay_select` = 4, the center of the stable window.

This behavior is consistent with the clocking relationship: the `osum` output domain runs at 280 MHz and is sampled relative to the 40 MHz system clock. Since $280 = 7 \times 40$, there are seven unique phase positions modulo the 40 MHz period; scanning the eight discrete settings 0–7 covers a full cycle, with settings 0 and 7 effectively equivalent.

REMAP	OSUM	Result
2	0	CRC ERROR
2	1	CRC ERROR
2	2	CRC ERROR
2	3	PASSED
2	4	PASSED
2	5	PASSED
2	6	PASSED
2	7	CRC ERROR

Table 8 – SOP-delay scan in the Point 1 system with `remap` fixed at 2 and `osum` swept from 0 to 7.

Despite minor timing violations in the final 6.2 implementation, the full system was deployed and exercised at P1 with stable operation on all critical paths, including the `osum` clock-domain crossing. These results validated the 6.2 architectural simplifications and prepared the ground for version 6.3, which introduced the `jfex` processing path alongside a comprehensive clock-tree study to improve timing closure and future scalability.

6.4 Firmware Version 6.3

Firmware version 6.3 introduced the jFEX (Jet Feature Extractor) processing path, extending the LATOME architecture to support multiple FEX destinations in parallel. Designed for reconstructing transverse energy quantities over broader spatial windows—such as jets and global sums—this path processes Super Cell inputs through two dedicated branches: an adder path that accumulates energy deposits across regions of interest, and a single path that forwards individual Super Cell data without summation. These dual processing ramifications prepare the data for processing algorithms implemented downstream in the jFEX system.

The figure below illustrates this architectural addition. The 320 parallel Super Cell inputs, each 18 bits wide, are routed by two dedicated switch matrices—the jASM (Adder Switch Matrix) and the jSSM (Single Switch Matrix)—which drive separate downstream pipelines for summed and raw data.

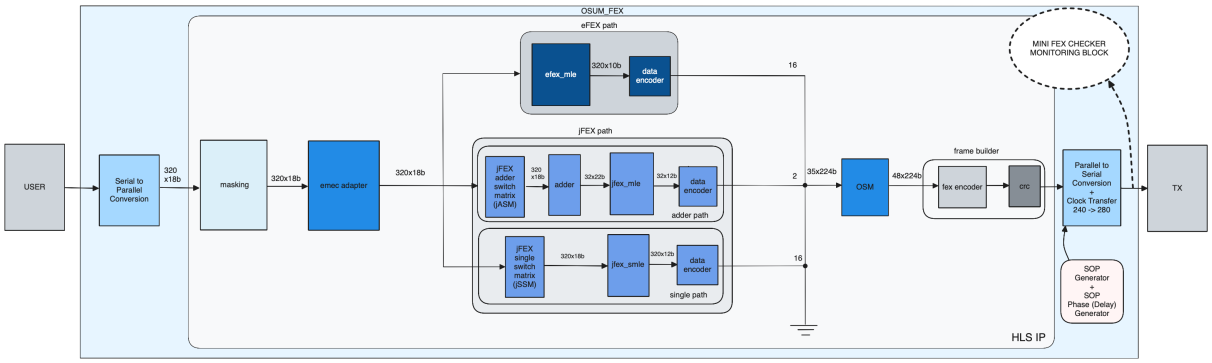

Figure 46 – OSUM block diagram showing the integrated jFEX path.

The existing logic for REMAP and OSUM remained unchanged from version 6.2, allowing previously validated configurations to be preserved without introducing architectural risk.

In addition to this functional enhancement, version 6.3 was the first LATOME firmware successfully compiled without any timing violations. This result followed a comprehensive clock tree and constraint analysis by the firmware development team. The improvements in timing closure increased overall robustness and paved the way for version 6.4, which will introduce the gFEX path and finalize the routing infrastructure to all target destinations.

6.4.1 Adder Path

The adder path is responsible for computing regional energy sums from incoming Super Cell data, as required by the jFEX algorithm. It comprises four processing blocks: the jFEX Adder Switch Matrix, which performs region-dependent stream routing; the Adders, which accumulate the selected inputs; the Multi Linear Encoder, which encodes each adder output into a compact representation based on its energy value and associated control flags, following the jFEX encoding specification; and the Data Encoder, which formats and aligns the resulting data for integration into the OSM. The following subsections describe the structure, functionality, and signal interface of each of these components.

6.4.1.1 jASM (jFEX Adder Switch Matrix)

The adder path begins with the jASM, a two-stage multiplexer circuit responsible for routing the 320 Super Cell inputs to downstream processing units. Its internal logic follows a mechanism analogous to the ISM block and is driven by mapping-specific configuration settings. For each of the 320 output channels, the jASM configuration matrix specifies a statically defined group of 7 candidate inputs, which feed the first stage of a cascaded

two-stage multiplexer structure. This matrix statically defines, for each of the 320 output channels, a set of 7 candidate inputs selected from the global pool of Super Cells.

During firmware initialization, each output is controlled by a selection structure, which includes a 3-bit `sel` field to choose one of the 7 candidates in the first multiplexer and a 1-bit `ena` field that acts as an enable mask in the second cascaded multiplexer. These selection values are loaded from configuration files (e.g., `osum.ini`) and remain fixed during operation. If enabled, the selected candidate's data and flags are forwarded to the output; otherwise, the output is suppressed.

Each jASM output consists of an 18-bit data word and its associated *valid*, *saturation*, and *enable* flags, which are conditionally propagated based on the selected candidate and enable control. These 320 outputs—now dynamically routed and filtered—serve as the direct inputs to the subsequent adder instances, where regional energy accumulation takes place.

6.4.1.2 Adder Blocks

Each of the 320 outputs from the jASM is routed to one of 32 parallel instances of the adder block. Each instance processes a fixed group of 10 input streams, summing their values and evaluating their control flags to produce a single aggregated output.

The adder receives 18-bit signed data words along with three 1-bit flags per stream: *valid*, *saturation*, and *enable*. Internally, each instance computes the arithmetic sum of the input values and performs a bitwise OR across all flags to indicate whether any of the inputs were enabled, valid, or saturated. The result is a 22-bit signed output accompanied by three propagated control flags.

All operations are implemented using fully unrolled combinational logic, allowing the outputs to be computed in a single clock cycle. This modular and parallel structure ensures that regional energy contributions from multiple Super Cells are efficiently combined and forwarded to the MLE block for encoding.

6.4.1.3 jFEX MLE: Multi-Linear Encoder

The `jfex_mle` block is responsible for converting 22-bit signed transverse energy (E_T) values, produced by the adder blocks, into 12-bit encoded values compatible with the jFEX transmission format. This transformation is essential for reducing bandwidth while preserving precision across a wide dynamic range. The encoded values are subsequently formatted by the Output Switch Matrix (OSM) for transmission to the trigger system.

As in the eFEX path, energy values are represented in ADC counts and may be translated into physical units of MeV using the fixed conversion factor:

$$1 \text{ ADC count} = 12.5 \,\text{MeV} \tag{6.2}$$

For instance, an input value of `et = -252` corresponds to −3150 MeV, while `et = 64000` maps to 800 000 MeV, matching the endpoints of the jFEX encoding table.

The `jfex_mle` block implements a five-region piecewise-linear encoding scheme, with increasing step sizes to accommodate the broad energy spectrum:

- Region 1: from −3150 MeV to 6400 MeV, encoded in 25 MeV steps,
- Region 2: from 6400 MeV to 20 480 MeV, encoded in 50 MeV steps,
- Region 3: from 20 480 MeV to 102 400 MeV, encoded in 100 MeV steps,
- Region 4: from 102 400 MeV to 409 600 MeV, encoded in 200 MeV steps,
- Region 5: from 409 600 MeV to 799 600 MeV, encoded in 400 MeV steps.

In addition to these dynamic encoding regions, the logic reserves specific output codes for special conditions:

- Code `0` (0x000): used when encoding is disabled (`ena = 0`),
- Code `4095` (0xFFF): indicates invalid or missing input data (`vld = 0`),
- Code `1` (0x001): assigned to inputs below the minimum threshold,
- Code `4048` (0xFD0): assigned to inputs exceeding the upper bound,
- Other codes such as 0xFCF–0xFFE: reserved for future use.

This encoding strategy ensures accurate compression of transverse energy values across the full operational range, supporting both low-energy and high-energy phenomena without loss of resolution. Compared to the eFEX MLE, the jFEX encoder supports a wider dynamic range and offers finer granularity in handling special cases.

To support full parallelism, 32 instances of the `jfex_mle` block operate simultaneously—one per adder output—executing the encoding in a single combinational cycle. The logic is fully unrolled and follows precisely the definitions documented in the LATOME specification.

6.4.1.4 Data Encoder: Construction of the jFEX Data Frame

Following the encoding performed by the `jfex_mle` blocks, the 12-bit energy values from 16 Super Cells are organized into a structured 224-bit frame according to the jFEX data format. This task is handled by the jFEX data encoder block, which also embeds additional metadata and synchronization markers.

The jFEX data frame, shown in Figure 47, includes:

- **16 12-bit slots** (`DATA0` to `DATA15`): These fields store the MLE-encoded energy values for the processed Super Cells.
- **Saturation flags** (`SATUR[15:0]`): Two 8-bit fields mark saturated Super Cells. If all of `SATUR[7:0]` are zero, the field is replaced by the `K28.5` comma symbol for synchronization.
- **BCID**: A 7-bit subset of the Bunch Crossing Identifier for temporal alignment.
- **CRC**: A 9-bit checksum field reserved in the final word. The CRC is computed later by the Output Switch Matrix (OSM).

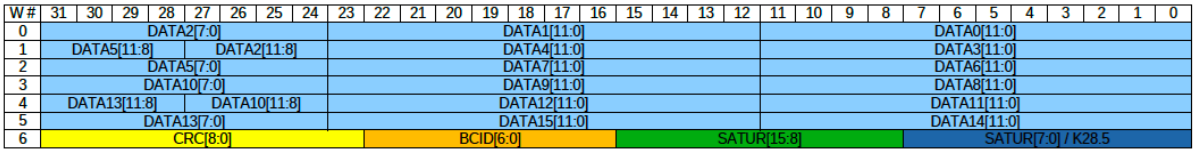

Figure 47 – Bit layout of the jFEX data frame constructed by `jfex_data`. The `K28.5` symbol replaces `SATUR[7:0]` when no saturation is detected.

In parallel, a second frame known as the align frame (Figure 48) is periodically generated to convey metadata and assist receiver synchronization:

- **FEX_ID**: Set to `1` to indicate the jFEX path.
- **FIBER_ID**: A 6-bit identifier for the output fiber (0 to 47).
- **LATOME_ID**: The 8 least significant bits of the LATOME board identifier.
- **LATOME_SRC_ID**: A 32-bit source identifier.
- **BCID**: Full 12-bit Bunch Crossing Identifier.
- **Comma symbols**: `K28.1` and `K28.5` mark the frame and boundaries.
- **CRC**: A 9-bit reserved field for the CRC checksum, computed downstream.

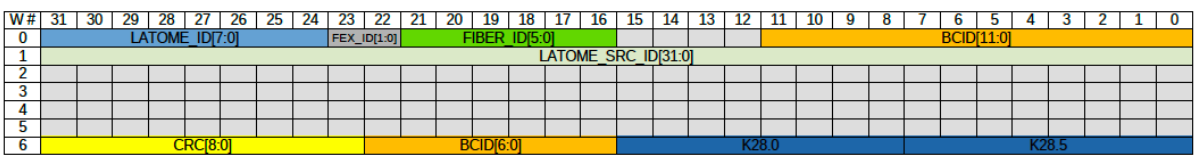

Figure 48 – Structure of the jFEX align frame. Includes FEX ID, fiber ID, full BCID, LATOME identifiers, and frame boundary markers.

These align frames synchronize the receiver to the LATOME data stream, establishing clear word and frame boundaries while also conveying identification and timing information. Together with the data frames, they ensure reliable and correctly interpreted data delivery to downstream hardware.

6.4.2 Single Path

The jFEX Single Path is a parallel processing route within the LATOME firmware that preserves the individual granularity of selected Super Cell signals. Unlike the adder path, which aggregates energy contributions across regions of interest, the single path handles one-to-one mappings from input Super Cells to output channels. The Single Path is composed of three key blocks: the `jSSM`, which handles input selection and routing; the `jfex_smle`, which applies a reduced encoding scheme; and the data encoder block, which formats the output frame according to the jFEX single path specification.

6.4.2.1 jSSM: Single Switch Matrix

The jSSM block implements the Single Switch Matrix used in the single path processing chain for the jFEX output. Its role is to route selected energy values from 320 input streams to 320 output channels based on a predefined configuration. As in the jASM block used in the adder path, each output channel is associated with two candidate inputs, and the actual selection is performed dynamically based on two control fields: a one-bit selector and an enable flag.

The configuration for the two candidate inputs, along with the corresponding selection and enable behavior, is loaded at initialization from the ‘osum.ini‘ configuration file. This mechanism allows flexible routing based on detector region.

The block instantiates 320 independent 2-to-1 multiplexers—one for each output stream. Each multiplexer selects between two predefined inputs and masks the output to zero if the enable signal is deasserted. This ensures that only active and authorized data streams are forwarded for further processing.

All relevant control flags—valid, saturation, and enable—are propagated from the selected input to the output, preserving data integrity. The entire logic operates in a single combinational cycle and is fully unrolled for maximal throughput.

The 320 resulting streams are passed directly to the `jfex_smle` encoder, which compresses the 18-bit energy values into the jFEX transmission format.

6.4.2.2 jFEX SMLE: Multi-Linear Encoder for Single Path

The `jfex_smle` block performs the same five-region piecewise-linear encoding as the `jfex_mle` described in Section 6.4.1.3, including the same special output codes for

disabled, invalid, out-of-range, and reserved cases. The encoding logic, thresholds, and assigned codes are identical and follow the jFEX specifications to ensure consistent energy representation across both processing paths.

The key difference lies in the input type and placement within the firmware pipeline. While `jfex_mle` receives 22-bit signed energy values resulting from the addition of multiple Super Cells in the adder path, the `jfex_smle` processes 18-bit signed energy values coming directly from individual Super Cells routed through the `jSSM` switch matrix.

This block is instantiated 320 times—once per output of the `jSSM` matrix—and operates fully in parallel, delivering encoded 12-bit energy codes in a single combinational cycle. The output of each `jfex_smle` instance is then passed to the data encoder described in Section 6.4.1.4, completing the single path processing chain for jFEX.

6.4.3 Integration into OSM

Firmware version 6.3 introduced modifications to the OSM block to accommodate the new data streams generated by the jFEX processing path. This required extending the OSM input space while preserving the interface with downstream serialization logic. The updated configuration, as illustrated in Figure 46, reflects the following input ordering:

- Inputs 0–15: eFEX path outputs,
- Inputs 16–17: jFEX adder path outputs,
- Inputs 18–33: jFEX single path outputs.

To accommodate the additional 18 jFEX data streams introduced in firmware version 6.3, the internal configuration of the OSM switch matrix was expanded to handle a total of 35 input slots. The selection architecture remained unchanged: each of the 48 output channels is associated with a dedicated 4-to-1 multiplexer that chooses among four statically assigned inputs. The selection among these inputs is performed dynamically at runtime using a two-bit select field, enabling flexible routing while preserving full throughput and compatibility with the existing OSUM output structure.

Despite these internal changes, the OSM block maintained full compatibility with previous firmware versions. The integration of the jFEX outputs preserves consistent formatting, synchronization, and integrity verification across all data paths, enabling seamless operation within the LATOME downstream infrastructure.

6.4.4 Timing Closure and Clock Tree Improvements

Firmware version 6.3 was the first LATOME release to achieve complete timing closure across all relevant clock domains. This result was made possible by a dedicated

effort from the firmware development team, who conducted a comprehensive analysis of the LATOME clock tree and placement constraints.

Prior versions employed cascaded PLL structures, which introduced clocking complexity and made it difficult to meet timing requirements across all regions of the firmware. The new approach adopted in version 6.3 eliminated the cascade, replacing it with a flatter and more modular clock tree where each domain is served by a dedicated PLL. This modification improved jitter characteristics, simplified timing constraints, and facilitated more reliable routing.

These improvements proved critical for ensuring stable clock transfers with the appropriate SOP delay selection and eliminating CRC inconsistencies previously observed. The updated clock tree contributed directly to the overall robustness of the design and to the successful deployment of the new HLS LATOME Firmware.

6.4.5 Simulation Campaign

Firmware version 6.3 introduced the new jFEX processing paths while reusing validated components such as REMAP and OSUM. To verify the full LATOME HLS data path prior to hardware deployment, a comprehensive simulation strategy was adopted, following the same layered approach used in earlier versions.

The campaign was structured as follows:

- **Layer 0:** Standalone simulation of the new blocks (*adder*, *single path*, and updated *osm*) using pure C++ models.
- **Layer 1:** Functional simulation of the complete pipeline in a single clock domain, validating interactions between HLS-generated and handcrafted components.
- **Layer 2:** Verification of the serializer/deserializer chain and Clock Domain Crossing (CDC) logic, including transitions from 320 MHz to 240 MHz (pre-ISM) and 240 MHz to 280 MHz (post-OSUM), confirming data integrity and latency alignment.
- **Layer 3:** Full integration using the top-level firmware and Python-based low-level software, covering the complete eFEX and jFEX paths.

Layers 1 through 3 were executed within a Cocotb-based testbench, enabling fast iterations and precise debugging. The simulations focused on verifying the correctness of the updated REMAP and OSUM blocks and ensuring seamless integration into the expanded firmware design.

A new enumeration class was introduced into the testbench infrastructure to enhance test coverage and streamline debugging. This class allowed the generation of structured input patterns—including all-zeros, all-ones, sequential indices, boundary

values, and custom data—enabling both stress testing and controlled verification of specific processing scenarios.

The simulation campaign relied on a firmware-aware software model that replicates the bit-accurate functional behavior of the DUT. Each test compared the DUT outputs against this golden reference to ensure correctness at each processing stage. Figure 49 illustrates one such comparison for the REMAP block, where all outputs for a single bunch crossing match exactly. This confirms that the remapping logic, including serializers and clock transfer blocks, functions correctly in simulation. However, since metastability cannot be captured in this environment, these results do not guarantee proper timing alignment in hardware. For this reason, dedicated hardware tests are necessary to determine the optimal SOP delay and ensure reliable data capture under real-world conditions. For clarity, custom formatting functions were introduced to present simulation outputs in a structured and readable layout.

BC 0:

Golden: ISM output

	OUT0	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8	OUT9
0	6003	4923	7209	6373	7128	5428	7810	7084	5812	7854
1	7208	4888	4933	4855	5369	6372	6853	5859	7882	5348
2	7036	6913	4451	6262	4741	7196	5118	7859	6101	7337
3	7204	6307	5895	6752	4358	6680	5972	4554	6283	4471
4	7069	6615	5012	4255	4100	5419	8098	6751	5179	4353
5	6998	7627	7565	7463	4902	4592	4539	7921	5655	6253
6	6865	4268	5981	5900	4578	5971	4677	4154	4868	6057
7	5831	7562	6649	5743	5448	6267	5147	7989	7970	7155
8	4421	4767	7530	6687	6051	6669	6232	5769	7338	4944
9	6526	5879	5364	5651	6432	7823	8082	7561	4172	4852
10	4575	4595	4564	6665	4561	5385	8001	4506	6686	6548
11	7850	4915	4696	5842	6955	4659	5181	6268	5168	6022
12	6035	5427	5362	7425	6140	5816	6993	4938	7670	5558
13	7079	5710	7629	4669	5116	6556	6667	6800	4459	7616
14	6378	4664	6382	6972	4322	5040	6808	7370	7668	5546
15	5918	7951	7866	5730	7229	4413	5400	4981	5653	6596

DUT: ISM OUTPUT

	OUT0	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8	OUT9
0	6003	4923	7209	6373	7128	5428	7810	7084	5812	7854
1	7208	4888	4933	4855	5369	6372	6853	5859	7882	5348
2	7036	6913	4451	6262	4741	7196	5118	7859	6101	7337
3	7204	6307	5895	6752	4358	6680	5972	4554	6283	4471
4	7069	6615	5012	4255	4100	5419	8098	6751	5179	4353
5	6998	7627	7565	7463	4902	4592	4539	7921	5655	6253
6	6865	4268	5981	5900	4578	5971	4677	4154	4868	6057
7	5831	7562	6649	5743	5448	6267	5147	7989	7970	7155
8	4421	4767	7530	6687	6051	6669	6232	5769	7338	4944
9	6526	5879	5364	5651	6432	7823	8082	7561	4172	4852
10	4575	4595	4564	6665	4561	5385	8001	4506	6686	6548
11	7850	4915	4696	5842	6955	4659	5181	6268	5168	6022
12	6035	5427	5362	7425	6140	5816	6993	4938	7670	5558
13	7079	5710	7629	4669	5116	6556	6667	6800	4459	7616
14	6378	4664	6382	6972	4322	5040	6808	7370	7668	5546
15	5918	7951	7866	5730	7229	4413	5400	4981	5653	6596

Figure 49 – Comparison between golden model and firmware outputs for the REMAP block. All values match.

The simulation using cocotb also provides detailed logging of any detected mismatches. Figure 50 illustrates this mechanism in action: for a particular bunch crossing, the output of the OSUM block diverges from the model, and the system flags the exact

frame, word index, and values for both the expected and actual results. This fine-grained feedback significantly accelerates debugging and functional validation of the firmware.

```
OSUM output data does not match for BC 31, frame 3, word 5. Expected: 0x691a7fe, Got: 0x8c227fe
OSUM output data does not match for BC 31, frame 3, word 6. Expected: 0xd471a068, Got: 0xd572308d
OSUM output data does not match for BC 31, frame 8, word 3. Expected: 0x83f103fe, Got: 0x8350d7fe
OSUM output data does not match for BC 31, frame 8, word 4. Expected: 0x4a517c49, Got: 0x4a50d034
```

Figure 50 – Example of mismatch detection in OSUM output: BC, frame, word, expected value (model), and actual value (DUT) are shown.

Finally, Figure 51 presents a simulation run in which no mismatches were detected between the firmware-aware model and the DUT. In this example, the complete HLS dataflow—including REMAP, ISM, OSUM, and output formatting—was validated successfully. The test encompassed six representative LATOME mapping families for one detector side: EMBA_1, EMECA_1, EMBA_EMECA_1, EMECA_HECA_1, FCAL1A, and FCAL2A. Although multiple LATOME mappings exist, these six families cover all relevant architectural configurations, as the internal structure within each family remains identical across its variants (e.g., EMBA_1, EMBA_2, etc.). Moreover, the C-side and A-side of the detector are symmetric in topology and processing requirements, further reducing the need for redundant tests. As such, this selection ensures exhaustive coverage while maintaining simulation efficiency. These results collectively confirm the functionality of the firmware design in simulation, establishing a solid foundation for subsequent hardware validation.

```
**************************************************************************************
** TEST                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
**************************************************************************************
** l2_sim.l2_sim_001              PASS         678.57           7.07         96.00  **
** l2_sim.l2_sim_002              PASS         682.14           3.08        221.66  **
** l2_sim.l2_sim_003              PASS         682.14           3.21        212.46  **
** l2_sim.l2_sim_004              PASS         682.14           3.31        206.22  **
** l2_sim.l2_sim_005              PASS         682.14           3.16        215.82  **
** l2_sim.l2_sim_006              PASS         682.14           3.17        214.93  **
**************************************************************************************
** TESTS=6 PASS=6 FAIL=0 SKIP=0              4089.29          32.01        127.74  **
**************************************************************************************
```

Figure 51 – Cocotb simulation summary showing successful validation of the complete LATOME HLS DUT (REMAP and OSUM blocks included).

6.4.6 Hardware Validation

While simulation offers detailed functional verification of the REMAP and OSUM blocks—including data integrity, latency alignment, and matching against a bit-accurate software model—it cannot capture the timing uncertainties inherent to real hardware. In particular, the clock domain crossing logic relies on proper phase alignment between clock transferring. Since metastability effects and latch timing violations do not manifest in

simulation, hardware tests remain essential to determine the correct SOP delay setting for stable data capture. This phase adjustment ensures that data is reliably latched at the output of serializers and deserializers, avoiding corruption due to marginal setup or hold conditions.

The following sections describe the validation procedures used in the laboratory to determine the optimal SOP delay for each LATOME mapping configuration.

6.4.6.1 REMAP Validation

Following the simulation campaign, the firmware was validated in the laboratory using the same methodology adopted for previous versions, based on the `mon_tdaq` infrastructure. The objective was to determine a metastability-free configuration for the `sop_delay_select` parameter, which governs the clock domain crossing from 320 MHz to 240 MHz within the REMAP block. Importantly, due to the revised clock tree design and improved placement strategies introduced in firmware version 6.3, the optimal delay setting had to be re-evaluated, as the phase alignment conditions were no longer guaranteed to match those of version 6.2.

Two types of tests were performed to validate the REMAP block under different operating conditions. First, a stress test was conducted using the EMBA_1 mapping, consisting of 50 consecutive configuration cycles per `sop_delay_select` value. This was followed by a broader mapping sweep, in which 12 representative LATOME regions were exercised with 10 runs for each delay configuration, ensuring coverage across diverse mapping topologies.

The results are summarized in Tables 9 and 10. All latching phases except 4 yielded a 100% success rate. Delay 4 consistently failed across all mappings and was interpreted as being within the metastability window. Among the successful configurations, delay 1 was chosen as the optimal operating point due to its maximum margin from unstable regions.

`sop_delay_select`	EMBA_1 (%)
0	100.00%
1	100.00%
2	100.00%
3	100.00%
4	0.00%
5	100.00%

Table 9 – Percentage of success on complete tests for firmware version v6.3.0 using the EMBA_1 mapping (50 completed tests)

sop_delay_select	EMBA_1	EMBC_1	EMBA_EMECA_1	EMBC_EMECC_1	EMECA_1	EMECC_1
0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
sop_delay_select	EMECA_HECA_1	EMECC_HECC_1	FCAL1A	FCAL1C	FCAL2A	FCAL2C
0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 10 – Percentage of success on complete tests for firmware version v6.3.0 across LATOME mappings.

Firmware version 6.3 was compiled with full timing closure across all clock domains, in contrast to version 6.2, which still exhibited minor timing violations. This was achieved through a dedicated clock tree study and improved placement constraints carried out by the firmware team. The impact of these improvements is clearly reflected in the results: while version 6.2 showed high—but not perfect—success rates in several delay configurations, version 6.3 exhibited a sharp separation between stable and unstable regions, with all viable delays achieving 100% across all mappings. This unambiguous behavior significantly simplifies the identification of metastability-free configurations and increases confidence in the robustness of the REMAP clock domain crossing logic under operational conditions.

6.4.6.2 OSUM Validation

The OSUM validation process evaluates the Start-of-Packet (SOP) generation and phase selection logic responsible for reliably managing the clock domain crossing from 240 MHz to 280 MHz. As in the REMAP block, a configurable delay selector (`sop_dly_sel`) determines the latching phase, while the `sop_refresh` control governs whether the SOP signal must be received externally or maintained internally. When `sop_refresh` is disabled, a circular shift register autonomously preserves the SOP pattern, ensuring stable operation even in the presence of small variations or glitches in the input SOP signal. Crucially, this mechanism guarantees that the `sop_out` signal remains unaffected by SOP period errors on the input, thus providing robust and deterministic synchronization. This mechanism is consistently used across all LATOME clock domain crossings.

For firmware version 6.3, a dedicated hardware test analogous to the `mon_tdaq` infrastructure was developed for the OSUM path. This test analyzes the output of the firmware on all 48 fiber channels and reports a boolean `data_result` indicating whether each event passed or failed the validation. The dataset used for this validation was `48ch-48ch-EMBA_1-3564-constant_id`, which belongs to the EMBA mapping and

features fixed identifiers per Super Cell stream. This facilitates unambiguous tracking of data through the full processing chain, making it ideal for mapping verification.

The integration of the Mini-FEX monitoring logic in firmware version 6.3 enabled real-time observation of internal error counters during both laboratory and ATLAS-level tests, significantly improving the granularity and interpretability of validation metrics. This validation, was focused on four indicators: data errors, `sop_240_period_errors`, `sop_280_period_errors`, and CRC errors.

Table 11 summarizes the results of a stress test in which the `sop_refresh` parameter was kept enabled throughout. The test was executed 50 times for each `sop_delay_select` setting, using the EMBA_1 mapping. The percentages in the table reflect the proportion of test runs in which errors were observed (or, in the case of data success, correctly matched results between firmware and expected outputs).

`sop_delay_select`	Data Success (%)	CRC Errors (%)	240 MHz SOP Errors (%)	280 MHz SOP Errors (%)
0	84.00	14.00	14.00	100.00
1	78.00	14.00	14.00	14.00
2	82.00	16.00	16.00	16.00
3	44.00	56.00	12.00	12.00
4	0.00	100.00	16.00	16.00
5	48.00	52.00	16.00	16.00
6	80.00	20.00	20.00	20.00

Table 11 – Stress test results for the OSUM block on the EMBA_1 mapping, with 50 runs per `sop_delay_select` configuration. All values represent the percentage of tests (out of 50) in which errors were detected or data matched successfully.

The initial OSUM validation campaign, with `sop_refresh` continuously enabled, revealed unstable behavior and inconsistent error margins across several `sop_delay_select` values. After observing these issues, a focused debugging session identified that keeping `sop_refresh` permanently asserted allowed external SOP timing variations to interfere with the internal SOP phase generator. This prompted the execution of a second batch of tests under more realistic conditions.

In this second campaign, `sop_refresh` was enabled at the start of the run but deasserted after one second, allowing the circular shift register to autonomously maintain the SOP signal for the remainder of the test. Table 12 summarizes the results of this updated configuration. A clear improvement in stability is observed: all `sop_delay_select` values, except delays 4, 5, and 6, achieved 100% data success with no period or CRC errors. Delay 4 once again failed entirely and is identified as the metastability window, where the sampled SOP phase likely falls near a clock edge and causes non-deterministic behavior. Delay 5 showed CRC instability, while delay 6 was nearly error-free but showed a slight reduction in reliability.

REMAP SOP	sop_delay_select	Data Success (%)	CRC Errors (%)	240 MHz SOP Errors (%)	280 MHz SOP Errors (%)
2	0	100.00	0.00	0.00	0.00
2	1	100.00	0.00	0.00	0.00
2	2	100.00	0.00	0.00	0.00
2	3	100.00	0.00	0.00	0.00
2	4	0.00	100.00	0.00	0.00
2	5	90.00	74.00	0.00	0.00
2	6	98.00	0.00	0.00	0.00

Table 12 – OSUM validation results with `sop_refresh` deasserted after 1 second of initialization, using the EMBA_1 mapping. Each `sop_delay_select` was tested over 50 runs. The REMAP SOP was fixed at delay 2 for all tests. All values represent the percentage of tests in which errors were detected or successful outcomes achieved.

These results confirm that disabling `sop_refresh` after initialization stabilizes the SOP signal and eliminates residual phase inconsistencies introduced by the external SOP source. Delays 1 and 2 stood out as the most reliable options, being maximally distant from the metastability zone and exhibiting no data or synchronization errors across all 50 runs. This test provides the final validation of the OSUM clock domain crossing in firmware version 6.3.

To further reinforce these results, successful tests performed directly on the ATLAS detector demonstrated the same behavior in a real deployment. As shown in Figures 52 and 53, both the REMAP and OSUM blocks initially registered a burst of SOP period and CRC errors during system startup. However, once `sop_refresh` was programmatically set to zero across all LATOME instances, all error counters converged to zero. These operational results validate the robustness of the internal SOP propagation mechanism, confirming its effectiveness in maintaining synchronization across all fiber channels under realistic deployment conditions at the ATLAS detector.

Figure 52 – SOP period error counters for the REMAP and OSUM blocks during ATLAS online testing. Errors are observed at startup but converge to zero after `sop_refresh` is disabled.

Figure 53 – Evolution of CRC errors and `sop_refresh` values over time. The drop in errors aligns with the moment the `sop_refresh` signal is set to zero.

7 Conclusion

This thesis presented the validation and simulation of the upgraded digital signal processing pipeline implemented through High-Level Synthesis (HLS) on the real-time trigger path of the ATLAS Liquid Argon Calorimeter, specifically within the LATOME firmware. A structured and modular approach was adopted to understand, simulate, and verify the design and integration of the REMAP and OSUM blocks across firmware versions 6.0 through 6.3, leveraging a multistage simulation strategy and advanced monitoring mechanisms.

The role of LATOME in the Phase-I upgrade of the ATLAS LAr calorimeter was first contextualized, emphasizing the importance of fine-grained Super Cell readout and the need for robust digital signal handling across multiple clock domains. The transition to HLS-based IP blocks introduced architectural shifts that required careful simulation and validation. The demonstration firmware introduced in versions 6.0 and 6.1 was used to dissect the internal structure of the REMAP and the newly introduced Input Switch Matrix, highlighting their reliance on parallel interfaces, clock domain crossing strategies, and serializer/deserializer logic. In these initial versions, validation efforts concentrated on the monitoring path and on the functionality of the REMAP logic.

Firmware version 6.2 marked the introduction of the *eFEX* processing path, extending the validation scope to include the Output Summing (OSUM) block for the first time, thereby initiating validation of the real-time trigger path. This version also enabled the application of the full Firmware-Aware simulation model to end-to-end signal paths involving both REMAP and OSUM. Firmware 6.3 subsequently introduced support for the *jFEX* path, further increasing architectural complexity and necessitating precise validation across multiple high-speed data channels. To support debugging and simulation during these stages, a lightweight internal monitoring module—the Mini-FEX—was incorporated. This module exposed real-time counters for SOP period and CRC errors, significantly enhancing observability both in simulation and during ATLAS deployment.

The multi-layered simulation strategy, structured into three progressive levels of complexity, enabled thorough testing from standalone HLS components to full LATOME integration, including interactions with legacy RTL blocks. By employing both Firmware-Agnostic and Firmware-Aware models, this thesis validated the consistency of remapping and output summing operations across all 116 mappings, demonstrating strong agreement between model and firmware outputs, as well as consistent and deterministic behavior across all tested configurations.

A major contribution of this work lies in the validation of clock synchronization and data integrity mechanisms introduced in firmware version 6.3. Extensive SOP delay calibration campaigns were conducted, and the impact of the `sop_refresh` signal across the

OSUM block was systematically analyzed. Stress test results identified the metastability window and highlighted optimal delay configurations, while lab tests and real-time ATLAS tests confirmed that disabling `sop_refresh` after initialization effectively eliminated CRC and SOP period errors. The Mini-FEX block, integrated at the top-level of OSUM, proved instrumental in tracking and interpreting these behaviors both in simulation and operational environments.

The validation workflow developed and exercised throughout this thesis provides a robust methodology for the future integration of HLS-generated modules into the LATOME firmware. It ensures that the transition from traditional RTL design to HLS does not compromise timing closure, functional correctness, or system reliability under the demanding conditions of the LHC trigger environment.

Future developments, including the implementation of the gFEX path and the User Code block in firmware version 7.x, will build directly upon the simulation infrastructure and validation strategies established here. As HLS continues to mature within the LAr firmware ecosystem, the approaches and results presented in this thesis provide a foundation for subsequent design phases and for other subdetectors pursuing similar high-throughput, low-latency digital signal processing solutions.

In conclusion, the work presented provides a foundation for further developments as the LHC and the LATOME firmware continue to evolve. These enhancements are essential for adapting to the increased demands of future LHC operations and for ensuring that the ATLAS experiment remains at the forefront of particle physics research. By tackling the challenges introduced by higher luminosity and the resulting demands on real-time signal processing, this work supports the continued evolution of the ATLAS trigger system and reinforces the technological foundation required for future discoveries at the LHC.

A HLS Wrapper Generator Output

This appendix presents the VHDL code relevant to the HLS wrapper generation tool described in Section 5.3. It includes the original flattened entity generated by Catapult HLS, the automatically generated wrapper that restructures the signals using named array types, and the corresponding package file containing type definitions.

A.1 Original Entity Generated by Catapult

```
entity ism is
  port(
    clk : IN STD_LOGIC;
    rst : IN STD_LOGIC;
    x_rsc_dat : IN STD_LOGIC_VECTOR (4991 DOWNTO 0);
    x_triosy_lz : OUT STD_LOGIC;
    s_sel : IN STD_LOGIC_VECTOR (1599 DOWNTO 0);
    s_ena : IN STD_LOGIC_VECTOR (319 DOWNTO 0);
    y_rsc_dat : OUT STD_LOGIC_VECTOR (4159 DOWNTO 0);
    y_triosy_lz : OUT STD_LOGIC;
    bcid_in_rsc_dat : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
    bcid_in_triosy_lz : OUT STD_LOGIC;
    bcid_out_rsc_dat : OUT STD_LOGIC_VECTOR (11 DOWNTO 0);
    bcid_out_triosy_lz : OUT STD_LOGIC
  );
end ism;
```

Listing A.1 – Original Catapult-generated `ism` entity with flattened signals.

A.2 Generated Wrapper Entity

```
entity ism_wrapper is
    port (
        clk : IN  STD_LOGIC;
        rst : IN  STD_LOGIC;
        x_rsc_dat : IN array13_t(0 to 383);
        x_triosy_lz : OUT  STD_LOGIC;
        s_sel : IN array5_t(0 to 319);
        s_ena : IN  STD_LOGIC_VECTOR(319 DOWNTO 0);
        y_rsc_dat : OUT array13_t(0 to 319);
        y_triosy_lz : OUT  STD_LOGIC;
        bcid_in_rsc_dat : IN  STD_LOGIC_VECTOR(11 DOWNTO 0);
        bcid_in_triosy_lz : OUT  STD_LOGIC;
        bcid_out_rsc_dat : OUT  STD_LOGIC_VECTOR(11 DOWNTO 0);
        bcid_out_triosy_lz : OUT  STD_LOGIC
```

```
15      );
16  end ism_wrapper;
17
18  architecture Behavioral of ism_wrapper is
19      signal x_rsc_dat_flat: std_logic_vector(4991 downto 0);
20      signal s_sel_flat: std_logic_vector(1599 downto 0);
21      signal y_rsc_dat_flat: std_logic_vector(4159 downto 0);
22  begin
23      -- Flatten input signal x_rsc_dat
24      gen_x_rsc_dat_unflatten: for i in 0 to 383 generate
25          x_rsc_dat_flat((i*13) + (13-1) downto i*13) <= x_rsc_dat(i)
                ((13-1) downto 0);
26      end generate gen_x_rsc_dat_unflatten;
27
28      -- Flatten input signal s_sel
29      gen_s_sel_unflatten: for i in 0 to 319 generate
30          s_sel_flat((i*5) + (5-1) downto i*5) <= s_sel(i)((5-1) downto 0)
                ;
31      end generate gen_s_sel_unflatten;
32
33      -- Unflatten output signal y_rsc_dat
34      gen_y_rsc_dat_unflatten: for i in 0 to 319 generate
35          y_rsc_dat(i)((13-1) downto 0) <= y_rsc_dat_flat((i*13) + (13-1)
                downto i*13);
36      end generate gen_y_rsc_dat_unflatten;
37
38      ism_inst: entity work.ism
39          port map (
40              clk => clk,
41              rst => rst,
42              x_rsc_dat => x_rsc_dat_flat,
43              x_triosy_lz => x_triosy_lz,
44              s_sel => s_sel_flat,
45              s_ena => s_ena,
46              y_rsc_dat => y_rsc_dat_flat,
47              y_triosy_lz => y_triosy_lz,
48              bcid_in_rsc_dat => bcid_in_rsc_dat,
49              bcid_in_triosy_lz => bcid_in_triosy_lz,
50              bcid_out_rsc_dat => bcid_out_rsc_dat,
51              bcid_out_triosy_lz => bcid_out_triosy_lz
52          );
53  end Behavioral;
```

Listing A.2 – Wrapper entity automatically generated for simulation and modular integration.

A.3 Generated Package File

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 package latome_hls_pkg is
5     type array13_t is array (natural range <>) of std_logic_vector(12
        downto 0);
6     type array5_t  is array (natural range <>) of std_logic_vector(4
        downto 0);
7 end latome_hls_pkg;
```

Listing A.3 – VHDL package used for array signal types in the wrapper.

B Example of LATOME Mapping Configuration: EMBA_1

This appendix presents the complete mapping configuration for the EMBA_1 region, corresponding to the LATOME input mapping version 03.00.00. This file serves as the basis for understanding how detector signals are routed into the LATOME firmware and highlights the correspondence between input fibers, LTDBs, MTPs, and Super Cell identifiers.

```
# LATOME input mapping v03.00.00
# Comments: ATLAS mapping (folder names changed)
# AMCType: EMB
# Side: A
# QuadrantID: 0
# AMCHashNumber: 0
# CarrierID: 1
# AMCNumberInCarrier: 1

# Input information section
# Number connected LTDBs: 2
# Connected LTDBs: EMB(I01L)  EMB(I02R)
# Number connected input MTPs: 4
# Number connected input fibers: 40
# Input_ID LTDB_MTP_PIN RX_PIN SC1 ... SC8
IN F3   I01L_01_10  RX01_03  B_1A_B1  B_2A_B1  B_4A_B1  B_3A_B1
   B_6A_B1  B_5A_B1  B_8A_B1  B_7A_B1
IN F4   I01L_01_09  RX01_04  B_2A_M2  B_2A_M1  B_2A_M4  B_2A_M3
   B_1A_M3  B_1A_M4  B_1A_M1  B_1A_M2
IN F5   I01L_01_08  RX01_05  B_3A_M1  B_3A_M2  B_3A_M4  B_3A_M3
   B_4A_M4  B_4A_M3  B_4A_M2  B_4A_M1
IN F6   I01L_01_07  RX01_06  B_6A_F4  B_6A_F3  B_6A_F2  B_6A_F1
   B_5A_F1  B_5A_F2  B_5A_F3  B_5A_F4
...
IN F48  I02R_04_01  RX04_12  B_7D_B1  B_8D_B1  B_6D_B1  B_5D_B1
   B_1D_B1  B_2D_B1  B_3D_B1  B_4D_B1

# Output information section
# Number connected output fibers: 38 (unique=19)
# Number eFEX fibers: 30 (unique=16)
# Number jFEX fibers: 6 (unique=2)
# Number gFEX fibers: 2 (unique=1)
# hash_ID Type C1 C2 ....
OUT F1   eFEX   B_7A          B_8A
```

```
30 OUT F5   eFEX   B_7A          B_8A
31 ...
32 OUT F48  gFEX   B_1A+B_2A+B_1B+B_2B  B_1C+B_2C+B_1D+B_2D  B_3A+
      B_4A+B_3B+B_4B  B_3C+B_4C+B_3D+B_4D  B_5A+B_6A+B_5B+B_6B  B_5C+
      B_6C+B_5D+B_6D  B_7A+B_8A+B_7B+B_8B  B_7C+B_8C+B_7D+B_8D
```

Bibliography

[1] PERKINS, Donald H. *Introduction to High Energy Physics*. Cambridge University Press 2000.

[2] WILLIE, Klaus. *The physics of particle accelerators: An introduction*. Clarendon 2000.

[3] CERN. *The Large Hadron Collider*. Available at: `https://home.cern/science/accelerators/large-hadron-collider`. Accessed on: Jan. 16, 2025.

[4] CERN. *High-Luminosity LHC*. Available at: `https://home.cern/resources/faqs/high-luminosity-lhc` Accessed on: Jan. 16, 2025.

[5] MOUCHE, P *Overall View of the LHC*. CERN Document Server, OPEN-PHO-ACCEL-2014-001, Jun 2014.

[6] AAD, G. et al. *The Phase-I trigger readout electronics upgrade of the ATLAS Liquid Argon calorimeters.* Journal of Instrumentation, vol. 17, no. 5, P05024, 2022. Available at: `https://doi.org/10.1088/1748-0221/17/05/P05024`. Accessed on: July 18, 2024.

[7] AAD, G. et al. *Readiness of the ATLAS Liquid Argon Calorimeter for LHC collisions*. European Physical Journal C, vol. 70, pp. 723-753, 2010. Available at: `https://doi.org/10.1140/epjc/s10052-010-1354-y`. Accessed on: Jan. 20, 2025.

[8] ATLAS Collaboration. *Trigger and Data Acquisition System*. ATLAS Experiment. Available at: `https://atlas.cern/Discover/Detector/Trigger-DAQ`. Accessed on: Jan. 20, 2025.

[9] CERN. *ATLAS*. Available at: `https://www.home.cern/science/experiments/atlas`. Accessed on: Jan. 21, 2025.

[10] EVANS, L. R.; BRYANT, P. *LHC Machine*. Journal of Instrumentation, vol. 3, pp. S08001.164, 2008.

[11] ATLAS Collaboration, ATLAS experiment schematic or layout illustration, CERN Document Server. Available: `https://cds.cern.ch/record/2837191`.

[12] CERN. *The ATLAS Inner Detector*. Available at: `https://atlas-public.web.cern.ch/Discover/Detector/Inner-Detector`. Accessed on: Jan. 27, 2025.

[13] CERN. *The ATLAS Magnet System*. Available at: `https://atlas-public.web.cern.ch/Discover/Detector/Magnet-System`. Accessed on: Jan. 20, 2025.

[14] CERN. *The ATLAS Muon Spectrometer*. Available at: `https://atlas-public.web.cern.ch/Discover/Detector/Muon-Spectrometer`. Accessed on: Jan. 20, 2025.

[15] CERN. *The ATLAS Calorimeter*. Available at: https://atlas-public.web.cern.ch/Discover/Detector/Calorimeter. Accessed on: Jan. 20, 2025.

[16] G. Aad et al. *The ATLAS Trigger System for LHC Run 3 and Trigger Performance in 2022*. Journal of Instrumentation, vol. 19, no. 6, P06029, 2024. DOI: 10.1088/1748-0221/19/06/P06029. Available at: https://cds.cern.ch/record/2887853/files/ATL-DAQ-PROC-2024-002.pdf. Accessed on: Jan. 27, 2025.

[17] ATLAS liquid-argon calorimeter: Technical Design Report. Place: Geneva Series: Technical design report. ATLAS. 1996. DOI: 10.17181/CERN.FWRW.FOOQ. Available at: https://cds.cern.ch/record/331061. Accessed on: Jan. 22, 2025.

[18] ATLAS Collaboration. *Performance of the ATLAS Trigger System in 2010*. European Physical Journal C, vol. 72, 2012, p. 1849. Available at: https://arxiv.org/abs/1110.1530. Accessed on: Jan. 29, 2025.

[19] ATLAS Collaboration. *Upgrading the experiment for LHC Run 3*. Available at: https://atlas.cern/Discover/Detector/Long-Shutdown-2. Accessed on: Jan. 29, 2025.

[20] ATLAS Collaboration. "The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022". In: Journal of Instrumentation 19.6 (June 1, 2024), P06029. ISSN: 1748-0221. DOI: 10.1088/1748-0221/19/06/P06029. arXiv: 2401.06630[hep-ex]. Available at: http://arxiv.org/abs/2401.06630. Accessed on: Jan. 22, 2025.

[21] CERN. *High-Luminosity LHC (HL-LHC)*. Available at: https://voisins.cern/en/high-luminosity-lhc-hl-lhc. Accessed on: Feb. 13, 2025.

[22] ATLAS Collaboration. *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*. CERN, Geneva, Sep. 2013. CERN-LHCC-2013-018, ATLAS-TDR-023. Available at: https://cds.cern.ch/record/1602235. Accessed on: July 18, 2024.

[23] RONDOT, Paolo. *Leveraging Space and Time-Division Multiplexing Logic for Trigger Systems in High Energy Physics using High-Level Synthesis*. Master's Thesis Nr. 510, Systems Group, Department of Computer Science, ETH Zurich, in collaboration with CERN, April 2024–September 2024.

[24] KLEEMAN, L.; CANTONI, A. *Metastable Behavior in Digital Systems*. IEEE Design & Test of Computers, vol. 4, no. 6, pp. 4–19, Dec. 1987. Available at: https://doi.org/10.1109/MDT.1987.295189. Accessed on: Jan. 16, 2025.