amU
Aix Marseille Université

NNT : 2024AIXM0001

THÈSE DE DOCTORAT

Soutenance à AMU — Aix-Marseille Université
le 3 juillet 2025 par

Alberto PERRO

Emerging Technologies and Development Methodologies for High-Throughput Data Acquisition Systems

Discipline
Physique et Sciences de la Matière

Spécialité
Instrumentation

École doctorale
ED 352 - Physique et Sciences de la Matière

Laboratoire/Partenaires de recherche
European Organization for Nuclear Research

Composition du jury

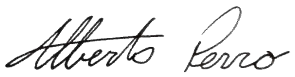
Mossadek TALBY CPPM, Aix-Marseille Université	Président du jury
Renaud LE GAC CPPM, Aix-Marseille Université	Directeur de thèse
Olivier LEROY CPPM, Aix-Marseille Université	Co-directeur de thèse
Pascal VINCENT LPNHE, Sorbonne Université	Rapporteur
Cynthia HADJIDAKIS IJCLab, Université Paris-Saclay	Rapporteure
Vincent TISSERAND LPCA, Université Clermont Auvergne	Examinateur
Paolo DURANTE CERN	Membre invité

Affidavit

I, undersigned, Alberto Perro, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific supervision of Renaud Le Gac and Olivier Leroy, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the french national charter for Research Integrity and AMU charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, 3rd July 2025



Liste de publications et participation aux conférences

Liste des publications et/ou brevets réalisées dans le cadre du projet de thèse:

1. A low-cost, low-power media converter solution for next-generation detector readout systems
   A. Perro et al 2025 JINST 20 C02027 [61] http://doi.org/10.1088/1748-0221/20/02/C02027

2. Guidelines for FPGA Gateware Development in LHCb
   A. Perro et al 2025 LHCb-PUB-2025-009 [63] https://cds.cern.ch/record/2929526

3. Evaluating FPGA Acceleration with Intel® oneAPI Toolkit for High-Speed Data Processing
   A. Perro et al 2025 (*submitted and accepted*) https://indi.to/LdBc7

Participation aux conférences et écoles d'été au cours de la période de thèse:

1. Topical Workshop on Electronic for Particle Physics 2024 - Glasgow, Scotland [61] https://indi.to/jsW6K https://indi.to/jKpgR

2. Conference on Computing in High Energy and Nuclear Physics 2024 - Krakow, Poland https://indi.to/LdBc7

3. FPGAWorld 2024 - Stockholm, Sweden https://youtu.be/qs6Lh95qiYk

4. FPGA Developers' Forum 2024 - Geneva, Switzerland https://indi.to/fYDfT

5. International School on Trigger and Data Acquisition 2023 - Istanbul, Turkey https://indi.to/hYTfw

Abstract and keywords

High Energy Physics experiments depend heavily on FPGA-based data acquisition systems to manage the massive data volumes produced by particle detectors. With CERN upgrading its accelerator complex, experiments such as LHCb must handle higher throughput, necessitating the development of advanced FPGA-based readout systems. To meet these demands, more efficient design methodologies are required to streamline development while maintaining system reliability.

This work explores modern approaches to improving FPGA gateware development, focusing on verification-driven design and open-source solutions. Functional verification frameworks such as UVVM, OSVVM, and VUnit were evaluated and implemented to automate testing, improve test coverage, and create robust verification environments. Formal verification techniques were applied to critical components, demonstrating their ability to detect design flaws early and significantly reduce debugging time.

A major contribution of this research is the development of the colibri library, an open-source, vendor-independent collection of reusable FPGA components. By standardizing commonly used functionalities, colibri improves portability across different hardware platforms and simplifies integration. The library was successfully applied in various projects, including a high-speed Ethernet-based readout system for detector data acquisition.

High-level synthesis was also explored as a means to simplify FPGA development by allowing designs to be written in C++ instead of traditional hardware description languages. While this approach showed potential for accelerating trigger and reconstruction algorithms, current high-level synthesis tools still present challenges in achieving the performance and reliability required for large-scale applications.

The findings of this work highlight the benefits of open-source methodologies, structured verification strategies, and reusable component libraries in the development of FPGA-based readout systems for high-energy physics experiments. These results will help guide the design of next-generation data acquisition systems at LHCb.

Keywords: high-energy physics, data acquisition systems, FPGA, functional verification, formal verification, high-level synthesis

Résumé et mots clés

Les systèmes d'acquisition de données basés sur les field-programmable gate array (FPGA) revêtent une importance critique pour les expériences de physique des hautes énergies, notamment pour gérer les énormes volumes de données produits par les détecteurs de particules. Avec la mise à niveau de l'accélérateur du CERN, des expériences comme le LHCb doivent gérer un débit de données toujours plus élevé. Ceci nécessite le développement de systèmes d'acquisition avancés basés sur les FPGA. Pour répondre à ces exigences, des méthodologies de design efficaces sont nécessaires afin de faciliter le développement tout en garantissant la fiabilité du système.

La présente thèse propose des approches innovantes pour améliorer le développement du gateware FPGA, en mettant l'accent sur la conception basée sur la vérification et les solutions open source. Des cadres de vérification fonctionnelle tels que UVVM, OSVVM et VUnit ont été évalués et mis en œuvre pour automatiser les tests, en améliorer la couverture et créer des environnements de vérification robustes. Des techniques de vérification formelle ont été appliquées aux composants critiques, et ont démontré leur capacité à détecter les erreurs de conception dès les premières étapes tout en réduisant considérablement le temps de mise au point.

Une contribution majeure de cette recherche est le développement de la bibliothèque colibri, une collection open source et indépendante des fournisseurs de composants FPGA. En standardisant les fonctionnalités les plus utilisées, colibri améliore la portabilité entre différentes plateformes matérielles et simplifie l'étape d'intégration. La bibliothèque a été utilisée pour divers projets, notamment un système de lecture haute vitesse basé sur Ethernet pour l'acquisition de données des détecteurs.

La possibilité pour la synthèse de haut niveau de simplifier le développement FPGA a également été explorée, car elle permet l'écriture de code directement en C++ au lieu des langages traditionnels. Bien que cette approche ait montré un potentiel pour accélérer les algorithmes de « trigger » et de reconstruction, il reste de la marge de progression pour les outils actuels pour la synthèse de haut niveau en matière de performance et de fiabilité pour les applications à grande échelle.

Les résultats de cette étude mettent en évidence les avantages d'une approche open source, des stratégies de vérification structurées et des bibliothèques de composants réutilisables dans le développement des systèmes d'acquisition basés sur FPGA pour les expériences de physique des hautes énergies. Ces travaux contribuent à améliorer la conception des futurs systèmes d'acquisition de données pour le LHCb.

Mots clés : physique des hautes énergies, systèmes d'acquisition de données, FPGA, vérification fonctionnelle, vérification formelle, synthèse de haut niveau

Résumé étendu

L'Organisation européenne pour la recherche nucléaire (CERN) est le plus grand laboratoire de physique des hautes énergies au monde. Au cœur du complexe des accélérateurs du CERN se trouve le Grand collisionneur de hadrons (LHC), un accélérateur circulaire de 27 km de long. Dans ce collisionneur, des faisceaux de protons sont accélérés jusqu'à 7 TeV et entrent en collision en quatre points, où sont installées les expériences ATLAS, CMS, ALICE et LHCb. Ces expériences possèdent chacune un programme scientifique distinct, visant à faire progresser nos connaissances du Modèle standard de la physique des particules et la recherche de nouvelles physiques. Le collisionneur et ses expériences fonctionnent en alternance entre des périodes de prise de données, appelées *runs*, et des phases d'arrêt. Durant ces arrêts, l'accélérateur et les expériences peuvent être améliorés afin d'optimiser leurs performances et leur sensibilité.

LHCb L'expérience LHCb étudie principalement les propriétés des quarks charmé et beau. Les mesures collectées lors des deux premiers runs, de 2010 à 2018, ont permis d'explorer les propriétés des différentes saveurs et de rechercher des signes de nouvelle physique grâce à des mesures de précision de processus de désintégration rares. LHCb a déterminé avec précision l'angle de CKM γ en étudiant la violation de CP dans les désintégrations de B^0 et B_s^0. Ils ont observé des désintégrations extrêmement rares médiées par des courants neutres changeant de saveur, et découvert de nouveaux hadrons exotiques, y compris des tétraquarks et des pentaquarks.

Lors du Long Shutdown 2 (2018-2022), le détecteur a subi une mise à niveau majeure, améliorant sa sensibilité aux processus de physique de la saveur. Cette amélioration permet à LHCb de collecter 50 fb^{-1}, soit cinq fois la luminosité des runs précédents. Le programme de recherche se concentre sur les mesures de violation de CP, qui sont fortement contraintes par des considérations à la fois expérimentales et théoriques. De plus, l'augmentation de la luminosité améliorera les mesures des désintégrations rares en augmentant considérablement l'ensemble de données disponible.

Le Long Shutdown 3 (2026-2029) marquera la transition vers la phase de haute luminosité du LHC, fournissant 300 fb^{-1} de luminosité intégrée au point de collision de LHCb. Pour exploiter pleinement l'augmentation du taux de collisions, il est prévu une mise à niveau majeure du LHCb, aussi appelée le *LHCb Upgrade II*. Cette mise à niveau améliorera considérablement les capacités de collecte de données de l'expérience, surpassant celles des autres expériences et doublant effectivement le spectre d'énergie accessible pour les études de physique de la saveur de précision.

La phase de haute luminosité présente des défis importants en raison de l'augmentation des multiplicités de particules. Tous les détecteurs devront être capable

de résister à des environnements de radiation plus intenses et de transmettre des données à des taux plus élevés. En conséquence, les systèmes de déclenchement et d'acquisition de données devront aussi être améliorées pour gérer l'augmentation du débit de données.

Le Détecteur Le détecteur LHCb a une conception unique qui maximise la section efficace de production des paires de quarks beau et charmé lors de collisions proton-proton. Il est conçu comme un spectromètre, tirant parti de la haute pseudorapidité de ces paires de quarks qui est due aux asymétries dans la distribution des impulsions des partons. Le détecteur est composé de plusieurs sous-détecteurs, chacun contribuant à la mesure de propriétés spécifiques des particules. Le premier groupe de sous-détecteurs constitue le système de tracking, qui comprend trois composants clés : le Vertex Locator (VeLo), l'Upstream Tracker (UT) et le Scintillating Fibre Tracker (SciFi). Le VeLo est le sous-détecteur le plus proche du point de collision. Son objectif principal est d'identifier les vertex primaires et de mesurer le paramètre d'impact des particules chargées. Ces mesures sont essentielles pour reconstruire les événements impliquant des hadrons charmé et beau, qui ont des durées de vie courtes et se désintègrent généralement après avoir parcouru environ 1 cm. Le VeLo est constitué de deux parties rétractables qui se referment autour de la ligne de faisceau uniquement lorsque les conditions de faisceau sont stables. Cela permet de protéger les capteurs en pixels de silicium tout en permettant une résolution du point d'impact élevée. L'Upstream Tracker est composé de quatre couches de pistes en silicium. Les données de l'UT, combinées aux informations du VeLo, permettent une estimation rapide de l'impulsion et de l'impulsion transverse des particules chargées. Cette capacité facilite une reconstruction approximative en temps réel des trajectoires des particules, réduisant ainsi les besoins de calcul tout en améliorant la qualité de la reconstruction.

Un aimant conventionnel est utilisé pour courber les trajectoires des particules chargées, permettant des mesures précises des leurs impulsions. L'aimant possède une intensité de champ de 4 $T \cdot m$, et sa polarité peut être inversée afin d'atténuer les asymétries de la configuration du détecteur qui pourraient autrement affecter les mesures de violation de CP.

Au-delà de l'aimant, le Scintillating Fibre Tracker (SciFi) complète le système de tracking. Le SciFi est composé de trois stations, chacune contenant quatre couches. Il utilise des fibres scintillantes de 2,5 m de long comme milieu de détection actif, offrant une excellente résolution spatiale.

Le deuxième grand système de détection est dédié à l'identification des particules et comprend trois sous-détecteurs : les détecteurs à imagerie par anneaux de Cherenkov (RICH), les calorimètres et les stations à muons.

Les détecteurs RICH exploitent l'effet Cherenkov pour mesurer la vitesse des particules. Une particule chargée traversant un milieu induit des dipôles électriques temporaires, entraînant l'émission de rayonnement électromagnétique. Si la vitesse de la particule dépasse celle de la lumière, une interférence constructive produit un front d'onde à un angle spécifique, appelé l'angle de Cherenkov. L'anneau de Cherenkov émis est détecté par des photomultiplicateurs, et son rayon est reconstruit

pour déterminer la masse de la particule. LHCb utilise deux détecteurs RICH couvrant différentes gammes de impulsions.

Le système de calorimétrie extrait de l'information relative à l'énergie et est composé de deux sous-systèmes : l'un dédié aux hadrons et l'autre aux particules interagissant électromagnétiquement. Les deux calorimètres sont conçus comme des calorimètres à échantillonnage, avec des couches alternées d'absorbeurs et de scintillateurs. L'énergie d'une particule est estimée en mesurant sa profondeur de pénétration.

Le dernier système de sous-détection est dédié à l'identification des muons. Les muons jouent un rôle crucial dans la physique de LHCb, car ils apparaissent fréquemment dans les états finaux des désintégrations des hadrons beauty. Étant donné que les muons sont hautement pénétrants, les stations à muons sont placées après tous les autres sous-détecteurs. Chaque station à muons est constituée d'une couche de détection active basée sur des chambres proportionnelles multi-fils et d'une couche d'absorption en fer.

Acquisition des Données Le LHCb fonctionne avec une fréquence de croisement de paquets de 40 MHz. Le système d'Acquisition des Données (DAQ) doit identifier et stocker efficacement les événements d'intérêt pour les analyses physiques tout en écartant les événements non pertinents. Lors des Runs 1 et 2, le système DAQ avait une conception classique où le premier filtre de sélection était implémenté via un déclencheur matériel. Cependant, ce système constituait un goulot d'étranglement pour les désintégrations de signaux hadroniques. Cela empêcher de tirer pleinement parti de l'augmentation de la luminosité. En conséquence, le système DAQ a été entièrement repensé pour adopter une approche de sélection et de désintégration entièrement basée sur un logiciel.

Dans ce nouveau système, l'électronique frontale doit digitaliser les données des interactions par croisement de faisceaux à une fréquence de 40 MHz. Cela est réalisé grâce à des circuits intégrés spécifiques (ASICs) et à une liaison à haute débit résistante aux radiations, appelée GBT. Les données provenant du frontale, situé dans la caverne expérimentale, sont transmises vers le centre de serveurs d'acquisition, en surface à travers 11 000 fibres optiques.

Les données sont ensuite décodées par une carte de lecture personnalisée basée sur FPGA, connue sous le nom de PCIe40. Ces cartes existent en trois variantes : superviseurs de lecture, cartes de contrôle et cartes de lecture. Les deux premières, appelées SODIN et SOL40, synchronisent la lecture des événements, distribuent le signal d'horloge et contrôlent l'électronique frontale. La troisième, appelée TELL40, reçoit les données des détecteurs, construit les fragments d'événements et les transmet au Event Builder. Ces cartes sont hébergées dans les serveurs, et constituent la première étape de l' Event Builder. L'Event Builder est un système à haut débit chargé d'assembler les fragments d'événements provenant des différentes parties du détecteur en événements complets.

Ce processus doit être réalisé à la fréquence complète de croisement de paquets de 40 MHz, correspondant à un débit total d'environ 4 To/s.

Une fois les événements construits, ils sont transmis à la première étape de la

sélection, appelée High Level Trigger 1 (HLT1). Cet algorithme de trigger effectue une reconstruction partielle et une sélection préliminaire, réduisant ainsi le débit des données par un facteur 30. Pour répondre aux exigences de performance, HLT1 fonctionne sur le même centre de serveurs que l'Event Builder et utilise des GPU haute performance pour l'accélération. De plus, HLT1 sélectionne des événements spécifiquement dédiés à l'alignement et à l'étalonnage en temps réel. La sortie de HLT1 est stockée dans un tampon de 40 Po, offrant plus d'indépendance vis-à-vis des étapes ultérieures tout en permettant l'exécution des routines d'alignement et d'étalonnage en temps réel.

L'alignement et l'étalonnage en temps réel sont cruciaux pour garantir des performances optimales en physique. Le détecteur est sensible à divers facteurs environnementaux pouvant altérer son positionnement ou sa réponse. Cette procédure est effectuée au début de chaque remplissage du LHC, en utilisant des sélections dédiées de HLT1. Les informations d'alignement et d'étalonnage sont ensuite transmises à la prochaine étape de filtrage, le High Level Trigger 2 (HLT2), éliminant ainsi le besoin de reconstruction hors ligne et améliorant l'efficacité de la sélection des événements.

L'étape finale de sélection est assurée par HLT2, qui s'exécute sur un centre de serveurs dédié et équipé de plus de 256 000 cœurs. HLT2 comprend plus d'un millier de lignes de déclenchement conçues pour sélectionner les événements pertinents à partir des données reconstruites. Grâce aux informations d'alignement et d'étalonnage, la sortie de HLT2 atteint une qualité équivalente à celle de l'analyse hors ligne, permettant aux analyses physiques de commencer quelques jours seulement après l'acquisition des données. HLT2 optimise également l'utilisation du stockage grâce au format de données Turbo Stream. Ce format ne conserve que les candidats sélectionnés durant le filtrage, en supprimant les composants non indispensables des événements, et réduisant ainsi la taille des données d'un ordre de grandeur.

La sortie de HLT2 est stockée sur un tampon disque de 10 Po, garantissant une indépendance vis-à-vis de toute défaillance de stockage permanent. Ce tampon permet à l'expérience de continuer l'acquisition de données pendant jusqu'à six jours sans perte d'événements.

Systèmes futurs pour la physique des hautes énergies Comme mentionné précédemment, les avancées et mises à niveau du complexe d'accélérateurs du LHC provoquent de nouveaux défis expérimentaux, notamment une multiplicité accrue du nombre d'interactions par croisement de faisceaux, des taux de données plus élevés et une exposition plus intense aux radiations. À LHCb, les sous-systèmes des détecteurs intégreront des informations temporelles de haute précision afin d'améliorer les capacités de reconstruction. Par exemple, dans le cas du VeLo, les informations temporelles permettront une meilleure séparation des vertex primaires en raison de leur dispersion dans le temps, améliorant ainsi la reconstruction des traces et la mesure des paramètres corrélés. Cette amélioration nécessite la conception d'une nouvelle puce de lecture capable de gérer des débits de données plus élevés tout en maintenant une tolérance aux radiations. De plus, SciFi et UT seront mis à niveau afin d'améliorer leur sensibilité et leur résistance aux dommages causés par les radiations.

Le système RICH bénéficiera également d'une meilleure granularité et de l'introduction d'informations temporelles, permettant une meilleure suppression des bruits combinatoires dans la reconstruction. Une nouvelle puce sur-mesure, appelée FastRICH, implémentera des convertisseurs temps-numérique de haute précision. En outre, cette puce introduit un nouveau protocole de communication appelé Aurora, conçu pour améliorer l'utilisation des liens seriales et leur fiabilité. Le décodeur de ce protocole a été développé dans le cadre de cette thèse.

Le système calorimétrique subira des améliorations similaires, augmentant sa résistance aux radiations et intégrant des informations temporelles. Ces améliorations nécessiteront le développement de nouveaux ASICs capables de gérer les exigences supplémentaires sur le traitement des données.

Par conséquent, le système d'acquisition devra également être mis à niveau pour s'adapter à l'augmentation du volume de données et à la complexité accrue des événements traités. Le système devra gérer un débit cinq fois plus important par rapport au Run 3, tout en réduisant les données de quatre ordres de grandeur avant le stockage permanent. Pour y parvenir, de nouvelles techniques de sélection sont explorées dans cette thèse afin de réduire le volume des données le plus tôt possible dans la chaîne de traitement. L'utilisation de seuls CPU n'est pas envisageable à ce stade, l'utilisation d'accélérateurs externes tels que les FPGA et les GPU est nécessaire. L'informatique hétérogène avec des GPU a déjà été mise en œuvre avec succès dans le High Level Trigger 1 de LHCb pendant Run 3. Une autre approche prometteuse consiste à effectuer une reconstruction bas niveau à l'aide de FPGA haute performance. Cette thèse étudie l'utilisation de la High-Level Synthesis (HLS), une technique qui traduit des algorithmes de haut niveau en architecture FPGA, permettant potentiellement d'optimiser ces tâches.

L'introduction de nouveaux liens de données à haute vitesse et résistants aux radiations, tels que le lpGBT, nécessitera le développement de cartes de lecture FPGA de nouvelle génération, comme la PCIe400, successeur de l'actuelle PCIe40.

Des exigences de synchronisation plus strictes imposeront des contraintes rigoureuses sur le système de distribution du temps et les FPGA impliqués, nécessitant des performances en gigue afin d'atteindre une précision de l'ordre de $\mathcal{O}(10)$ ps.

L'introduction de nouvelles connections de données à haute vitesse et résistantes aux radiations, tels que le lpGBT, nécessitera le développement de cartes de lecture FPGA de nouvelle génération, comme la PCIe400, successeur de l'actuelle PCIe40. Des exigences de synchronisation plus strictes imposeront des contraintes rigoureuses sur le système de distribution du temps et les FPGA impliqués, nécessitant des performances en gigue afin d'atteindre une précision de l'ordre de O(10) ps.

La réduction des coûts est un autre enjeu crucial, en particulier lorsque le système sera mis à l'échelle pour gérer 30 000 liens optiques. Cette thèse explore des solutions basées sur FPGA permettant de convertir des connections de données sur-mesure en Ethernet standard, facilitant ainsi l'utilisation de composants commerciaux disponibles sur le marché. Cette approche réduit considérablement les coûts tout en maintenant les performances.

D'autres expériences explorent également des avancées technologiques similaires.

Par exemple, ATLAS a développé une nouvelle carte de lecture, le Frontend Link eXchange (FELiX), capable d'ingérer et de traiter un grand nombre de connection de données à haute vitesse. Bien que FELiX utilise toujours PCIe, il prend également en charge l'Ethernet à haut débit comme liaison montante vers la ferme de filtrage des évé- nements. ATLAS explore également des architectures hétérogènes pour son filtre d'événements, intégrant à la fois des GPU et des FPGA. En adoptant cette approche, le système vise à améliorer les performances tout en restant bas-coût et économe en énergie.

Architecture FPGA à LHCb Les cartes de lecture basées sur FPGA, PCIe40, servent d'interface entre l'électronique frontale et le système d'acquisition des données. Ces cartes ont été conçues comme une solution standardisée répondant aux exigences d'acquisition des données, de distribution des signaux d'horloge et de contrôle. Le système de lecture utilise le bus PCIe pour s'interfacer directement avec les serveurs de l'Event Builder, permettant des transferts de données allant jusqu'à 100 Gb/s vers la mémoire hôte via le mécanisme de Direct Memory Access (DMA). Cette approche minimise l'intervention du processeur et en améliore l'efficacité. En outre, PCIe est largement utilisé pour des accélérateurs tels que les GPU, ce qui en fait un standard pertinent sur le marché des serveurs.

Chaque carte de lecture peut gérer jusqu'à 48 liens optiques provenant du détecteur, à condition que le débit total reste dans les limites de la bande passante PCIe. Tous les sous-détecteurs, à l'exception de VeLo, utilisent un protocole résistant aux radiations pour la communication bidirectionnelle entre le frontale et le Back-End. Ce protocole est implémenté par un ASIC sur-mesure conçu par le CERN, appelé GBT, qui prend en charge un débit de 4.8 Gb/s vers le Back-End. De plus, la puce GBT peut communiquer avec des ASICs compagnons tels que le GBT-SCA, permettant le contrôle de l'électronique frontale via des protocoles standards comme I^2C et SPI.

Le lien GBT est également responsable de la transmission du signal d'horloge du LHC et des commandes de contrôle rapide. Ces signaux sont distribués par une architecture de contrôle dédiée qui utilise les mêmes cartes FPGA. Le maître du Trigger et Fast Control (TFC), appelé SODIN, relaie l'horloge du LHC et distribue les commandes synchrones émises par le système de contrôle de l'expérience. Un réseau optique passif interconnecte le maître avec des cartes de contrôle, appelées SOL40, qui transmettent ensuite les signaux de contrôle au frontale via les liens GBT, elles relayent aussi les commandes TFC aux cartes de lecture, TELL40. SODIN et SOL40 nécessitent tous deux un matériel spécialisé pour répondre aux exigences strictes de précision temporelle et de faible gigue.

La fonctionnalité de chaque carte est déterminée par sa configuration de gateware, qui existe en trois versions : SODIN, SOL40 et TELL40. Le gateware de TELL40 est conçu pour répondre aux exigences des sous-détecteurs tout en respectant des principes de modularité et de standardisation. La majorité du gateware repose sur des composants communs, partagés entre tous sous-détecteurs. Ces modules communs gèrent le décodage du protocole GBT, l'alignement des fragments sur différents liens à l'aide de l'identifiant de croisement de faisceau, ainsi que le traitement des informa-

tions de contrôle. Ensuite, des composants spécifiques à chaque détecteur décodent les formats de données non-standard, gèrent la récupération d'erreurs et, éventuellement, effectuent un prétraitement pour réduire le débit des données et optimiser leur formatage pour le logiciel de déclanchement. Chaque détecteur a la flexibilité d'implémenter des algorithmes de traitement spécifiques, à condition qu'ils respectent l'interface d'entrée-sortie spécifiée et tiennent dans les contraintes de ressources du FPGA. Les données traitées sont ensuite encapsulées dans des Multiple Fragment Packets et transférées via PCIe vers l'Event Builder.

L'équipe LHCb Online est responsable du développement, de la maintenance et du support des composants communs du gateware. Cependant, avec moins de dix membres, cette tâche est chronophage et exige une expertise spécialisée. Pour faciliter le développement et faciliter le support, le gateware suit un modèle d'intégration continue (CI). La base de code est structurée en différents dépôts Git, chacun étant inclus comme sous-module dans un dépôt central. Git fournit le contrôle de version, le suivi des problèmes et la gestion des jalons. Une pipeline automatisée dans le dépôt central est déclenché à chaque soumission d'une requête de fusion. Cette pipeline simule le code et compile différentes variantes de gateware, jusqu'à un total de 30. Pour optimiser le temps de compilation, le processus n'est exécuté que lorsque des modifications pertinentes sont détectées. Le gateware compilé est ensuite encapsulé dans des distributions RPM, accompagné du logiciel associé, et déployé sur les serveurs de production.

Cette approche de développement a été couronnée de succès, mais retour d'expérience doivent être prises en compte pour les conceptions futures. Un problème clé réside dans le nombre limité de tests et de simulations effectués sur le gateware. Actuellement, le cadre de simulation ne teste que le flux complet de données à l'aide d'un fichier de stimulus manuel, ne couvrant qu'un seul scenario de test. De plus, l'absence de tests unitaires entraîne une faible couverture des tests et un débogage laborieux, risquant de gaspiller un temps de d'expérimentation précieux. Les simulations nécessitent également d'importantes ressources de calcul, limitant leur fréquence et leur utilisation pendant le développement.

Une autre limitation vient de la dépendance aux logiciels propriétaires pour les simulations, notamment les simulateurs et compilateurs spécifiques aux fournisseurs. Ces outils nécessitent des licences coûteuses, restreignant le nombre d'instances parallèles et réduisant l'efficacité des simulations. En outre, les logiciels propriétaires contraignent les utilisateurs à des chaînes d'outils spécifiques, limitant la portabilité et la flexibilité.

Malgré ces défis, la conception de PCIe40 a démontré la faisabilité d'une carte de lecture basée sur PCIe. Cependant, les tendances récentes du calcul haute performance indiquent un abandon progressif des architectures basées sur PCIe en raison des défis liés à la consommation d'énergie, au refroidissement et aux contraintes d'espace.

La main-d'œuvre est une autre contrainte critique. L'expérience nécessite de nombreuses configurations de gateware. Même avec des composants modulaires, la maintenance de la base de code demande un effort considérable. De plus, les équipes des sous-détecteurs ont des niveaux d'expertise en FPGA variables, nécessitant une

collaboration étroite avec les développeurs principaux.

En vue de la mise à niveau de LHCb, les sous-détecteurs nécessiteront un plus grand nombre de liens fonctionnant à des vitesses plus élevées. Par conséquent, le système de lecture doit être amélioré pour répondre à ces nouvelles exigences. Cela offre une opportunité d'adresser les lacunes de la conception précédente.

Le nouvel ASIC frontale du détecteur RICH adoptera le protocole durci aux radiations lpGBT, qui double le débit disponible. En outre, des exigences temporelles plus strictes imposeront de nouvelles contraintes sur la distribution du signal d'horloge. Une nouvelle carte de lecture est en cours de conception pour tirer parti des avancées technologiques des FPGA. Cette carte est basée sur le dernier FPGA d'Altera, intégrant des capacités de pointe telles que des émetteurs-récepteurs à 112 Gb/s et de la High Bandwidth Memory. La nouvelle carte, appelée PCIe400, permettra un débit quatre fois supérieur à celui de PCIe40. De plus, ces fonctionnalités avancées permettront d'explorer le traitement des données en temps réel et l'accélération de la reconstruction.

Le développement de cette nouvelle carte nécessite une refonte complète du gateware actuel afin d'intégrer le matériel amélioré. Pour optimiser cette transition et garantir la pérennité du design, plusieurs lignes directrices ont été proposées. L'axe principal sera l'adoption d'une approche de développement pilotée par les tests (*Test-Driven Development*). Dans cette méthodologie, les développeurs conçoivent d'abord la structure de test avant d'implémenter les composants, garantissant ainsi le respect des spécifications et des exigences. Un autre principe clé est l'utilisation d'outils et de composants open-source afin d'améliorer l'efficacité des simulations et de faciliter la portabilité sur différentes plateformes. Dans le cadre de cette thèse, divers outils et frameworks de simulation ont été évalués, et une bibliothèque commune de composants indépendants des fournisseurs a été développée.

Outils modernes de développement FPGA Une partie essentielle du développement de gateware est le processus rigoureux de vérification, nécessaire pour garantir que la conception respecte les spécifications. Ce processus consiste à tester tous les cas normaux et limites que le composant pourrait rencontrer. Des études indiquent que les tâches de vérification représentent souvent plus de 50% du temps de développement afin d'obtenir des résultats satisfaisants.

De nombreuses techniques de vérification ont été initialement développées pour la conception d'ASIC, où une seule erreur dans une puce fabriquée peut entraîner des pertes de plusieurs millions de dollars. Récemment, ces mêmes techniques ont été adoptées pour le développement FPGA, car l'augmentation de la taille et de la complexité des conceptions exige une vérification plus rigoureuse.

La méthodologie traditionnelle de vérification repose sur la simulation fonctionnelle de la conception. Les simulations Register Transfer Level (RTL) sont utilisées pour valider fonctionnellement les composants écrits dans des langages de description matérielle (HDL) comme VHDL. Ces simulations fournissent un suivi précis du comportement cycle par cycle et exécutent directement la même conception destinée à la synthèse matérielle. Cependant, les simulations RTL ne tiennent pas compte des

délais de propagation des signaux ni d'autres effets matériels, mais ces informations ne sont généralement pas cruciales pour la vérification fonctionnelle.

Un *testbench* est composé d'un dispositif sous test (DUT), d'un générateur de stimuli et d'un vérificateur. Pour chaque simulation, le concepteur doit évaluer la couverture des tests et proposer de nouveaux scénarios si nécessaire. Cependant, le nombre de scénarios possibles croît de manière exponentielle avec la complexité du DUT, ce qui rend difficile l'obtention d'une couverture suffisante. Une façon de gérer cette complexité est de se concentrer sur la vérification des plus petites unités fonctionnelles avant de les intégrer dans des composants plus grands.

Pour quantifier la qualité des tests, une métrique appelée *couverture* est utilisée. La couverture fonctionnelle garantit que la vérification est alignée sur les spécifications de conception. Cette métrique, définie par le concepteur, mesure si des scénarios spécifiques ont été observés, validés et testés.

Cependant, l'implémentation manuelle d'un nombre suffisant de cas de test est une tâche fastidieuse. Les cadres de vérification modernes proposent diverses techniques pour rationaliser ce processus. L'une des plus efficaces est le Constrained Random Testing. Cette méthode définit les cas de test à l'aide d'un ensemble de contraintes appliquées à un générateur de stimuli pseudo-aléatoire. Un vérificateur évalue les niveaux de couverture et fournit un retour d'information au générateur, garantissant que toutes les propriétés pertinentes sont testées. Cette méthodologie permet de générer un grand nombre de cas de test avec un effort minimal, car seules quelques lignes de code sont nécessaires pour décrire les contraintes.

Une autre approche qui simplifie la création des bancs de test est le Transaction-Level Modeling (TLM). La plupart des composants FPGA communiquent via des protocoles et interfaces bien définis. En rendant abstraites ces interfaces dans des modèles communs, les concepteurs peuvent se concentrer sur l'implémentation du contenu des transactions plutôt que sur la gestion des signaux bas niveau. Les cadres TLM classent généralement les interfaces en deux catégories principales : les *streaming interfaces*, où les données circulent en continu à travers le composant, formant un pipeline de traitement. Les *memory-mapped interfaces*, où les transactions reposent sur un protocole de bus adressé, similaire aux opérations mémoire traditionnelles.

Ces techniques innovantes ont été intégrées dans divers cadres de vérification open-source, permettant aux développeurs de les réutiliser dans plusieurs projets. Pour les conceptions basées sur VHDL, trois principaux cadres ont été évalués, chacun offrant des fonctionnalités différentes qui peuvent être combinées pour obtenir des résultats optimaux en matière de vérification.

UVVM est un cadre largement adopté dans l'industrie et le milieu académique. Il fournit des utilitaires de vérification de base, tels que des vérificateurs et des assertions, mais sa principale caractéristique est l'utilisation de composants de vérification (VVCs). Ceux-ci étendent la modélisation au niveau transactionnel en automatisant la planification et l'exécution des transactions. Un séquenceur central orchestre les bancs de test, améliorant ainsi la modularité. UVVM comprend également des VVCs intégrés pour de nombreux protocoles courants, et il est facile d'étendre ces derniers pour des applications personnalisées.

OSVVM, soutenu par le groupe de travail IEEE VHDL, offre des fonctionnalités de base similaires à UVVM mais se spécialise dans le constrained random testing et la couverture fonctionnelle. Il inclut de puissants générateurs pseudo-aléatoires, des histogrammes et des routines de couverture qui s'intègrent facilement aux bancs de test, améliorant ainsi considérablement la qualité des tests.

VUnit se distingue des deux autres par l'introduction d'une interface de script basée sur Python. Cette abstraction simplifie la gestion des fichiers, l'interfaçage avec les simulateurs et la compilation, améliorant ainsi l'efficacité du flux de vérification.

L'utilisation de ces cadres améliore significativement la qualité des tests et fournit une mesure quantitative de la couverture des test. Cependant, la vérification basée uniquement sur la simulation ne peut couvrir tous les scénarios possibles, rendant l'obtention d'une couverture complète impossible.

Vérification formelle Une méthodologie différente, basée sur la preuve mathématique, offre des solutions à certains défis de la vérification FPGA. La vérification formelle (FV) traduit les conceptions et leurs spécifications en formules logiques qui doivent être satisfaites. Ces problèmes sont mathématiquement connus sous le nom de problèmes de satisfiabilité booléenne. Cependant, les résoudre pour tous les états possibles est un problème NP-difficile, ce qui signifie que la complexité computationnelle croît de manière exponentielle avec la taille des entrées.

Pour gérer cette complexité, diverses optimisations sont appliquées afin de limiter la taille du problème. La technique la plus utilisée est le Bounded Model Checking, qui exprime le problème sous la forme d'un système à états finis. L'outil FV traduit la conception en une formule logique et l'analyse sur un nombre limité d'étapes. Bien que cela ne fournisse pas une preuve absolue de correction, cette approche est souvent suffisante pour vérifier le comportement fonctionnel tout en maintenant des un cout de calcul raisonnable.

Les outils de FV nécessitent que le concepteur définisse des propriétés à l'aide de langages de spécification de propriétés (PSL). Ces langages décrivent avec précision le comportement et les objectifs de vérification à travers des assertions et des hypothèses. Les assertions définissent les états valides du DUT, tandis que les hypothèses contraignent ses entrées pour garantir une vérification pertinente.

Un banc de test basé sur la vérification formelle peut réduire considérablement le temps et l'effort nécessaires au développement des cas de test. Cependant, les limitations actuelles des outils empêchent leur utilisation sur des conceptions avec plusieurs domaines d'horloge ou des chemins de données complexes. Par conséquent, la FV est surtout utilisée comme complément à la vérification traditionnelle par simulation.

Pour évaluer l'efficacité des outils de vérification formelle open-source, un problème connu dans le gateware TELL40 du RICH a été analysé. Le problème provenait du bloc de traitement des données, mais les tests traditionnels ne pouvaient être implémentés en pratique en raison de la présence de plusieurs domaines d'horloge et de composants propriétaires. À la place, la FV a été appliquée à un sous-composant chargé de la compression des données entrantes et de la génération de paquets fragmentés.

En modélisant les spécifications des interfaces de flux en entrée et en sortie et en définissant des propriétés en PSL, l'outil a réussi à identifier le problème. L'ensemble du processus a pris quelques jours, contre plusieurs semaines avec une simulation traditionnelle.

Bibliothèque de composants commune Les concepteurs s'appuient souvent sur des composants propriétaires, nécessitant des outils et licences coûteux, ou sur des solutions artisanales qui manquent de robustesse. Cela entraîne des bases de code fragmentées, difficiles à maintenir et à tester. Pour répondre à ce défi, une bibliothèque de composants commune, *colibri*, a été développée. Cette bibliothèque open-source permet aux développeurs de contribuer et de modifier des composants tout en garantissant une portabilité indépendante du fournisseur sur différentes plateformes FPGA. Chaque composant inclut une suite de tests complète, intégrant à la fois la vérification fonctionnelle et la vérification formelle, exécutée dans des pipelines d'intégration continue pour fournir des résultats de vérification à jour.

L'un des développements clés de la bibliothèque est un décodeur pour le protocole Aurora utilisé par l'ASIC FastRICH. Ce protocole comprend l'équilibrage de charge DC, la récupération d'horloge et l'agrégation multi-liens. Une stratégie de vérification rigoureuse a été appliquée, incluant des bancs de test pour les sous-composants individuels ainsi que pour le décodeur complet. Les trois cadres de vérification – UVVM, OSVVM et VUnit – ont été utilisés dans ce processus.

Le décodeur a été entièrement implémenté en VHDL-2008, permettant une portabilité fluide entre différentes plateformes FPGA telles que PCIe40 d'Altera et les kits de développement d'AMD Xilinx sans nécessiter de modifications du code source. Les mesures de performance ont permis d'analyser l'utilisation des ressources, aidant l'équipe RICH à optimiser leur conception.

La portabilité a été validée en migrant la conception d'un kit de développement AMD Xilinx vers un FPGA Microsemi. Les mesures de performance ont confirmé la transmission réussie de paquets de 4 ko sans congestion à 10 Gb/s. De plus, la faible utilisation des ressources permet d'intégrer à l'avenir des fonctions de traitement des données directement dans le FPGA.

Synthèse de haut niveau La *synthèse de haut niveau* (*High-Level Synthesis* – HLS) permet le développement FPGA en utilisant des langages de haut niveau tels que C++, réduisant ainsi la barrière à l'entrée tout en offrant de puissants outils de débogage logiciel.

Une application prometteuse de la HLS est l'accélération des charges de calcul. Cette approche a été évaluée en portant une partie de l'algorithme de décodage et de clustering VeLo depuis une implémentation GPU utilisée dans le High-Level Trigger 1 vers une conception accélérée sur FPGA. Lae transfert initial vers FPGA a été simple et a produit des résultats fonctionnels en quelques semaines. Cependant, les performances étaient initialement bien en deçà des attentes.

L'optimisation de l'architecture de l'algorithme s'est révélée extrêmement complexe,

nécessitant une refonte complète de la disposition mémoire, des transactions et des formats de données. Le système était initialement limité par les transferts PCIe en raison de conflits de bus générés par le compilateur. Une fois l'accès mémoire optimisé, la limitation principale est devenue la performance de calcul, nécessitant une parallélisation accrue. La solution proposée utilise plusieurs pipelines de calcul en parallèle, exploitant pleinement les ressources du FPGA. Cela a nécessité des mécanismes de synchronisation rigoureux pour éviter toute corruption des données.

Les tests de performance du design final ont montré des résultats prometteurs, atteignant les objectifs fixés. Cependant, une limitation critique demeure : un bogue non résolu du compilateur entraîne un kernel panic lorsque le FPGA tente d'écrire en mémoire hôte, soulignant le manque de maturité technologique des outils HLS actuels pour les applications de traitement des données.

Conclusion Cette étude a évalué à la fois l'état actuel et le potentiel d'utilisation des cadres de vérification open-source ainsi que des outils de vérification formelle. L'adoption de ces méthodologies a significativement amélioré la qualité du code et la couverture des tests. Parmi les cadres de vérification, VUnit s'est imposé comme le meilleur choix en raison de sa flexibilité et de sa portabilité sur différents simulateurs.

Bien que moins matures, les outils de vérification formelle se sont montrés d'une grande valeur dans les efforts de vérification. Leur application à des composants critiques a permis d'identifier des bogues rares en un temps réduit, prouvant leur efficacité. Cependant, une expertise en PSL et en spécifications formelles est nécessaire pour les utiliser efficacement.

Le développement de la bibliothèque commune *colibri* a fourni un terrain d'expérimentation pour ces méthodologies de vérification. Les composants standardisés ont réduit le temps de développement et assuré la fiabilité grâce à des tests rigoureux. L'indépendance vis-à-vis des fournisseurs a été validée par la portabilité des conceptions sur plusieurs plateformes, et la bibliothèque est désormais largement adoptée par les développeurs des principales expériences du CERN.

Enfin, bien que la HLS soit un outil puissant, l'optimisation des performances reste un défi nécessitant une expertise approfondie en architecture matérielle. Ces méthodologies et résultats seront maintenant adoptés par l'équipe centrale de LHCb pour les futurs développements FPGA.

Acknowledgements

I would like to thank my father and my mother for supporting me during my studies and my life abroad. In particular, I would like to dedicate this thesis to my dear grandfather Attilio, who taught me how important is curiosity and creativity in enjoying life.

I have made many good friends along this journey with whom I shared many adventures. All of you made it easier in the hard moments and fun in the happy ones.

I hope to share my days ahead with all of you, even when distance makes it difficult. Thank you for everything!

Mi vorìa ringrassié me pare e mia mare për avèj-me sustenù durant ij mei studi e la vita da l'autra part dle montagne. An manera particular, mi vorìa dedichè sta tesi a mè car nòno Attilio, ch'a m'ha mostrà com important ch'a sia la curiosità e la creatività për gòd-se la vita.

Durant sto viage, mi son fame tanti bon amis con coj ch'i l'hai spartì tante aventure. Vojàutri tuti l'eve rendù pì fàcil ant ij moment dur e pì divertent an cheuj bon.

Spero ëd podè passé ij mè giorn futur con vojàutri tuti, anche quand la lontanansa a lo rend-rà pi difícil. Grazie ëd tut!

Contents

List of Figures

List of Tables

List of Acronyms

100GbE

100 Gigabit Ethernet. 121, 125, 126

10GbE

10 Gigabit Ethernet. 24, 118, 121, 123, 124

1GbE

1 Gigabit Ethernet. 121, 122

ASIC

Application-Specific Integrated Circuit. 56, 63, 64, 72, 73, 85, 112

BE

Back-End. 24, 109, 112, 113, 117, 118

BMC

Bounded Model Checking. 95–97

BU

Builder Unit. 22, 58

BXID

Bunch Crossing ID. 75, 77, 109

CERN

European Organization for Nuclear Research. 21, 22, 35, 36, 38, 51

COTS

Commercial Off-The-Shelf. 64, 65, 67, 121, 122

CRC

Cyclic Redundancy Checking. 121

DAQ

Data Acquisition. 22, 23, 26, 54, 56–58, 63–75, 99, 118, 125, 126, 137

DMA

Direct Memory Access. 57, 71, 77, 129

DPDK

Data Plane Development Kit. 126

DUT

Device Under Test. 23, 87, 88, 90, 96, 97, 101, 102, 109, 110, 117

EB

Event Builder. 22, 26, 58, 59, 71, 73, 77, 81, 126

ECAL

Electromagnetic CALorimenter. 21, 22, 42, 51, 52, 61

ECS

Experiment Control System. 57, 75, 77

EDA

Electronic Design Automation. 94

EOP

End of Packet. 109, 110

FE

Front-End. 24, 54, 69, 75, 77, 82, 118, 122, 124, 125, 137

FEE

Front-End Electronic. 55–57, 64, 65, 72, 73, 78, 81, 112, 118, 122

FELiX

Front-End Link Exchange. 67, 69

FF

Flip-Flops. 24, 114, 115

FIFO

First-In First-Out. 23, 90, 91, 96, 97, 100, 115, 116, 121, 122, 129

FPGA

Field-Programmable Gate Array. 22–24, 56, 57, 64, 65, 67, 69, 71, 72, 75, 78, 81, 82, 85, 86, 91–95, 97, 98, 100, 103–108, 112, 118, 121, 122, 126, 128, 129, 131

FSM

Finite State Machine. 110, 115, 116, 122, 129

FV

Formal Verification. 24, 94–97, 101, 103, 106, 109, 111, 135, 136

GBT

GigaBit Transceiver. 26, 56–58, 72, 73, 75, 77, 117

GPU

Graphical Processing Unit. 55, 59, 64, 67, 69, 71, 72, 81, 126, 128, 129, 131, 133

HCAL

Hadronic CALorimenter. 21, 22, 42, 51, 52

HDL

Hardware Description Language. 23, 86, 87, 93, 94, 98

HEP

High Energy Physics. 22, 63, 65, 69, 118, 126

HL-LHC

High Luminosity LHC. 63, 67

HLS

High-Level Synthesis. 64, 86, 104, 106, 108, 109, 126, 137

HLT

High Level Trigger. 78, 109, 122, 126, 127

HLT1

High Level Trigger 1. 54, 55, 58–60, 62, 126

HLT2

High Level Trigger 2. 54, 55, 59–62

IP

Intellectual Property. 24, 75, 83, 86, 98, 106, 108, 112, 113, 115, 117, 118, 121, 122

IPv4

Internet Protocol version 4. 121, 122

L0

Level-0 Trigger. 22, 54, 55

LFSR

Linear Feedback Shift Register. 24, 114, 115

LHC

Large Hadron Collider. 21, 35–39, 43

LLI

Low Level Interface. 75

lpGBT

Low Power GigaBit Transceiver. 24, 26, 64, 65, 69, 82, 112, 115, 117, 118, 120, 121

LS

Long Shutdown. 37, 82, 112

LSB

Least Significant Bit. 127

LSU

Load-Store Unit. 24, 129, 130

LUT

Look-Up Table. 24, 118, 119

MAC

Media Access Control. 122

MEP

Multi Event Packet. 58

MFP

Multi Fragment Packet. 57, 58, 77, 79

MQTT

Message Queue Telemetry Transport. 122

NIC

Network Interface Card. 121

OSVVM

Open Source VHDL Verification Methodology. 94, 101, 106, 135

PCIe

Peripheral Component Interconnect Express. 22, 57, 64, 65, 67, 69–72, 75, 77, 81, 82, 98, 121, 129, 133

PCS

Physical Coding Sublayer. 112, 115, 116

PLL

Phase-Locked Loop. 75

PMA

Physical Medium Attachment. 112

PoC

Proof of Concept. 24, 65, 118, 121–123, 125, 129, 133

PON

Passive Optical Network. 73, 75, 77

PS

Proton Synchrotron. 36

PSB

Proton Synchrotron Booster. 36

PSL

Property Specification Language. 96, 97, 103, 109–111, 136

PV

Primary Vertex. 21, 43, 54

QCD

Quantum Chromo-Dynamics. 37–39

RICH

Ring Imaging Cherenkov. 46, 50, 51, 60, 61, 64, 82, 109, 112

RTA

Real-Time Alignment and Calibration. 60

RTL

Register Transfer Level. 78, 86, 104

RU

Readout Unit. 22, 58

SAT

Boolean Satisfiability Problem. 95

SciFi

Scintillating Fiber Tracker. 21, 42, 43, 45, 47, 50, 63, 73, 82

SM

Standard Model. 38, 39

SoM

System-on-Module. 125, 126

SOP

Start of Packet. 109, 110

SP

Super Pixel. 24, 127, 128

SPS

Super Proton Synchrotron. 36

TFC

Timing and Fast Control. 23, 57, 71, 73–75, 77, 109, 110

TLM

Transaction Level Modeling. 91, 93, 101, 135

UDP

User Datagram Protocol. 24, 121, 122, 124, 126

UT

Upstream Tracker. 21, 26, 42, 43, 45, 46, 50, 63, 73, 82

UVVM

Universal VHDL Verification Methodology. 93, 94, 101, 106, 117, 135

VeLo

Vertex Locator. 21, 41–46, 50, 54, 60, 63, 72, 73, 82, 125–127

VL

Versatile Link. 72

VVC

VHDL Verification Component. 23, 24, 93, 101, 102, 116, 117, 135

Introduction

High Energy Physics experiments, such as LHCb at CERN, produce enormous data volumes that require advanced FPGA-based data acquisition systems. With ongoing upgrades increasing data rates, improving FPGA development methodologies is essential to ensure efficient and reliable performance.

This thesis explores modern approaches to FPGA gateware design, emphasizing verification-driven development and open-source solutions. By evaluating frameworks like UVVM, OSVVM, and VUnit, this work demonstrates how automated functional verification enhances test coverage and reduces debugging time. Formal verification techniques are also applied to critical components, revealing their potential for early error detection.

A key contribution of this research is the development of colibri, an open-source library of FPGA components that improve portability and streamline integration across platforms. The library has been successfully implemented in high-speed Ethernet-based readout systems for detector data acquisition. Additionally, the feasibility of High-Level Synthesis (HLS) is explored as an alternative to traditional FPGA design, allowing C++-based development for trigger and reconstruction algorithms. While HLS shows promise, current tools still present challenges in meeting performance and reliability requirements.

The thesis begins with an introduction to the LHCb experiment, detailing its physics objectives and detector design. It then examines the architecture of LHCb's FPGA-based data acquisition system, followed by a discussion of future DAQ system upgrades and trends in High Energy Physics. The core methodologies for FPGA development, including verification strategies and common libraries, are presented before case studies illustrate their practical applications. The work concludes with a summary of key achievements and their implications for next-generation FPGA-based systems in High Energy Physics.

1 The LHCb Experiment at CERN

This chapter introduces the CERN laboratory, its accelerator complex, and the LHCb experiment, where the research for this thesis was conducted. Section 1.1 provides a brief overview of the CERN and its state-of-the-art accelerator complex, the LHC. Section 1.2 outlines the physics goals and timeline of the LHCb experiment. Finally, Section 1.3 describes the experimental aspects of the current LHCb detector.

1.1 European Organization for Nuclear Research

CERN is an international research organization founded in 1954. The organization's headquarters are located near Geneva on the border between France and Switzerland. At the time of writing, the member states participating in the organization are 24.

The main purpose of CERN is to investigate the fields of Nuclear and Subnuclear Physics, focusing on High Energy Physics, for civil applications. The CERN laboratory is the largest particle physics laboratory in the world and has given the world numerous scientific achievements such as:

- **1973**: Discovery of Neutral Currents in the Gargamelle bubble chamber [39]
- **1983**: Discovery of W and Z bosons in the UA1 and UA2 experiments [10]
- **1989**: Determination of the number of light neutrino families in the ALEPH, DELPHI, L3, and OPAL experiments [27]
- **1995**: First creation of anti-hydrogen with the PS210 experiment [9]
- **1999**: Discovery of direct CP violation in the NA48 experiment [14]
- **2012**: Discovery of the Higgs boson by the ATLAS and CMS experiments [2], [23]
- **2015**: Discovery of pentaquarks by the LHCb experiment [50]
- **2019**: Discovery of CP violation in charm hadrons by the LHCb experiment [52]

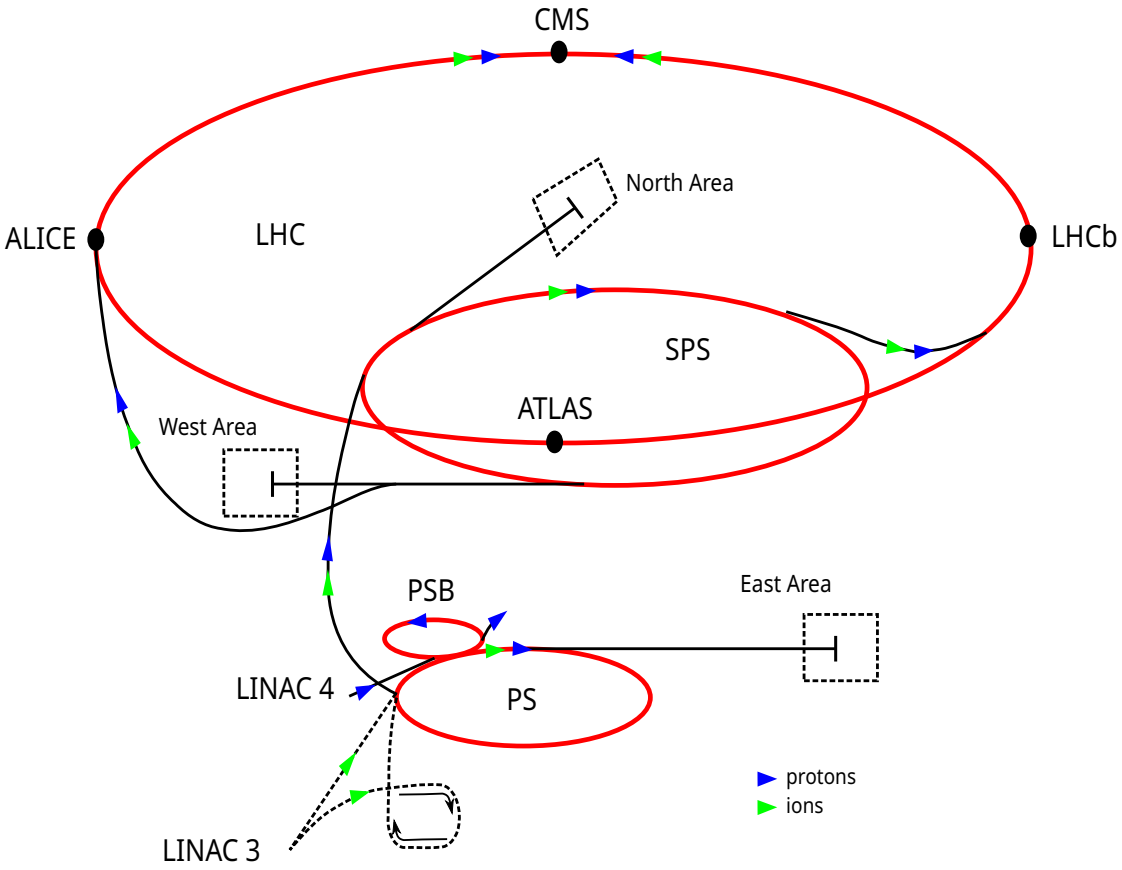


Figure 1.1: Schematic view of the CERN Accelerator Complex.

The Large Hadron Collider The LHC [18] is a particle accelerator located 100 meters underground within a 27-kilometer circular tunnel, and it is designed to collide hadrons. In this collider, proton beams are accelerated to a design energy of 7 TeV each, resulting in collisions with a center-of-mass energy ($\sqrt{s}$) of 14 TeV. The LHC operates at a peak luminosity of 10^{34} cm^{-2} s^{-1} for proton-proton collisions. In addition to protons, the LHC can also accelerate and collide heavy ions to study quark-gluon plasma and other exotic states of matter. The LHC is built using superconducting elements in order to achieve the specifications listed above.

The LHC is the final stage of a complex accelerator chain where protons are progressively accelerated to achieve high energies, as shown in Figure 1.1. The process begins with hydrogen ions (H^-), which are injected into the LINAC4 linear accelerator. Here, the H^- ions are accelerated to an energy of 160 MeV, after which their electrons are stripped, leaving protons. These protons are then injected into the Proton Synchrotron Booster (PSB), a system of four stacked synchrotron rings, which increases their energy to 1.4 GeV.

Following the PSB, the protons enter the Proton Synchrotron (PS), a 628-meter circular accelerator that further boosts their energy to 25 GeV. At this stage, the protons are moving at relativistic speeds. They are then injected into the Super Proton Synchrotron (SPS), where they are accelerated to 450 GeV before finally being transferred

to the LHC.

In the LHC, protons must circulate in opposite directions to enable collisions, so they are injected into two separate beam lines. The full accelerator complex is illustrated in Figure 1.1.

The LHC beams are filled with 2,808 bunches of 10^{11} protons each [31]. These bunches are maintained along their trajectory by 1232 superconducting dipole magnets, which produce a magnetic field of up to 8.34 T. The proton beams pass through the four interaction points around the ring, where the experiments have been built, at a frequency of 40 MHz. The four main experiments are:

- **A Toroidal LHC ApparatuS (ATLAS)**: A general-purpose detector designed to search for the Higgs boson and supersymmetric particles.
- **Compact Muon Solenoid (CMS)**: Another general-purpose detector, constructed with a different design but aiming for the same physics goals as ATLAS, thus providing a double-blind validation of results.
- **A Large Ion Collider Experiment (ALICE)**: This detector is optimized for heavy-ion physics, specifically for lead-lead (Pb-Pb) collisions, to investigate quark-gluon plasma. ALICE focuses on the properties of strongly interacting matter and explores Quantum Chromo-Dynamics (QCD).
- **Large Hadron Collider beauty (LHCb)**: Specialized in heavy quark flavor physics, LHCb aims to determine CP violation parameters and investigate rare decays of b and c hadrons. Further details of the LHCb physics program are presented in the next section.

1.2 Physics at LHCb

The data acquisition periods at the LHC are organized into distinct intervals called *Runs*. Each Run typically spans several years and includes approximately nine months of data collection with circulating beams. This period is further divided into about one month dedicated to heavy-ion collisions, with the remaining time focused on proton collisions. After each data-taking phase, a three-month technical stop is scheduled for maintenance and minor upgrades.

Runs are separated by extended maintenance intervals known as *Long Shutdown (LS)*, during which major upgrades and extensive maintenance are performed on both the accelerator complex and the experiments. The overall LHC time schedule is illustrated in Figure 1.2.

Runs 1 and 2 (2010-2018) The LHCb physics program for Runs 1 and 2 focused on exploring the flavour sector and searching for indirect signs of New Physics through precision measurements and rare decay processes. The research topics included:

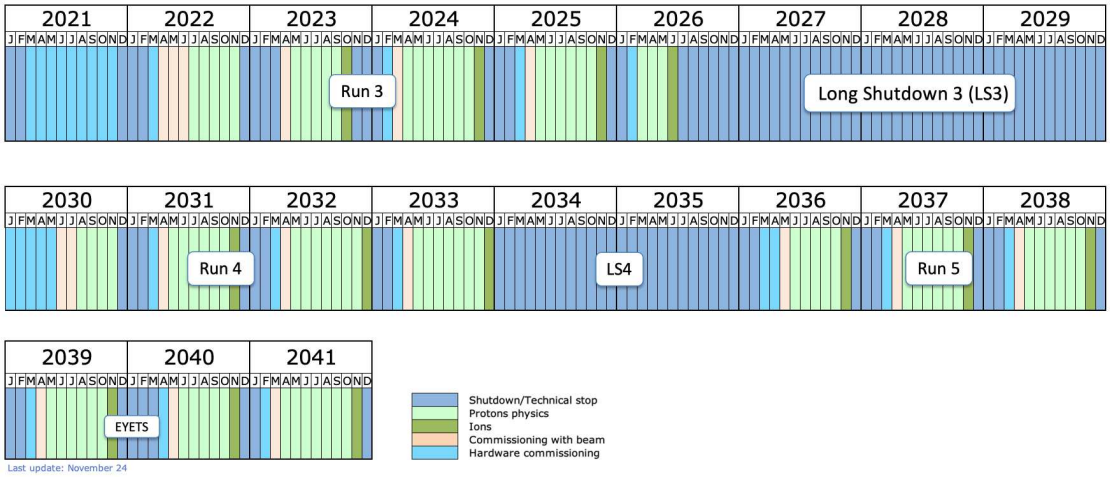


Figure 1.2: Long term time schedule of the LHC. Courtesy of CERN.

- **CP Violation in B-mesons**: LHCb measured CP-violating asymmetries in B^0 and B_s^0 decays and precisely determined the CKM angle γ, providing insights into matter-antimatter asymmetry.
- **Rare Decays & New Physics**: Rare flavour-changing neutral current (FCNC) decays, like $B_s^0 \to \mu^+\mu^-$, were studied to search for deviations from Standard Model (SM) predictions. LHCb also tested lepton flavour universality in decays like $B \to K\ell^+\ell^-$, with potential hints of new particles.
- **Hadron Properties**: The experiment precisely measured B and D meson properties, such as lifetimes and mixing parameters, testing the SM's flavour sector with stringent constraints.
- **Charm Sector CP Violation**: Any significant CP violation in charm quark decays could hint at New Physics, given the SM predicts very small effects here.
- **Forward Region Physics and QCD**: LHCb's forward acceptance allowed studies in heavy-ion collisions, hadronization, and parton distributions, enriching QCD and heavy-ion physics knowledge.
- **Exotic Hadrons**: LHCb discovered more than 20 exotic hadrons (tetraquarks and pentaquarks), offering new insights into the strong force.

Overall, LHCb collected an integrated luminosity of 9.56 fb^{-1}. LHCb's program for Runs 1 and 2 delivered high precision in flavour physics, setting strong limits on New Physics while complementing direct searches from ATLAS and CMS.

Runs 3 and 4 (2022-2033) The current detector improves the sensitivity in flavour physics, in order to investigate possible New Physics in the flavour sector and make precision test of the SM.

LHCb aims to collect 50 fb^{-1}, five times the luminosity achieved in previous runs, which will enable significantly larger samples of B and D final states than those gathered by current B-factories. Such increases will support precision SM tests, potentially giving hints of the existence of TeV-scale particles that complement the direct searches conducted by ATLAS and CMS.

CP violation measurements are the focus of the current physics program due to their strong experimental and theoretical constraints. Higher luminosity will notably benefit studies of CP violation in B_s^0 decays, and measurements of branching ratios in rare decays involving flavour-changing neutral currents could offer insights into new heavy particles.

Investigations will also extend to the lepton sector and other topics beyond flavour physics. LHCb's forward region coverage enables measurements that probe a different regime of proton parton density functions compared to those in ATLAS and CMS, which may complement studies in areas ranging from the effective weak mixing angle for leptons to the W-boson mass. QCD studies will also benefit from the forward coverage, extending the results from the central region.

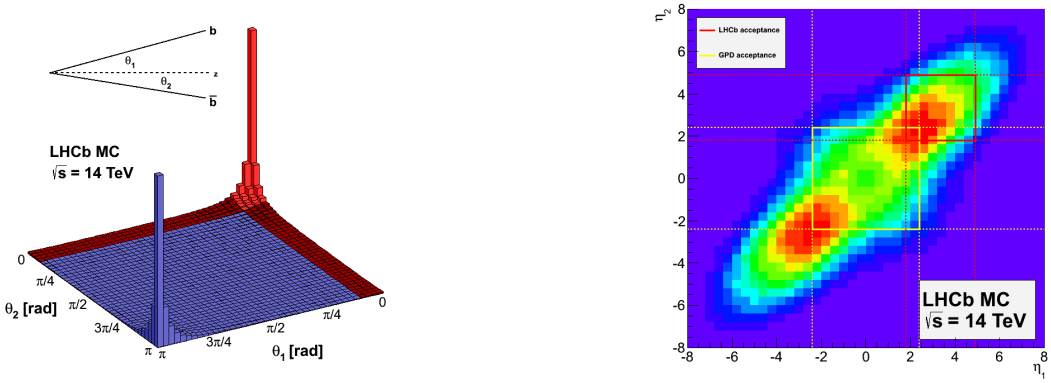
Run 5 and beyond (2036-2041) The LHC will undergo an upgrade during Long Shutdown 3 (2027-2029) to reach its High Luminosity phase, allowing it to deliver a luminosity of 1.5×10^{34} cm^{-2} s^{-1} and accumulate up to 300 fb^{-1} at LHCb's interaction point [46]. To fully exploit these increased collision rates, LHCb will also undergo a major upgrade, known as *LHCb Upgrade II*. This upgrade will significantly expand LHCb's data collection, surpassing other planned experiments and effectively doubling the accessible energy scale for precision flavour physics. The resulting dataset will improve sensitivity to potential New Physics scenarios and enhance the precision of numerous key observables.

The mean number of interactions per proton-proton bunch-crossing, μ, is expected to reach around 40 at the start of each fill, a 8 times increase from the Run 3 conditions. This increase in particle multiplicity and interaction rates poses significant challenges for accurate identification of secondary vertices. All detectors will face considerable difficulties due to these elevated rates and the enhanced radiation damage affecting certain components. Furthermore, the trigger and data acquisition systems will require major upgrades to handle these conditions, which are discussed in detail in the following chapters.

1.3 The LHCb Experiment

The LHCb experiment is designed to study flavor physics, focusing on heavy quarks like beauty (b) and charm (c). This is possible due to the high production cross-sections of $b\bar{b}$ [51] and $c\bar{c}$ [4] pairs in proton-proton (pp) collisions:

$$\begin{aligned} \sigma(pp \to b\bar{b}X) &= (154.3 \pm 1.5 \pm 14.3)\ \mu b \qquad (\sqrt{s} = 14\ \text{TeV}) \\ \sigma(pp \to c\bar{c}X) &= (2369 \pm 3 \pm 152 \pm 118)\ \mu b \quad (\sqrt{s} = 14\ \text{TeV}) \end{aligned} \tag{1.1}$$



(a) $b\bar{b}$ production angle

(b) $b\bar{b}$ production η

Figure 1.3: Production characteristics of $b\bar{b}$ pairs: angles with respect to the beam direction (a) and pseudo-rapidity (b). The data comes from fully simulated events from pp collisions at $\sqrt{s} = 14$ TeV. LHCb acceptance region is highlighted in red. [30]

Because of the asymmetry in the parton momentum distribution during pp collisions, b and c quarks are produced with a significant boost in the beam direction as shown in Figure 1.3. For this reason, the LHCb detector is built as a forward spectrometer.

The LHCb detector is divided along the vertical plane that passes through the beam line, with the two sides referred to as side A (Airport) and side C (Centre). The geometrical acceptance of LHCb is [10, 300] mrad in the horizontal plane and [10, 250] mrad in the vertical plane. The trajectories of charged particles are curved in the horizontal plane by a dipole magnet. Therefore, the LHCb detector can detect particles with a pseudorapidity η between 1.8 and 4.9, where η is defined as:

$$\eta = -ln\left[tan\left(\frac{\theta}{2}\right)\right] = \frac{1}{2}ln\frac{|\vec{p}| + p_L}{|\vec{p}| - p_L} \tag{1.2}$$

in which θ is the angle between the particle and the beam axis and p_L is the longitudinal momentum.

During Runs 1 and 2, the LHCb experiment recorded over 9 fb^{-1} of integrated luminosity, as shown in Figure 1.4. In the current Run 3, at the time of writing, the LHCb experiment has already surpassed the total integrated luminosity recorded during the previous two runs combined.

The LHCb detector layout is described in the next sections and a complete side view is shown in Figure 1.5.

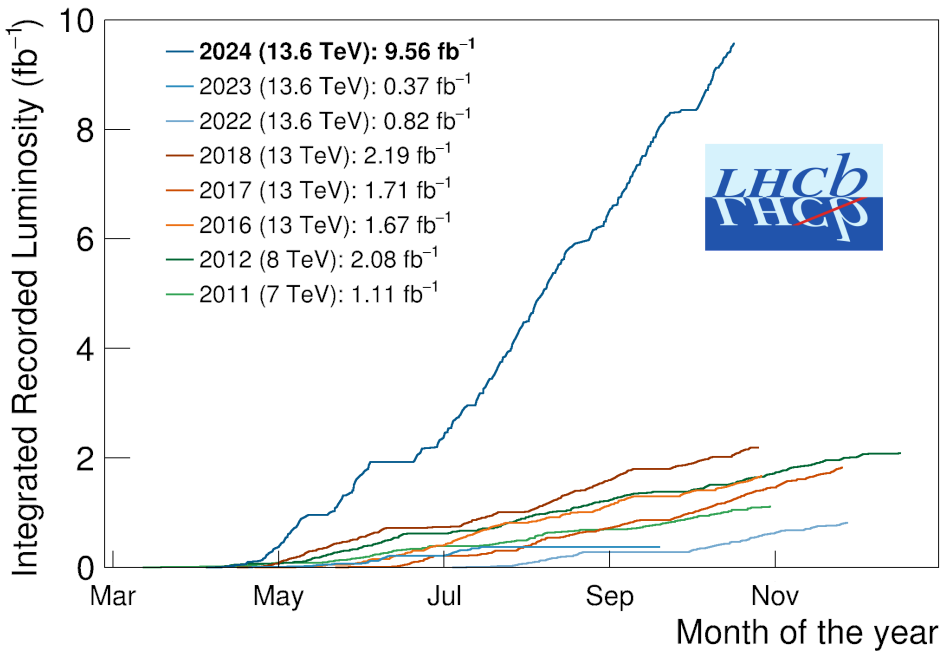


Figure 1.4: Integrated recorded luminosity at the LHCb experiment during Run 1 (2010-2012), Run 2 (2015-2018), and Run 3 (2022-2026). The first two years of Run 3 (2022, 2023) recorded lower luminosity because of delayed commissioning and issues with the VeLo detector. Courtesy of the LHCb Collaboration.

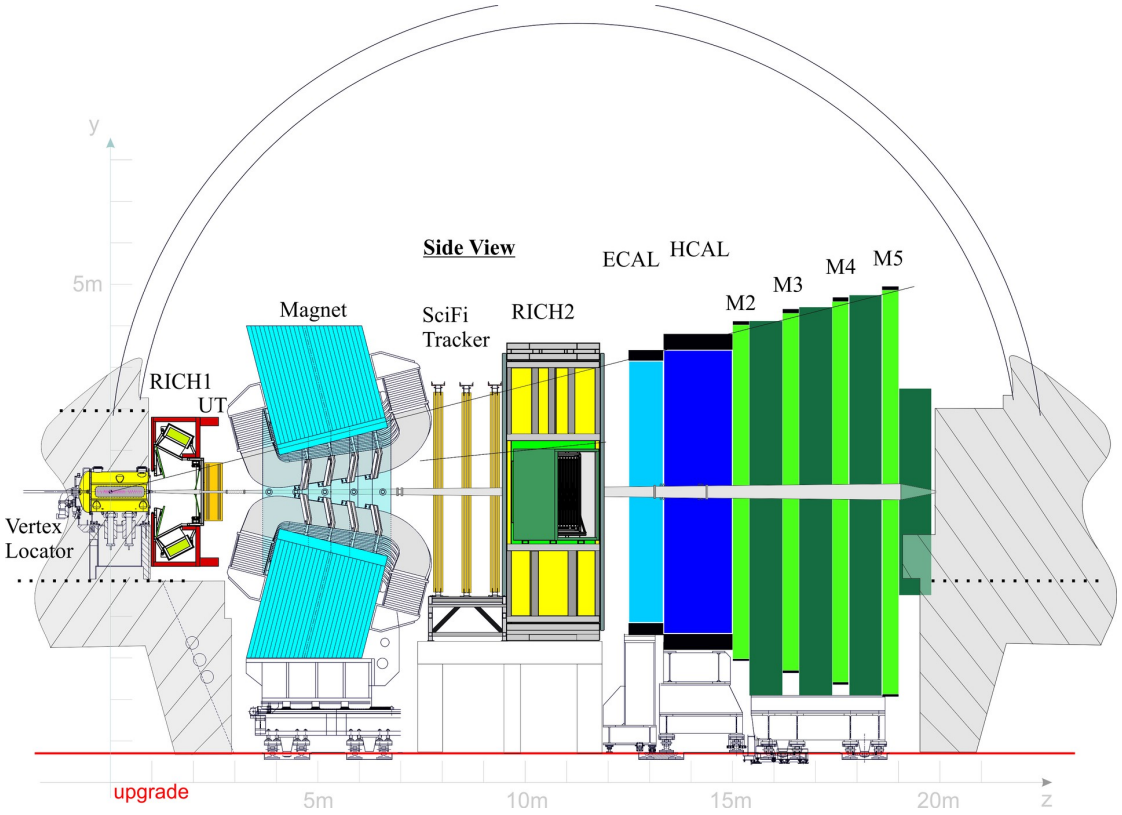


Figure 1.5: Side view of the LHCb detector layout in Run 3. From left to right all the different sub-detectors are shown: VeLo, RICH1, UT, SciFi, RICH2, ECAL, HCAL, and Muon stations. Original image from the LHCb collaboration.

1.3.1 The Tracking System

The current LHCb tracking system consists of three main sub-detectors: the VeLo, the UT, and the SciFi [3].

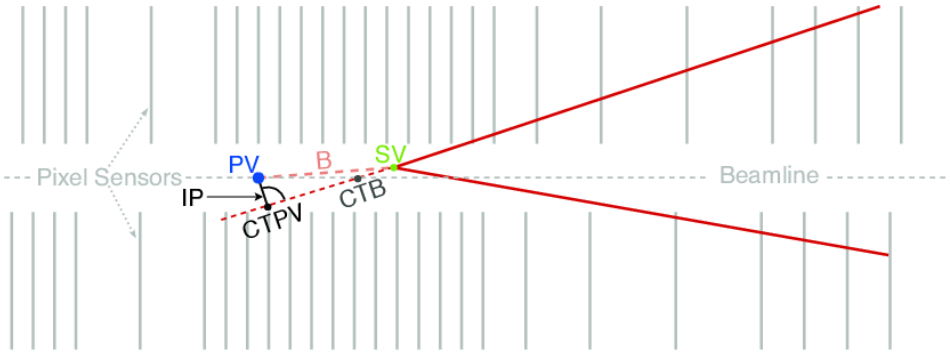
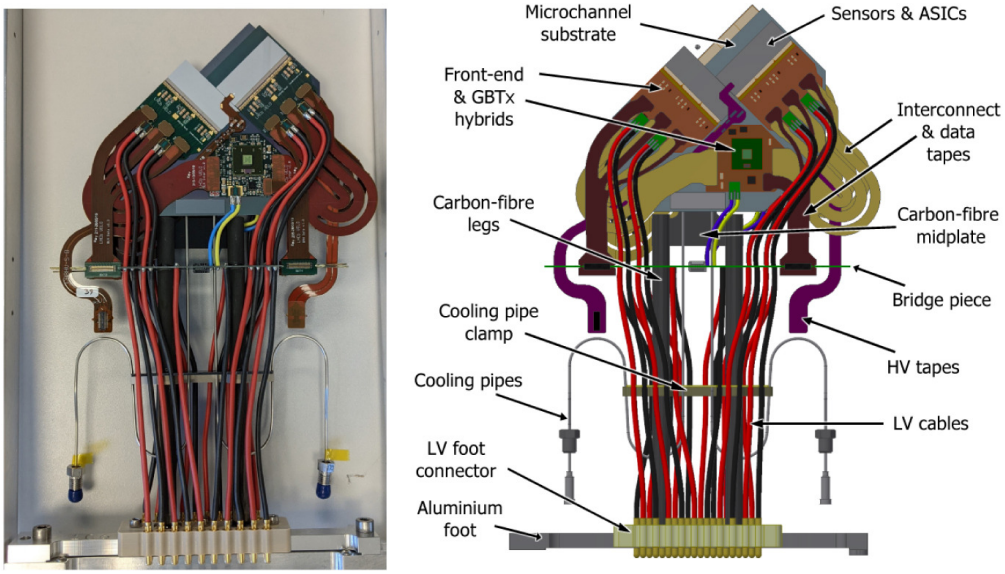


Figure 1.6: Sketch of a B meson coming from the Primary Vertex PV and decaying inside the VeLo at the Secondary Bertex (SV) emitting two daughter particles (red lines). The distances Closest To Beam position (CTB), Closest To PV position (CTPV), and Impact Parameter (IP) are shown. Original image from [8]

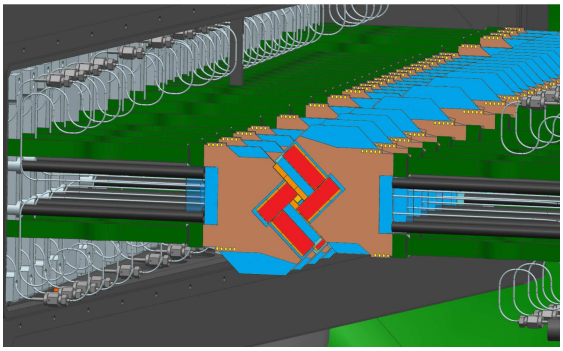
Vertex Locator At LHCb, charm and beauty hadrons typically travel about 1 cm before decaying, making it essential to achieve good Impact Parameter (IP) resolution. The VeLo is the closest sub-detector to the interaction point. Its primary role is to identify the Primary Vertices (PVs) of the pp collisions, locate the Secondary Vertices (SVs), and measure the Impact Parameter (IP). The IP represents the shortest distance between a reconstructed track and the actual origin of the particle, the primary pp collision vertex. This helps to reduce background noise and enables accurate identification of b and c flavored hadrons. A diagram of a B meson produced in a pp collision, highlighting the reconstructed parameters, is shown in Figure 1.6.

To meet the required specifications, the VeLo is constructed with 52 silicon sensors positioned on either side of the beam line and oriented perpendicular to it. Each module consists of four pixel sensors that are 200 μm thick and have an active area of 42.46×14.08 mm^2. The pixels are square, with a pitch of 55×55 μm. The VeLo is located just 5.1 mm from the LHC beams and is designed in two retractable halves that close onto the beam line only during stable collisions. This design helps to minimize the radiation damage to the sensors and protects the detector from potential beam losses. Figure 1.7 illustrates the VeLo modules and their mechanical construction, while Figure 1.8 shows the performance of VeLo in Run 3 based on measurements and simulations produced in 2024.

The VeLo is housed in an isolated vacuum vessel and is separated from the main LHC vacuum by an aluminum foil with a thickness of 150 um. The RF foil acts as a shield against interference from the circulating beams while keeping multiple Coulomb scattering effects minimal due to its thinness and low atomic number. A CO_2 cooling

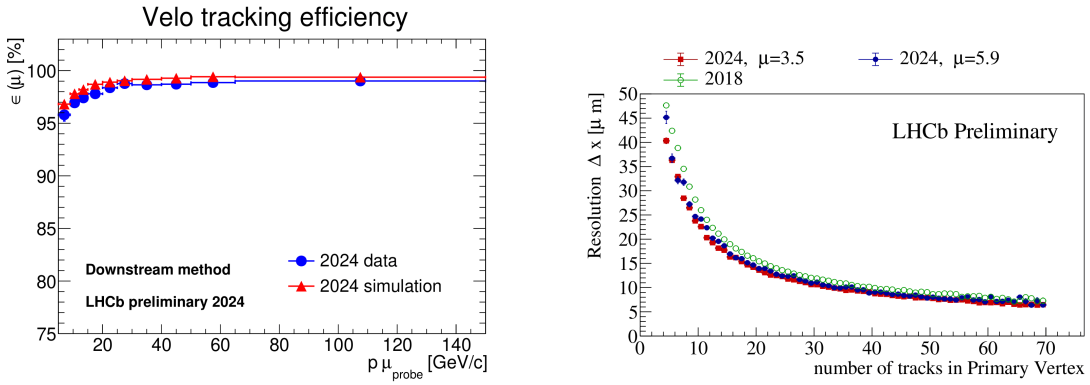


(a)



(b)

Figure 1.7: **(a)** Closeup picture and drawing of one side of the 26 stations of the VeLo before the installation. These stations will close onto the beam line in order to achieve high resolution in impact parameter measurements and vertex location. **(b)** An illustration of the VeLo closed during stable beams [19]. Courtesy of the LHCb collaboration.



(a) (b)

Figure 1.8: **(a)** The plot shows the VeLo tracking efficiency and its dependency on the momentum of the probe track using 2024 data and simulation. [49]. **(b)** The plot shows the PV resolution in the x-axis as a function of the number of tracks. [48].

system is used to remove the heat generated by the detector and the electronics within the vacuum vessel by keeping it at 245 K.

Upstream Tracker The UT is located before the magnet and consists of four layers of silicon strips. This sub-detector is divided into two stations, referred to as UTa and UTb, which are separated by a distance of 315 mm. Each station contains two layers with a skew angle of 5° between them, allowing for two-dimensional coordinates through stereoscopic projection and minimizing the misidentification of multiple tracks. The layout of the UT stations is depicted in Figure 1.9. The silicon sensors have a thickness of 250 μm and an expected hit resolution of 50 μm.

By utilizing data from both the VeLo and UT, it is possible perform a fast estimate of momentum and transverse momentum of charged particles. This capability allows the real-time reconstruction framework to approximate the particle trajectory, thereby reducing computational requirements and enhancing reconstruction quality.

Scintillating Fibre Tracker The final component of the tracking system is the SciFi, located between the magnet and the RICH2 detector. It consists of three stations (T1, T2, T3), each containing four layers. Each layer utilizes 2.5 m long multi-cladding wavelength-shifting scintillating fibers as the active material. In each station, the layers are tilted at ±5° relative to each other to provide two-dimensional coordinates through stereoscopic projection. Each layer is composed by 12 modules, resulting in a total of 144 modules across the entire sub-detector. The SciFi tracker stations are shown in Figure 1.10.

The SciFi tracker offers a hit resolution of 100 μm and provides high granularity in its active region. A higher resolution is not required given, the multiple scattering that occurs in the upstream detectors.

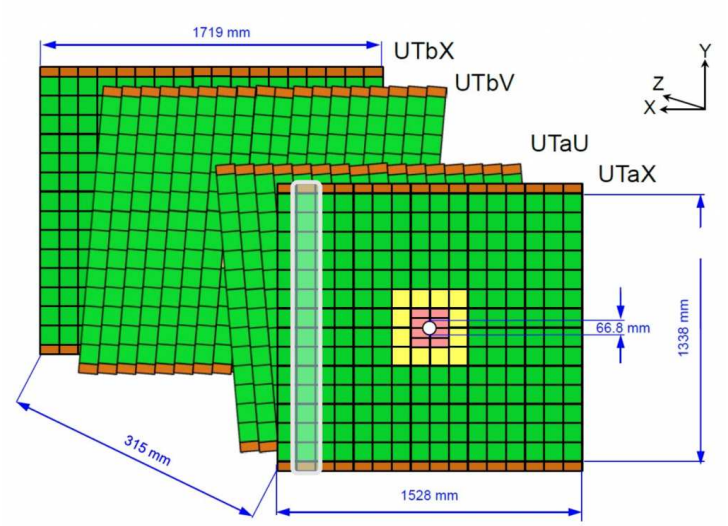


Figure 1.9: Illustration of the layout of the UT. It can be seen the 5° offset between the modules to achieve 2D resolution. Presented in [3].

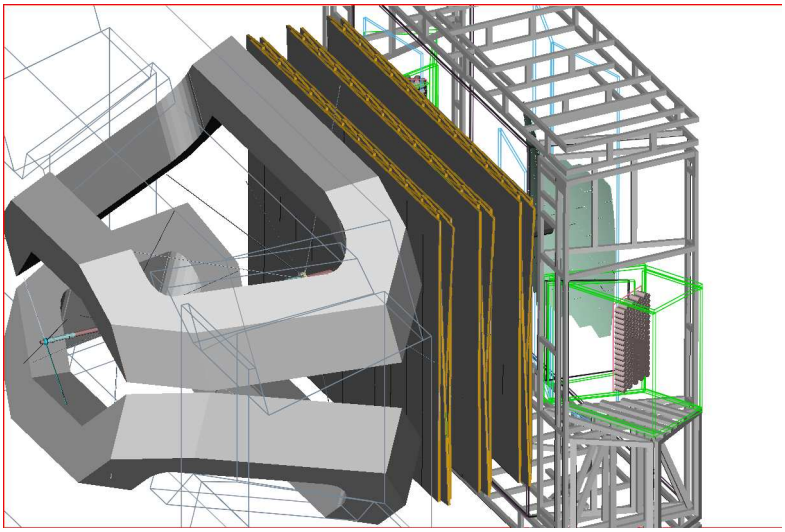
Magnet To measure the momentum (p) and transverse momentum (p_T) of charged particles, a conventional dipole magnet is used to bend their trajectories in the horizontal plane. This magnet consists of two identical coils, each weighing 25 tons, symmetrically placed inside a 1450-ton yoke. The primary component of the magnetic field is directed along the y axis, bending the particle trajectories in the x-z plane. The integrated field strength is 4 $T \cdot m$, and the field profile is well characterized, which is essential for accurate track reconstruction. The polarity of the magnet can be inverted during data collection to counterbalance the asymmetries in the detector layout, which have to be taken into consideration to evaluate CP violation parameters.

Track reconstruction Figure 1.11 demonstrates how the tracking system classifies tracks. The tracks relevant for event reconstruction are categorized as long and downstream tracks. Long tracks provide the best spatial and momentum resolution, thanks to the data gathered by the VeLo. Downstream tracks are primarily associated with the decay products of long-lived particles and exhibit lower resolution and efficiency because they do not involve VeLo data. Other tracks may be used for calibration purposes [34].

1.3.2 Particle Identification

The remaining sub-detectors are used for Particle Identification (PID), which is essential to identify final states and to reconstruct decay processes. PID is achieved through three types of sub-detectors: Ring Imaging Cherenkov (RICH) detectors, calorimeters, and the Muon system.

(a)



(b)

Figure 1.10: Overview of the SciFi. (a) shows six modules installed in the detector's cavern, while (b) illustrates the SciFi tracker's position between the magnet yoke and the RICH2 [3].

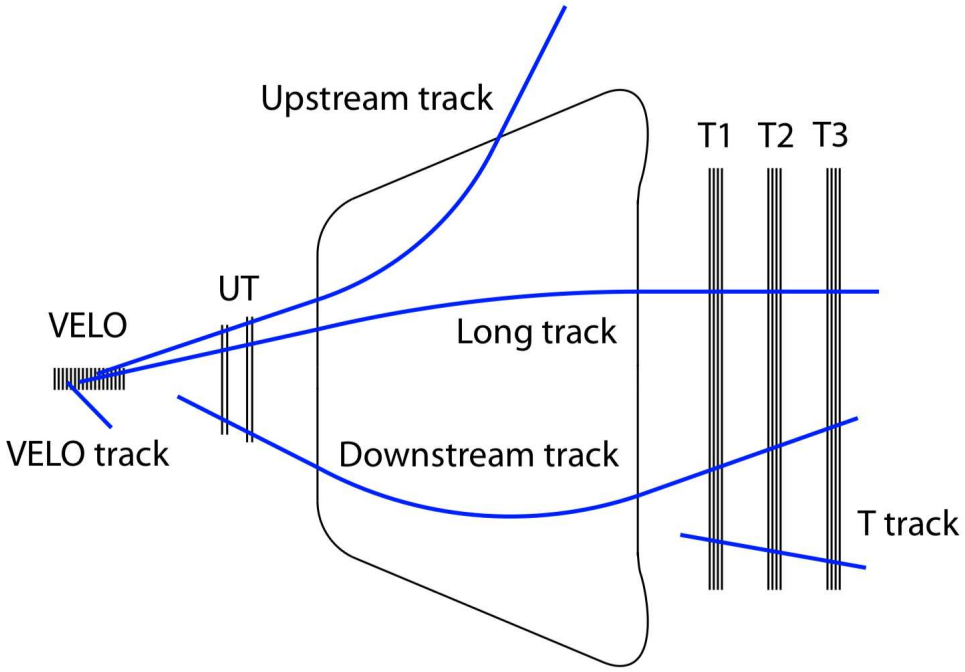
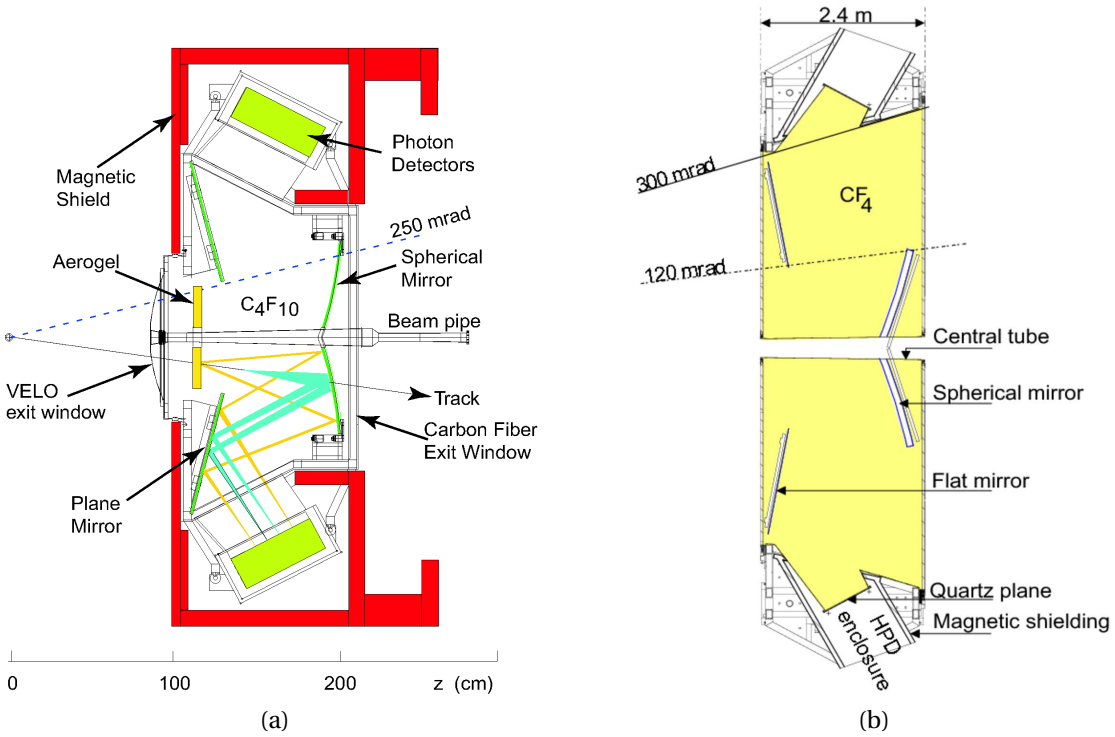


Figure 1.11: Illustration of the track reconstruction using the sub-detectors in the tracking system [8].



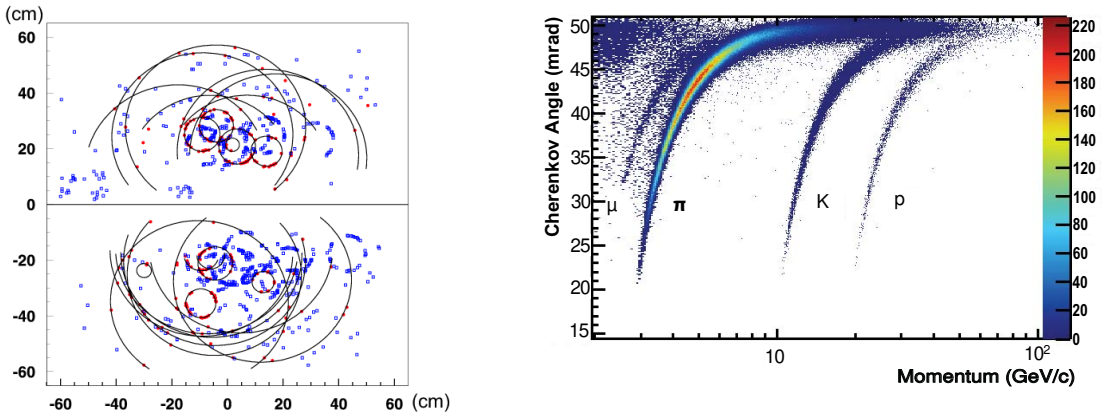
(a) (b)

Figure 1.12: **(a)** Side view of the RICH1 detector. **(b)** Top view of the RICH2 detector. Presented in [3].

Ring Imaging Cherenkov Detectors RICH detectors take advantage of the Cherenkov effect to measure the velocity of particles. When a charged particle moves through a dielectric medium, it generates temporary electric dipoles, resulting in the emission of electromagnetic radiation. If the particle travels faster than the speed of light in that medium, constructive interference occurs, creating a wavefront at a specific angle θ_c related to the particle's speed. The relationship is defined as follows, where n is the refractive index of the medium:

$$\cos\theta_c = \frac{1}{n\beta} \tag{1.3}$$

Here, $\beta = \frac{v}{c}$ represents the ratio of the particle's speed to the speed of light in a vacuum. As the name implies, Ring Imaging Cherenkov detectors reconstruct the emission ring, the radius of which gives a measure of the Cherenkov angle θ_c, and calculate β to identify and differentiate between charged particles using the momentum information derived from the tracking system. Light emitted from the medium is collected by mirrors and detected using photomultiplier tubes.



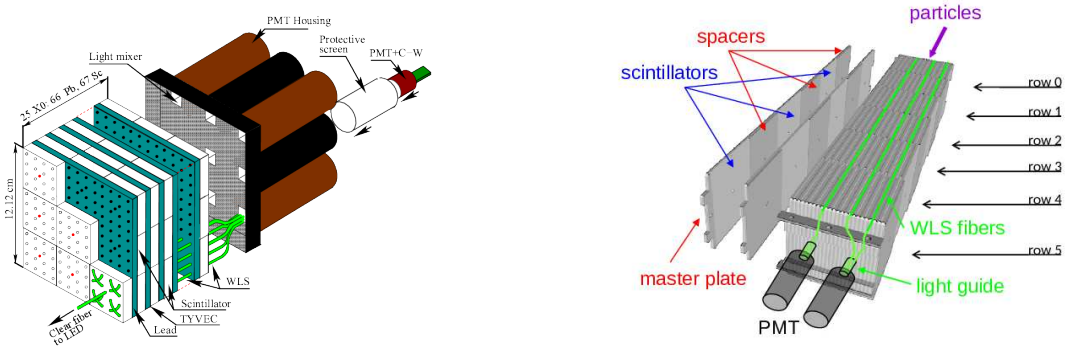
(a) (b)

Figure 1.13: **(a)** An example of the readout from the RICH1 (Run 1 data). Photons are detected by the Multi-anode Photo Multiplier Tubes and then rings are reconstructed. **(b)** Particle identification using the reconstructed Cherenkov angle as a function of track momentum in RICH1 (Run 1 data). Original plots from [3] and [6].

The RICH system consists of two stations shown in Figure 1.12, each utilizing mediums with different refractive indices to measure various momentum ranges. RICH1 is positioned between the VeLo and UT detectors, designed to operate within the momentum range of 1 GeV/c $< p <$ 60 GeV/c, using C_4F_{10} as the medium, with $n = 1.0014$. The sensors are an array of Multi-anode Photo Multiplier Tubes.

RICH2, located after the SciFi detector, covers the momentum range of 15 GeV/c $< p <$ 100 GeV/c. It uses CF_4 as the medium, with $n = 1.0005$, along with a 5% addition

of CO_2 to quench scintillation. An example of a RICH event is shown in Figure 1.13a, while Figure 1.13b illustrates how particles can be classified by their momentum and measured Cherenkov angle.



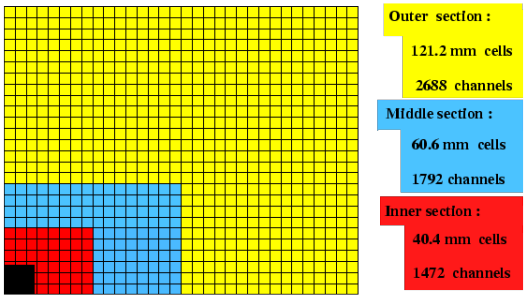
(a) (b)

Figure 1.14: Illustrations of calorimeters modules: **(a)** ECAL, **(b)** HCAL. Courtesy of CERN.

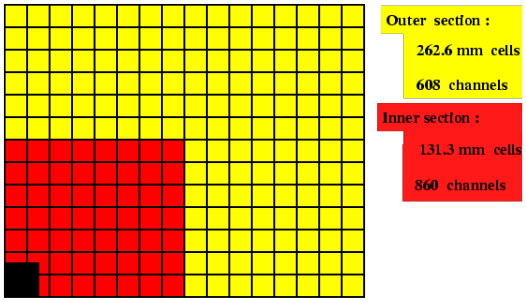
Calorimeters The calorimeter system comprises two sub-detectors: the ECAL and the HCAL. The ECAL is designed to measure the energy of particles that interact electromagnetically, such as photons and electrons. It operates as a sampling calorimeter segmented into modules, with each module consisting of alternating layers of absorber and scintillator arranged in a Shashlik configuration. The absorber is made of lead and has a thickness of 2 mm, while the scintillator is 4 mm thick. Each module measures 120 × 120 mm^2 and is made of 66 layers, resulting in a total depth of 25 radiation lengths. The light signal generated in the scintillator is transmitted to the photon detectors via wavelength-shifting fibers. The readout granularity varies based on the module's position within the detector: modules closest to the beam line feature a cell size of 40 × 40 mm^2, the middle region has a cell size of 60 × 60 mm^2, and the outer region is composed of cells measuring 120 × 120 mm^2. The energy resolution of the ECAL is given by the equation $\sigma(E)/E = (10/\sqrt{E} \bigoplus 1)\%$.

The HCAL uses a similar sampling design to the ECAL but uses steel as the absorber along with scintillating tiles. The absorber-to-scintillator ratio is 5.5:1, providing a total depth of 5.6 interaction lengths. The readout granularity is categorized into two regions, with an inner region cell size of 131 × 131 mm^2 and an outer region cell size of 263 × 263 mm^2. The energy resolution for the HCAL is expressed as $\sigma(E)/E = ((69 \pm 5)/\sqrt{E} \bigoplus (9 \pm 2))\%$. The design of the calorimeter modules is shown in Figure 1.14, while the segmentation of the calorimeters is illustrated in Figure 1.15.

Muon stations The muon system is crucial to identify muons, which appear in various final states of b-hadron decay modes, such as $B_s^0 \to J/\psi(\mu^+\mu^-)\phi$, $B_s^0 \to$



(a)



(b)

Figure 1.15: Details of the segmentation of the calorimeters: **(a)** ECAL, **(b)** HCAL [3].

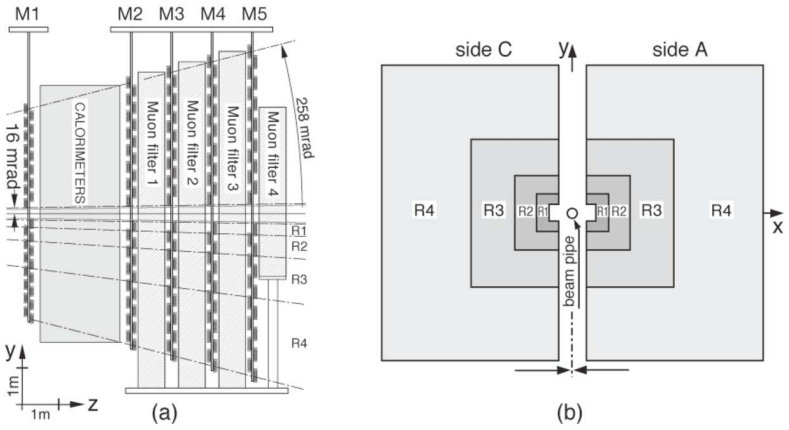


Figure 1.16: **(a)** Side view of the LHCb muon detector (M1 station is removed for Run 3), **(b)** Station layout view where the four granularity regions are indicated [3].

$J/\psi(\mu^+\mu^-)K_s^0$, and $B_s^0 \to \mu^+\mu^-$. Muons with high transverse momentum are also utilized to tag the spectator b-hadron associated with the signal b-hadron.

This system consists of four muon stations (M2-M5), each with an acceptance of ±300 mrad in both the horizontal and vertical planes. Due to muons' significant penetration ability, the muon stations are positioned after all other sub-detectors. Each station is composed by an active layer and an absorber layer. The absorber layer is made of iron, with a thickness of 80 mm. The active layer uses Multi-Wire Proportional Chambers (MWPCs) filled with a gas mixture of $Ar/CO_2/CF_4$ in a 5:4:1 ratio. To maintain uniform occupancy across the detector, the design incorporates varying granularity based on the distance from the beam line, dividing the stations into four regions (R1-R4).

The overall muon identification efficiency exceeds 96%, while the probability of misidentifying hadrons is less than 1%. The configuration and segmentation of the muon stations are illustrated in Figure 1.16.

2 Data Acquisition System

Designed for its unique objectives, LHCb is different from the other collider experiments. The experiment has to manage a 40 MHz bunch crossing rate with hard interactions, while maintaining comparable efficiencies for both hadrons and leptons. This chapter describes in detail the design of the current DAQ system providing a general overview in Section 2.1 and then expanding on the single elements in the next sections following the data flow from the Front-End (FE) to the data storage.

Like other experiments, LHCb requires a system to identify and retain the most interesting events for physics analysis, discarding the rest. In Runs 1 and 2, this was achieved through a L0 implemented in hardware. Selections were conducted at 40 MHz using data from calorimeters, muon systems, and the Vertex Locator (VeLo) [20]. The trigger criteria were based on significant deposits of transverse energy, typically several GeV, alongside momentum measurements from muons, hadrons, electrons, and photons [20]. While this method provided high efficiencies for dimuon events, it resulted in the exclusion of approximately half of the fully hadronic signal decays. In these hadronic decays, the E_T threshold necessary to keep the rate of triggered events within acceptable limits constituted a substantial fraction of the B meson mass. Consequently, the yield of b-hadrons remained nearly constant and was largely independent of luminosity, preventing the experiment from leveraging luminosities exceeding 4×10^{32} cm^{-2}s^{-1}, as illustrated in Figure 2.1. The maximum readout rate of the L0 was constrained to 1.1 MHz.

In Run 3, the DAQ System underwent a redesign, shifting to a triggerless approach with a full software reconstruction and selection. Although this comes at the cost of a 40 MHz readout, resulting in a total throughput of 32 Tb/s with an average event size of approximately 100 kB, the software selection offers greater flexibility. The selection criteria can now utilize information from the tracking system, providing not only particle momenta and energies but also their displacements from the PV. This enhancement leads to significant background reduction for hadrons produced in the PV while facilitating efficient selection of hadronic decays of beauty and charm hadrons. The new event filtering system is structured in to levels called High Level Trigger 1 (HLT1) and High Level Trigger 2 (HLT2), which are described in detail in the next sections.

2.1 Data Flow

This section provides an overview of the DAQ System for the LHCb experiment in Run 3 (2022-2026) [53]. A summary of the full Online chain is illustrated in Figure 2.2.

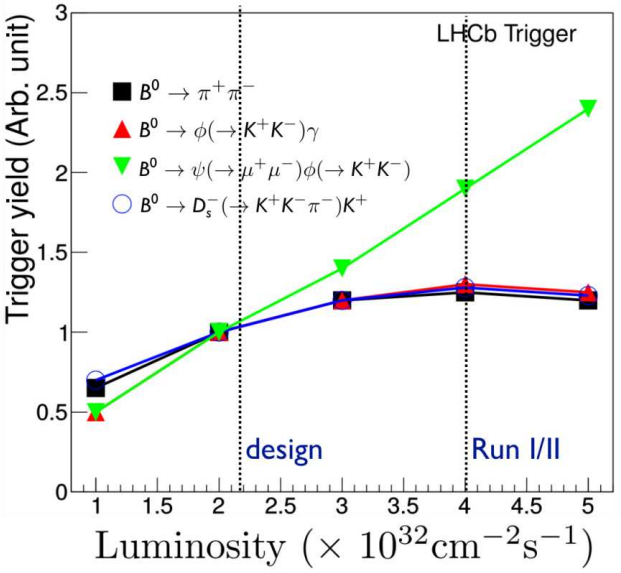


Figure 2.1: Trigger yield as a function of luminosity for the L0 of Run 2. Saturation of the yield in the hadronic channels shows the limits of the L0. Presented in [26].

Data from the sub-detectors is transmitted to the Event Building cluster via optical links. The Event Building process reorders and assembles the data into full events. These events are then processed by HLT1 on the Event Building cluster, utilizing Graphical Processing Units (GPUs). Selected and partially reconstructed events are stored in a buffer that links HLT1 to HLT2. HLT2 algorithms are run on a CPU-based computing farm. HLT2 receives the latest alignment and calibration data and performs event reconstruction at offline quality. The events are subsequently compressed before being transferred to a second buffer, from where they are finally copied to tape storage.

To meet design specifications, almost the entire Online system has been relocated to the surface, leaving only the Front-End Electronic (FEE) underground. This arrangement provides greater flexibility and reduces costs, as the system is no longer constrained by underground space and power limitations.

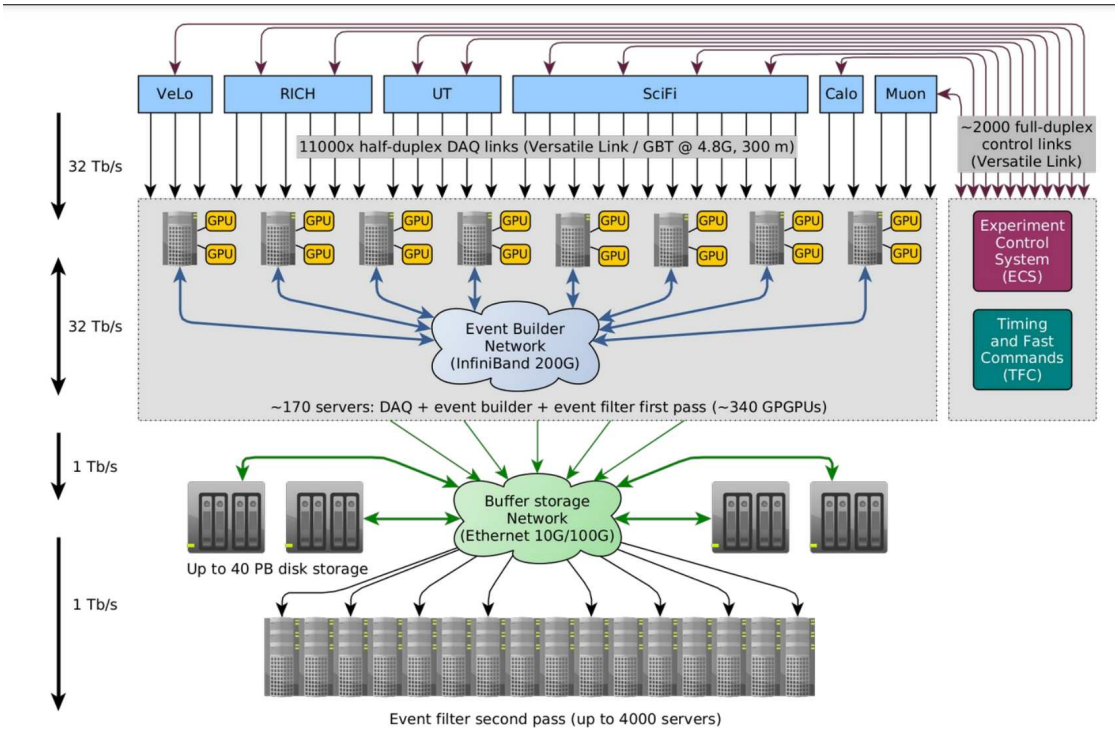


Figure 2.2: Overview of the DAQ for Run 3. Courtesy of Tommaso Colombo (CERN).

2.2 Front Ends

The Front-End Electronic is the first step in the data acquisition chain, converting particle interactions within the detector into measurable electrical signals and digitize them. FEE design depends on detector type, the specific physical quantities measured, and the detector location. For these reasons, FEE usually adopt custom Application-Specific Integrated Circuits (ASICs) to acquire and digitize the signal coming from the sensors. Once digitized, data may be further processed by other components like FPGAs before being sent via a high-speed transceiver ASIC to the Back-End using optical links. Given their placement within or near the detector, all FEE components should tolerate high radiation exposure.

In Run 3, LHCb requires the FEE to be capable of handling the full 40 MHz bunch-crossing rate and relay the event information to the Back-End. To achieve this, LHCb utilizes the radiation resistant GBT chipset [54] along with its associated components, providing a standardized readout interface to the Back-End. Data coming from the FEE ASICs is encapsulated in the GBT protocol which is then decoded in the Back-End. This simplifies the Back-End design, while allowing FEE ASIC designers flexibility in optimizing data formats to meet specific requirements. In addition to data transmission, FEE require control, monitoring, and synchronization, all of which are managed by the GBT chipset within a unified Back-End framework.

2.3 Back-End Boards

The readout cards, known as PCIe40 boards shown in Figure 2.3 [15], serve as the main data acquisition hardware for LHCb. Each board is equipped with high-end FPGA that communicates with the FEE via GBT protocol. Additionally, these boards are connected to the software control system via the PCIe bus of the host server in which they are installed.

The PCIe40 boards come in three configurations:

- **SODIN**: The readout supervisor, SODIN centrally synchronizes and manages event readout by distributing the LHC clock and issuing clock-synchronous commands across the system.
- **SOL40**: This configuration acts as an interface for TFC and the Experiment Control System (ECS) with the FEE. It distributes clock information from SODIN to the FEE via optical links and allows the ECS to control and monitor the FEE over the same link.
- **TELL40**: The primary configuration for data acquisition, responsible for reading event data from the FEE into the DAQ infrastructure. Successive event fragments are buffered into Multi Fragment Packets (MFPs) and transferred to the host server RAM via Direct Memory Access (DMA). Of the approximately 520 PCIe40 cards deployed, 445 are TELL40.

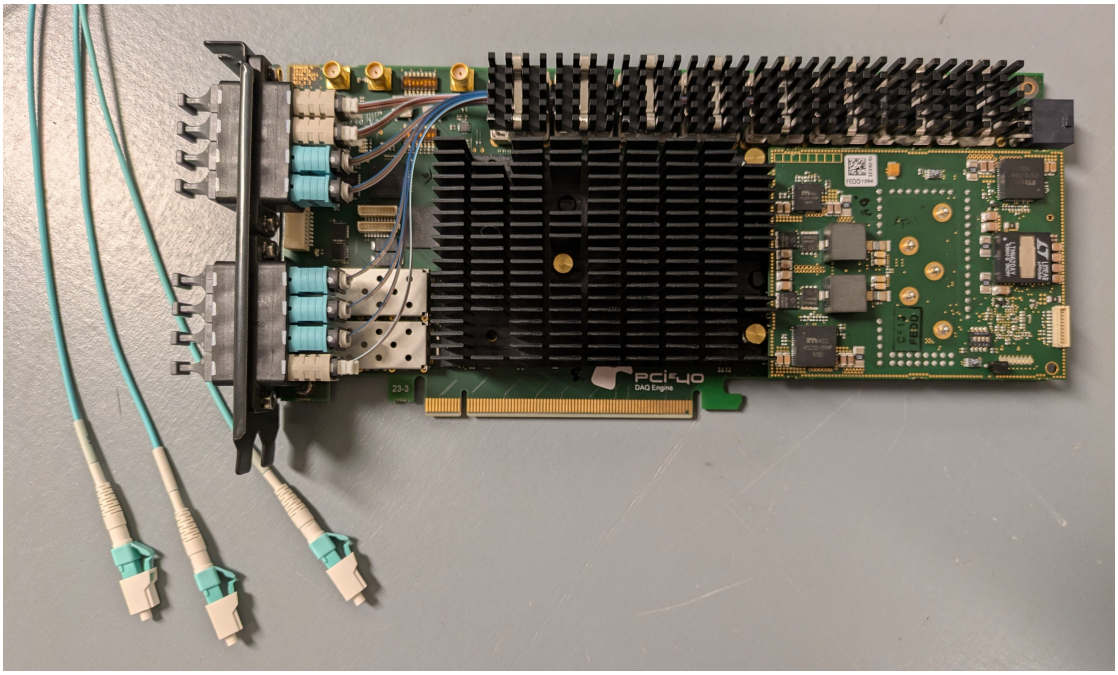


Figure 2.3: The PCIe40 Board: specifically designed FPGA board for data acquisition and control.

A more in-depth description will follow in Chapter 4. In total, around 11,000 GBT optical links connect the detector to the readout boards, and roughly 500 TELL40 boards are required to process the total data throughput at 40 MHz. The DAQ boards are hosted in approximately 170 servers where the event building process is carried out.

2.4 Event Builder

The readout cards are organized into groups of up to three within the EB servers, which results in data being fragmented across the entire cluster. To facilitate the event assembly, data from all sub-detectors must be collected, and all fragments of a single event need to be assembled in one location. This process is referred to as Event Building and it is illustrated in Figure 2.4. Consequently, servers in the Event Building cluster must be interconnected to transmit and receive these data fragments, utilizing a network based on the InfiniBand HDR 200 Gbps technology.

To simplify the Event Building task, the process can be divided into two groups of logical units:

- **RU** collects fragments from the TELL40 boards and transmit them to the BUs.
- **Builder Unit (BU)** receives and assembles the fragments into full events.

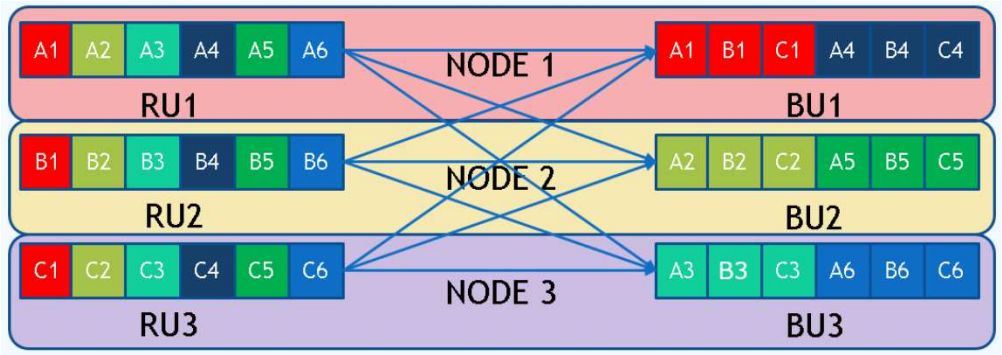


Figure 2.4: Example of the reordering done by the EB nodes. Fragments are sent in an all-to-one manner from RUs to BUs to construct full events. Courtesy of Flavio Pisani (CERN).

All EB nodes — servers hosting the necessary hardware and running the EB software — execute multiple instances of RUs and BUs based on the available hardware. Events generated by the LHCb detector during pp collisions have a nominal size of approximately 100 kB, with individual fragments on the order of $\mathcal{O}(100)$ Bytes. Since modern interconnection technologies are not optimized for transferring small packets of a few hundred bytes efficiently, as shown in Figure 2.5, fragments are grouped into MFPs and subsequently combined into Multi Event Packets (MEPs). The packing factor of the Event Builder is adjustable, with a value of 30000 utilized for Run 3. The MEPs are subsequently transferred to the HLT1.

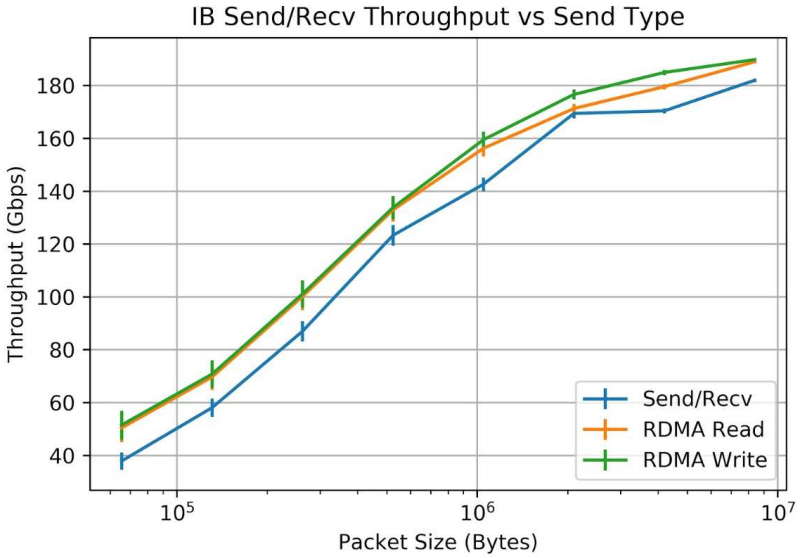


Figure 2.5: Throughput measurements with different packet sizes. These measurements were taken between two nodes of the test cluster of the EB [62].

2.5 High Level Trigger 1

The High Level Trigger 1 (HLT1) is the first software selection applied to the data, developed within the Allen framework [5]. HLT1 must manage an average collision rate of 30 MHz[1]. It performs partial reconstruction and coarse selection, reduces the input rate by a factor of 30, and stores selected events for further processing.

To meet these requirements, HLT1 operates on the same servers as the EB and utilizes GPUs for acceleration. Modern GPUs are well-suited for HLT1, as each physics event is processed independently and can be mapped to individual threads, maintaining high throughput. Additionally, GPUs are designed to execute a large number of floating-point operations per second, which aligns with the computational demands of HLT1 algorithms.

The HLT1 sequence employs three sub-detector systems for event selection. The tracking system reconstructs particle tracks and vertices and determines momentum, while data from the muon stations and calorimeters integrates the particle identification. Selections are based on parameters such as reconstructed momentum, vertex displacement, and muon identification as illustrated in Figure 2.6.

The output from HLT1 is stored in a 40 PB buffer, ready for real-time alignment, calibration, and subsequent processing by the High Level Trigger 2 (HLT2).

[1]While the LHC bunch crossing rate is 40 MHz, the LHCb experiment experiences a lower average collision rate (beam-beam) of approximately 30 MHz due to LHC design constraints. The remaining 10 MHz are beam-empty and empty-empty crossings.

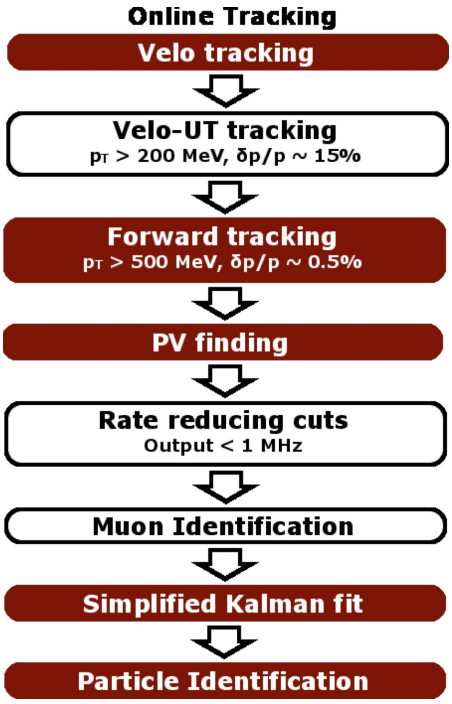


Figure 2.6: HLT1 selection and reconstruction algorithms for Run 3, respectively. Courtesy of LHCb.

2.6 Real-Time Alignment and Calibration

Precise spatial alignment and accurate calibration of detector components are necessary for optimal physics performance. Variations in temperature, pressure, and physical movements caused by operational conditions (e.g., magnet polarity changes, opening and closing of the VeLo) can alter the response of subdetectors. The position and orientation of detector elements within the global reference frame must be known to a precision higher than the single-hit resolution to maintain accuracy. Therefore, a Real-Time Alignment and Calibration (RTA) process is essential to account for these changes detector conditions.

The RTA procedure, pioneered during Run 2 [17], has become crucial in Run 3 [64]. This procedure is performed at the beginning of each LHC fill. Samples for the procedure are selected by dedicated lines in HLT1. These samples are stored in a buffer to gather sufficient statistics and allow the RTA and HLT2 processes to operate independently of HLT1. Once alignment and calibration are completed, HLT2 processes the data.

The RTA procedure, illustrated in Figure 2.7, aligns the tracking system, RICH mirrors, and muon system. Alignment is particularly important for the VeLo, since it is moved from a safe position to a distance of 5.1 mm from the beam when stable beams conditions are reached. After alignment, results are compared with previous data and updated if significant differences are observed.

The remaining component of the RTA procedure is calibration. This involves measur-

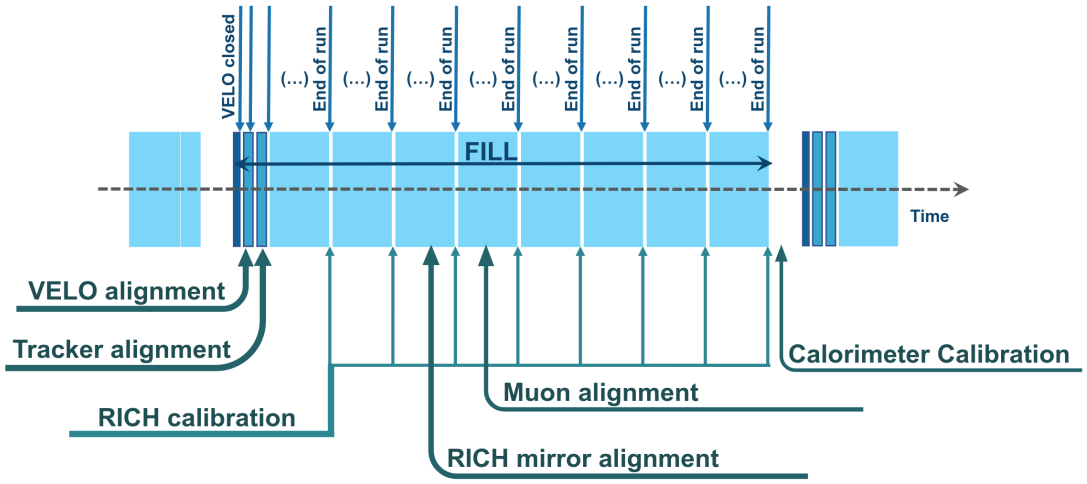


Figure 2.7: Real-Time Alignment and Calibration tasks in Run 3. executed for each fill. The ordering from left to right indicates the expected amount of time each task takes, from shortest to longest. Taken from [64]

ing the refractive index of the RICH gas, which is updated at each run, and performing an absolute calibration of the ECAL high voltage. This ECAL calibration is verified through analysis of the $\pi^0 \to \gamma\gamma$ decay mass distribution for each calorimeter cell. During Run 2, the ECAL calibration was updated on a monthly basis.

2.7 High Level Trigger 2

The final stage in the online data flow is the High Level Trigger 2 (HLT2). This second high-level trigger performs full event reconstruction and reduces the rate to approximately 100 kHz using more than 256,000 CPU cores. HLT2 utilizes more than a thousand selection lines to select relevant events using reconstructed data. Selections in HLT2 can be inclusive or exclusive, employing a range of techniques from rectangular cuts to neural network methods. Events output from HLT2 are reconstructed with offline-level quality, allowing for preliminary physics analyses to be conducted within a few days of acquisition.

HLT2 optimizes storage by reducing the information stored for each event. This is achieved through a custom data format known as the *Turbo Stream* [16], where only candidates identified by the trigger are retained, and non-essential event parts are discarded in a process called selective persistency. The Turbo format reduces event size by about an order of magnitude. This efficient data handling minimizes disk and tape space, as well as computing resources, by eliminating raw data and bypassing offline reconstruction. With a 10 Gb/s bandwidth limit for HLT2 output, the Turbo format enables the recording of more events within the same bandwidth compared to a Full Stream format.

2.8 Disk Buffers

The system has two disk buffers, one between HLT1 and HLT2 and the second between HLT2 and the permanent storage. These buffers decouple the filtering layers, containing any backpressure and avoiding discarding events. The first buffer, called *BigBuffer*, allows the collection of enough data to perform the alignment and calibration. This buffer size is a total of 40 PB distributed across approximately 3200 hard disks of 14 TB each. The buffer has to handle 100 GB/s sequential writes and 70 GB/s sequential reads. This throughput constraint dictates the number of hard drives required.

The second buffer of 10 PB stores the output of the HLT2, making the system independent for a few days in case of a permanent storage failure. Moreover, it allows the aggregation of events coming from the 5,000 computing nodes into large streams based on the physics content. The system has to handle a more moderate 20 GB/s in both sequential reads and writes and it is able to buffer around 6 days of run.

3 Future Data Acquisition Systems for High Energy Physics

This chapter explores future trends in DAQ systems for HEP experiments, focusing on the High Luminosity LHC (HL-LHC) and its implications for detector and DAQ design. The challenges and advancements necessary to adapt to the HL-LHC's demanding environment are discussed in detail in Section 3.1 for the LHCb Experiment. Future requirements for HEP experiments like ATLAS and DUNE are covered in Section 3.2. Finally, overall DAQ trends are discussed in Section 3.3.

3.1 LHCb

As outlined in the LHCb Upgrade II TDR [46], the LHCb experiment anticipates operating at over seven times the current (Run 3) instantaneous luminosity. This significant increase introduces new experimental challenges, with approximately 40 pp interactions per bunch crossing, leading to higher particle multiplicities and rates. Additionally, radiation damage to sub-detectors will be a growing concern.

The next generation of data acquisition systems will require technological advancements across the entire chain. A key point is integrating fast timing information to suppress combinatorial backgrounds under high pile-up conditions.

Tracking At peak luminosity, the LHCb detector must handle approximately 2000 charged particles per bunch crossing within its acceptance. This creates significant challenges for the tracking system for radiation hardness and data acquisition, particularly the VeLo. The higher pile-up means that the separation between primary vertices is reduced by almost a factor of three. This impacts the capability of differentiating primary vertices and maintaining the resolution, without misassociating heavy flavour decays. To overcome these limitations, a new approach is to introduce precision timestamping, converting the VeLo into a full 4D tracking detector. With a time resolution of approximately 20 ps, the detector will be able to exploit the spread in time of the vertices of 180 ps. Achieving these specifications requires redesigned electronics, including a new ASIC, as well as upgraded mechanical components.

Other tracking components must also be upgraded to ensure precise momentum measurements. The SciFi will be upgraded to the Mighty Tracker detector, which will employ a hybrid system composed of silicon pixel detectors for the inner regions and scintillating fibres for the outer regions. The UT will be upgraded as well to handle the

higher occupancy and, therefore, will produce a higher throughput, reaching 9 Gb/s on the hottest chip.

Particle Identification High-quality particle identification is essential for precision measurements. Existing sub-detectors will be upgraded to improve granularity and incorporate fast timing. The RICH system, for example, will undergo substantial modifications to manage increased track multiplicity and integrate fast timing to suppress combinatorial backgrounds. These upgrades require a new FEE capable of supporting silicon photomultipliers instead of the current multi-anode photomultiplier tubes. Furthermore, the development of the *FastRICH* ASIC is necessary to implement the new time-to-digital converters. This new chip will be able to separate the Cherenkov photons with a time resolution of 25 ps. The chip will also use a more complex data format based on the Aurora protocol [13], which should improve link utilization and reliability. The decoder of this new protocol, which will be used in the back-end, has been developed as part of this thesis and is described in Section 6.2.1.

Similarly, the ECAL will be redesigned to withstand extreme radiation doses, up to 1 MGy for the innermost modules, while incorporating fast timing capabilities. This redesign involves developing custom large-area sensors capable of timing resolutions below 50 ps. New ASICs will be required to measure the energy and arrival time of electrons and photons. These chips, called *ICECAL65* and *SPIDER*, are under development.

Data Acquisition At an instantaneous luminosity of 1.5×10^{34} cm^{-2}s^{-1}, the LHCb detector will generate approximately 200 Tb/s of data — five times the Run 3 throughput, as illustrated in Figure 3.1. Raw data must be processed in real-time and reduced by approximately four orders of magnitude before permanent storage. Given the high pile-up, new selection techniques will be evaluated to reduce the throughput as early as possible in the chain, such as performing low-level reconstruction using custom processors. These filtering tasks will be based on heterogeneous architectures using high-end FPGAs, such as the upgraded readout board and GPUs. The evaluation of event reconstruction acceleration with FPGAs using High-Level Synthesis (HLS) is part of this thesis' work and the results are presented in Section 6.3.

In addition to data rate challenges, the harsh radiation environment and spatial constraints necessitate radiation-hard custom links, which are typically slower than Commercial Off-The-Shelf (COTS) alternatives. Current LHCb DAQ systems utilize custom FPGA readout boards (PCIe40) to manage these links. For the upgrade, the number of links will rise to 30,000, and their speed will double with the adoption of the lpGBT protocol [60]. A new FPGA board, PCIe400, will handle these high-speed links, but lpGBT's distance limitations will require splitting the event-building farm between the surface and the cavern, as shown in Figure 3.2.

Precision timing information from the upgraded sub-detectors will play a crucial role in efficiently separating reconstructed objects by their parent pp interaction. These new timing requirements, targeting precisions of $\mathcal{O}(10\,\text{ps})$, will drive upgrades

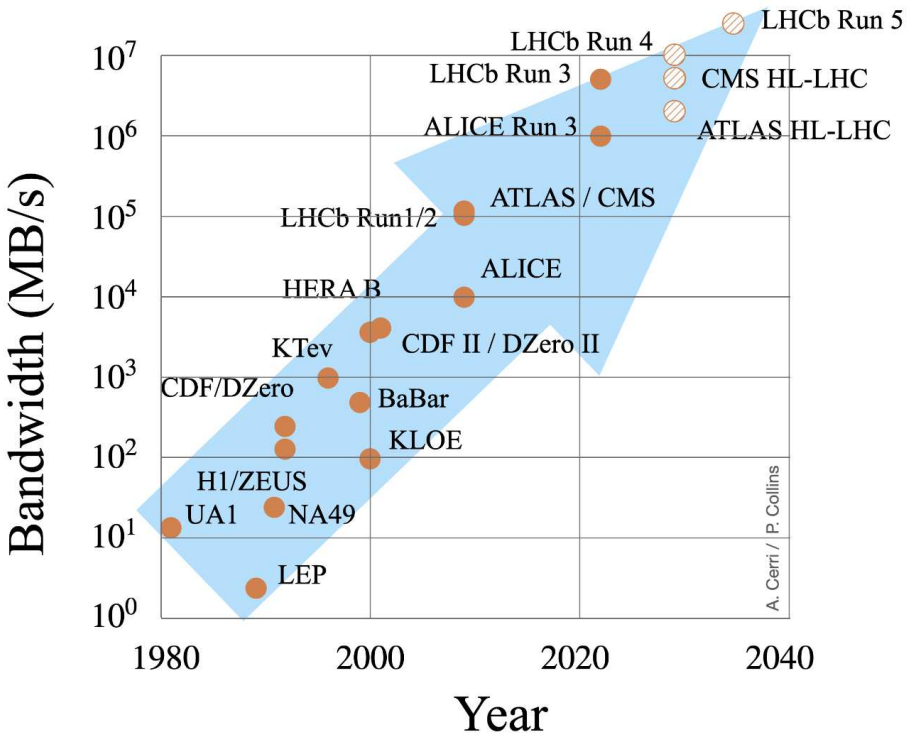


Figure 3.1: Throughput evolution of HEP experiments. Courtesy of CERN.

to the experiment's timing distribution system. The timing and fast control system will require FPGAs capable of reaching these specifications, distributing the information over the lpGBT links, and implementing precision timing protocols like White Rabbit [65].

Cost-effective DAQ system design remains a challenge. A potential solution is a system based on custom FPGA boards for link conversion to standard 100 Gb/s Ethernet. While Ethernet has already been used in many readout systems of HEP experiments, such as CMS and the Cherenkov Telescope Array (CTA), the latest advancements make it a viable option over PCIe for high throughput DAQ. This design will allow the use of existing COTS Ethernet hardware, significantly simplifying the DAQ architecture. A Proof of Concept (PoC) has been designed as part of this thesis and is described in Section 6.2.2.

Research is also focusing on integrating Ethernet directly on the FEE. This solution will greatly simplify the system design and reduce costs; however, the implementation is more challenging, and its use could be limited by the technology itself, such as the protocol's radiation hardness.

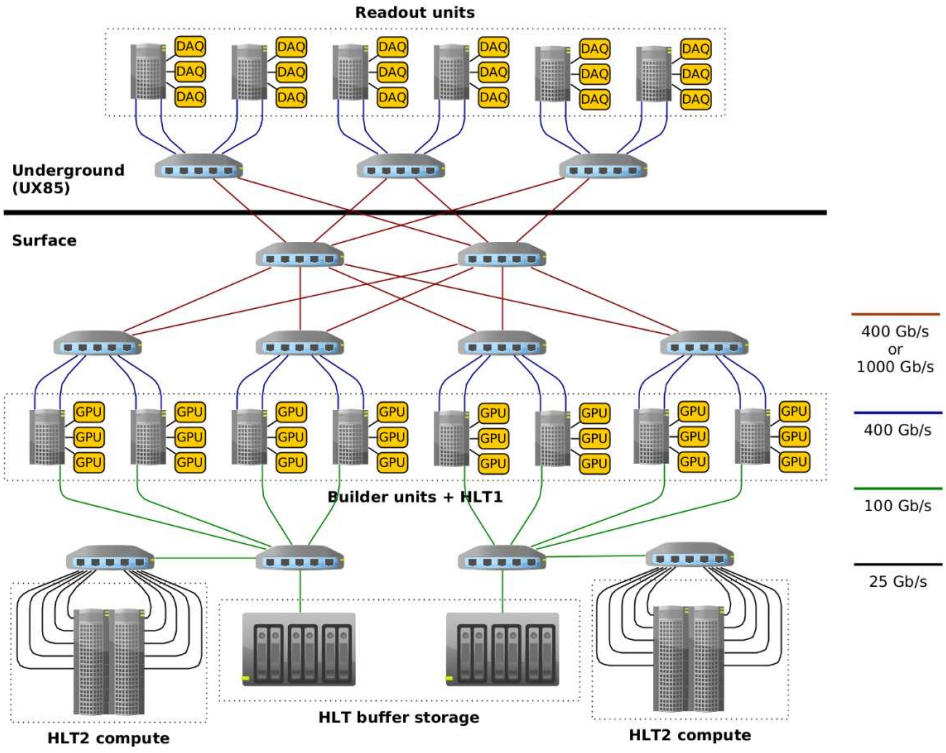


Figure 3.2: Illustration of the layout of the LHCb Upgrade II DAQ system [46].

3.2 Other Experiments

ATLAS The ATLAS Trigger and Data Acquisition (TDAQ) System is a conventional triggered system where the low level trigger is implemented in hardware and it controls the data acquisition path in real-time [11][12]. The current ATLAS detector is capable of producing a raw data throughput roughly 100 TB/s, hence the requirement of a hardware trigger which reduces the rate down to a more manageable 2.5 TB/s. Moreover, a design based on a hardware trigger is not a constraint to the ATLAS physics program as it was for LHCb.

For the HL-LHC program, ATLAS expects the accelerator complex to provide an instantaneous luminosity of 7.5×10^{34} cm^{-2}s^{-1}. In this scenario, the ATLAS experiment will need to operate with a high pile-up configuration of $\mu \simeq 200$, posing significant challenges for the trigger and reconstruction systems. To achieve the required performance, the trigger system will be redesigned, as illustrated in Figure 3.3, allowing for increased latency and rate. Additionally, the DAQ boards must support a greater number of faster links to the detector readout. The Level-0 trigger will continue to be implemented in hardware but will utilize more advanced FPGAs to enhance computational capabilities. The PCIe-based DAQ boards, known as Front-End Link Exchange (FELiX), will be upgraded with a new FPGA, capable of handling high-speed links reaching 25 Gb/s and incorporating high-precision timing for next-generation detectors, such as the High Granularity Timing Detector.

Furthermore, the Event Filter system, responsible for selecting events from data accepted by the Level-0 trigger, will transition from a CPU-only farm to a heterogeneous architecture. This new approach will integrate accelerators such as GPU and FPGAs, following the model pioneered by LHCb.

Deep Underground Neutrino Experiment The Deep Underground Neutrino Experiment (DUNE) represents a next-generation neutrino physics experiment based in the United States of America [25]. Its primary goal is to explore fundamental questions in particle physics, such as CP violation in the lepton sector, proton decay, and neutrino interactions. The experiment involves directing an intense beam of neutrinos from Fermilab to a massive detector located 1300 km away. The detector is based on a liquid argon time projection chamber (LArTPC) and its prototype - ProtoDUNE - is being developed at CERN.

Unlike collider experiments, neutrino experiments such as DUNE prioritize very high uptime and the ability to perform long-duration full readouts of the detector. Despite these differences, DUNE DAQ shares similarities with collider experiments in its approach to data management and processing.

Recently, the DUNE DAQ system has transitioned from a PCIe-based FELiX architecture to an Ethernet-based readout [66], as shown in 3.4. This shift was motivated by the benefits of leveraging COTS hardware and utilizing open-source software components. The new Ethernet-based design enables flexibility and cost-effectiveness while maintaining the required data throughput. Although DUNE handles a smaller data rate compared to LHC experiments, this approach demonstrates the potential

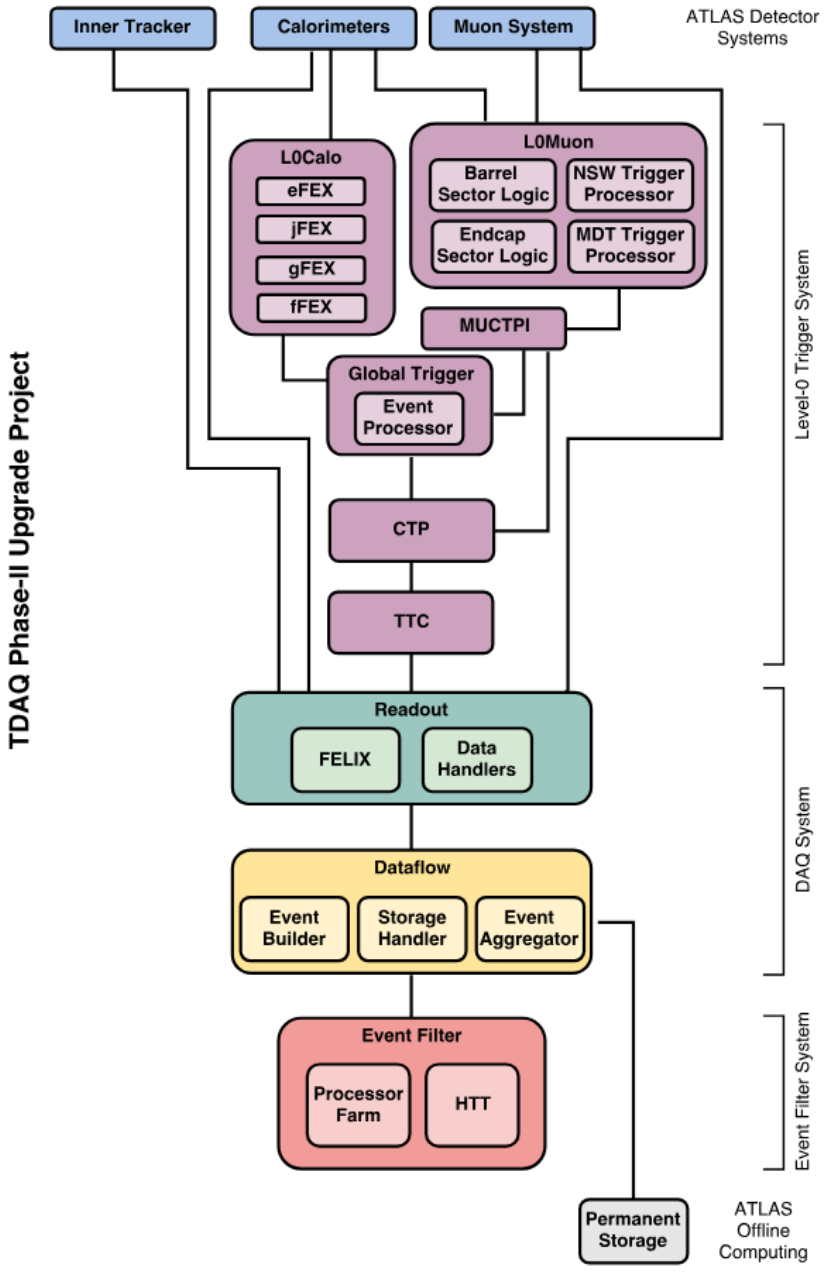


Figure 3.3: Illustration of the layout of the ATLAS Phase II Upgrade DAQ system [12].

for hight throughput HEP experiments to adopt multi-gigabit Ethernet technology without the need for a complete overhaul of DAQ architectures.

3.3 Trends

The plans for future data acquisition systems in high-energy physics experiments reveal several clear trends. One of the most critical challenges will be managing a significantly larger number of high-speed data links, operating at 10 Gb/s or higher. This demand requires the development of new readout boards equipped with modern FPGAs capable of efficiently handling the increased throughput. LHCb plans to adopt the PCIe400, the successor to the current PCIe40, while ATLAS will upgrade its system with a new version of FELiX.

Another significant trend is the evaluation and adoption of Ethernet-based systems for data acquisition. These systems provide several advantages over traditional PCIe-based solutions, including increased bandwidth, scalability, and cost efficiency. Designs following this approach will be discussed in detail in Section 6.2.2.

Moreover, the growing data volumes and real-time processing demands are driving a shift from CPU-centric processing to heterogeneous computing architectures. These architectures integrate CPUs with accelerators such as GPUs and FPGAs, which are better suited for high-throughput, parallelized tasks like event filtering and reconstruction. By leveraging accelerators, DAQ systems can sustain performance, efficiency, and cost-effectiveness as data rates continue to rise. FPGA acceleration will be explored in Section 6.3.

Finally, the integration of precise timing information into DAQ systems is becoming increasingly critical, particularly for experiments operating under high pile-up conditions. Precision timing enhances the separation of overlapping events and improves the overall accuracy of event reconstruction. Meeting these stringent requirements will necessitate the implementation of high-precision clock distribution to the FE via lpGBT. Additionally, adopting high-precision timing distribution systems and protocols, such as White Rabbit [65], will be essential in addressing these new challenges.

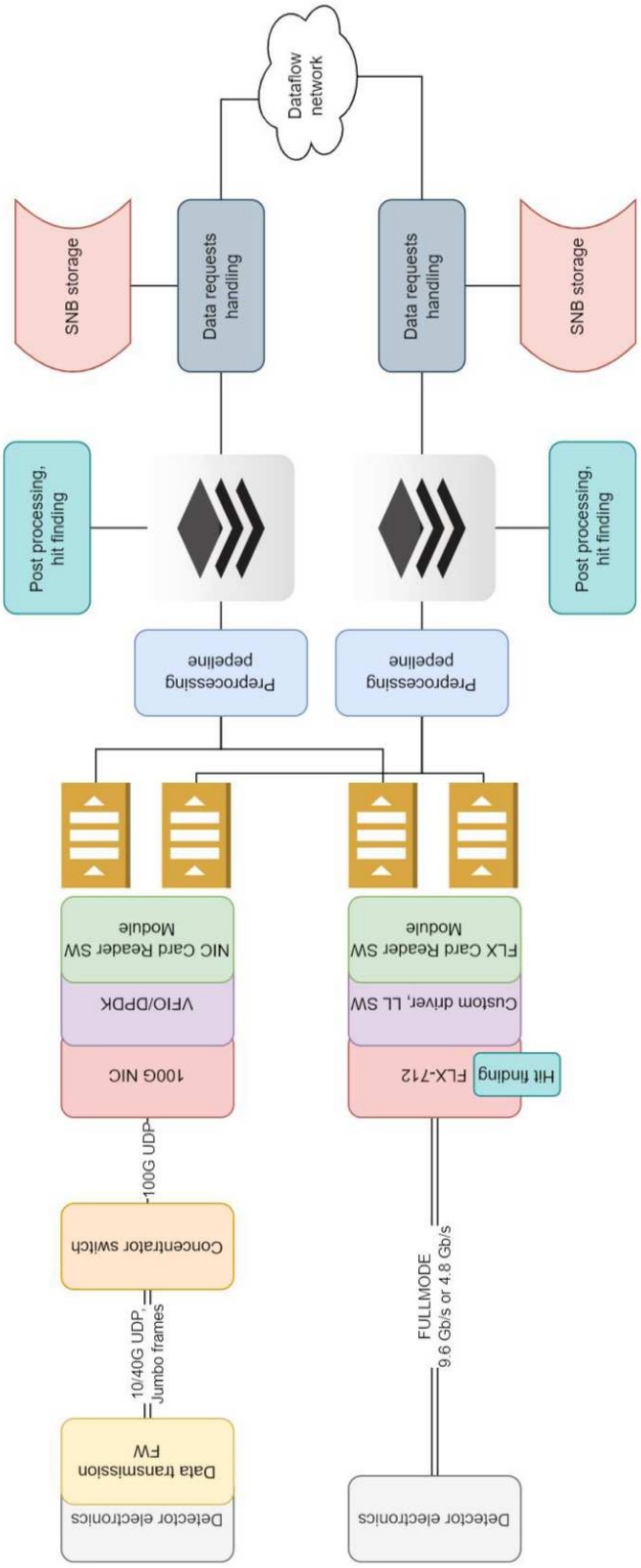


Figure 3.4: Architecture of the DUNE DAQ systems using Ethernet (left) or PCIe (right) [66].

4 FPGA Architecture at LHCb

FPGA-based readout boards serve as the interface between the sub-detector systems and the DAQ, managing both control and data acquisition. The design and architecture of these boards have imposed strong constraints on the overall DAQ system.

Section 4.1 provides a detailed overview of the current FPGA architecture, including the system's design principles, gateware, and the shortcomings identified during deployment and commissioning. Section 4.2 builds on these insights, outlining the lessons learned and their application to the requirements and design decisions for future systems.

4.1 Current Situation

4.1.1 Hardware Design

At the time of the system's design, the LHCb DAQ was transitioning from a triggered system, implemented through a combination of hardware and software, to a fully software-based architecture.

The current readout board, the PCIe40, was conceived with flexibility and modularity in mind, enabling the use of a single, standardized board to address the key functions of the DAQ system: data acquisition, timing distribution, and control. The design of this readout scheme was centered on advancements in computing technology, that changed the resource and cost constraints. This evolution made it feasible for the EB nodes—the initial software compute element in the data acquisition chain—to utilize the PCIe bus for ingesting data from the subdetectors. The key features of the PCIe40 board include the PCIe interface, the detector links, and the TFC interface, which are discussed in the following paragraphs.

PCIe The PCIe bus was introduced in 2003 as a successor to the older PCI and AGP standards. As the name suggests, this bus interconnects peripherals to the CPU via a high-speed, low-latency, and scalable interface. PCIe employs a point-to-point architecture, eliminating shared bandwidth and contention issues. Furthermore, it uses a packet-based protocol that enhances reliability through error detection and correction. Its flexibility also comes from lane aggregation, enabling devices to operate with one to 32 lanes, depending on throughput requirements and host capabilities.

The PCIe standard evolved rapidly due to its widespread adoption in networking and computing accelerators, such as GPUs, which demanded higher bandwidth and lower latency. A key feature supporting high throughput is DMA, a functionality

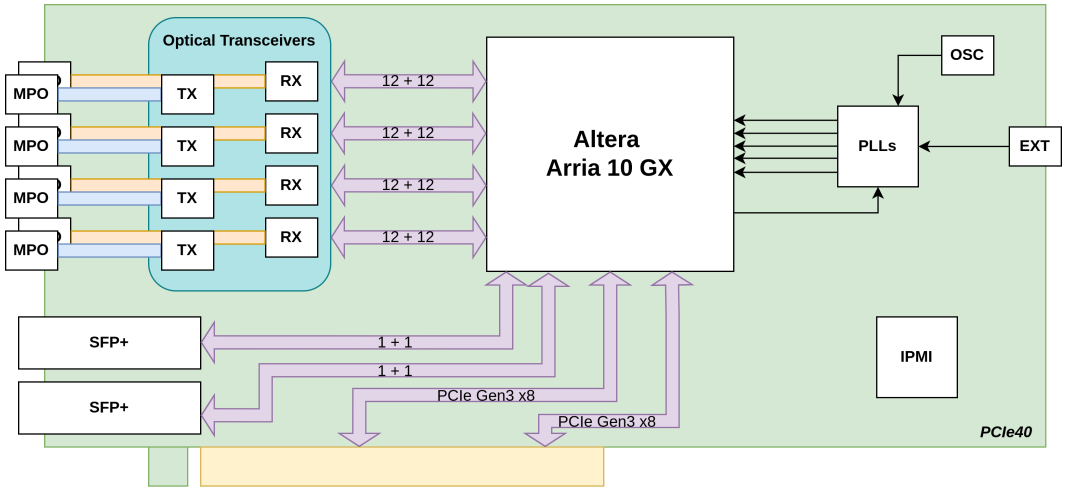


Figure 4.1: High Level Synopsis of the PCIe40 board. Purple arrows indicate high speed transceivers capable of up to 17.4 Gbps.

implemented by devices that allows data transfers directly between device memory and system memory without CPU intervention. This saves computing resources and avoids potential bottlenecks.

PCIe generation 3.0, introduced in 2010, significantly increased bandwidth, offering up to 1 GB/s per lane with improved link efficiency. It replaced the 8b10b encoding scheme with the more efficient 128b130b scheme. As a result, a 16-lane PCIe 3.0 link could achieve a full-duplex bandwidth of 100 Gb/s.

This standard, available at the time of the system's design, enabled the current architecture by delivering sufficient data bandwidth without necessitating a large number of PCIe-based readout boards. Furthermore, the rise of Artificial Intelligence, which demanded high-performance clusters of powerful GPUs, led to the widespread availability of servers with numerous PCIe slots and accelerator-friendly form factors, making them both accessible and cost-effective. The PCIe40, as outlined in Figure 4.1, offers a PCIe Gen3x16 physical interface which is split in two separate PCIe Gen3x8 interfaces because of technology limitations in the FPGA. This requires the host to support slot bifurcation, which is a common feature on compute servers.

Detector Links The current system employs 10,818 links to read out the entire detector, with an average of 24 links per card. The PCIe40 cards can support up to 48 links, organized in modules of 12, as long as the total throughput remains within the limits of the available PCIe bandwidth. All sub-detectors, except for the VeLo, utilize the GBT ASIC [54], which facilitates communication with the sub-detector front-end electronics (FEE). In one direction, collision data is transmitted to the DAQ, while in the other direction, control and timing information is forwarded from the backend to the FEE.

The GBT ASIC connects via the Versatile Link (VL) [67], a high-speed link capable of achieving a throughput of 4.8 Gb/s towards the backend. On the detector side, this

Sub-detector	DAQ links	Average Throughput (Tb/s)	Average Throughput per DAQ link (Gb/s)
VeLo	1246	3278	2.6
UT	1344	6593	†4.9
SciFi	4096	4089	1.0
RICH1	1584	741	0.5
RICH2	960	521	0.5
Calorimeters	1028	3066	3.0
Muon	560	697	1.2
Total	**10818**	**18984**	-

Table 4.1: Number of DAQ links and average throughput for each subdetector, based on data acquired during Run 3 at $\mu = 5.3$. Bandwidth measurements are derived from EB counters, which may include additional processing effects. For instance, in the case of the UT subdetector, padding inflates the throughput beyond the maximum available bandwidth of a GBT link.

link is driven by a radiation-hard transceiver called VTRx, whereas on the backend, standard commercial off-the-shelf (COTS) transceivers can be used because of the absence of radiation-hardness requirements. The GBT and VTRx system provides a common interface for sub-detectors, reducing development overhead for both the sub-detector teams and the online system developers. Additional companion ASICs, such as the GBT-SCA [22], interface the GBT to standard protocols like I^2C, SPI, and JTAG.

In LHCb, the GBT is configured to send 112 bits of data per bunch clock cycle, with the exception of the VeLo, which uses the Gigabit Wireline Transceiver [38] for readout and can transfer 128 bits of data. The measured average throughput on the links is presented in Table 4.1, which shows that most of the subdetectors only use a fraction of the available bandwidth. Based on geometrical, occupancy, and throughput constraints, each subdetector is responsible for implementing its own data format and the corresponding decoding in the gateware, which will be discussed in detail in the next section.

Timing Interface The Trigger and Fast Control (TFC) system [32] is responsible for managing and distributing clock signals, timing information, trigger data, and both synchronous and asynchronous commands across the entire readout system. Its architecture is shown in Figure 4.2. The TFC master, known as SODIN, handles the distribution of timing and synchronous commands, oversees the dispatching of events to the EB, and regulates the system's acquisition rate. SODIN is connected to the rest of the system through a Passive Optical Network (PON) [59], which interfaces with the control cards. These control cards then relay information to the DAQ cards via a secondary PON or to the FEE using the GBT protocol.

The entire system is based on the same PCIe40 card, which is programmed with

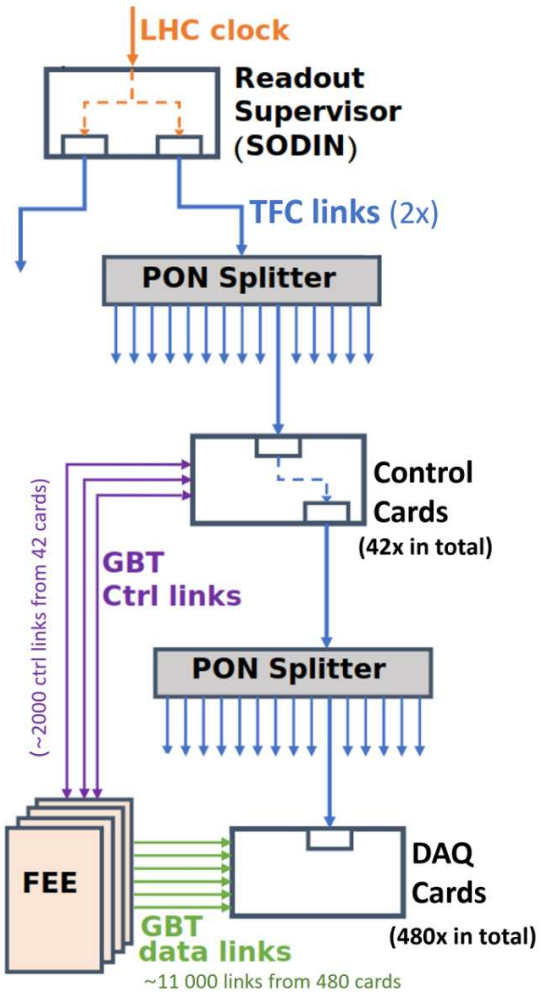


Figure 4.2: Architecture of the TFC system [32]. Readout supervisor, control cards, and DAQ cards are all based on the PCIe40 card.

different gateware[1] depending on its specific role within the TFC chain. This configuration requires that the PCIe40 hardware is equipped with high-precision and low-jitter components, such as Phase-Locked Loops (PLLs), to meet the stringent requirements of the timing chain. Specific attention is dedicated to the clock phase determinism. FEs recover the clock information from the GBT links used for control and use it as a reference for the digitalization of the detector signals. Therefore, it is necessary that the phase information is propagated in a fixed manner, avoiding shifts which would impact alignment and synchronization. These timing requirements are more demanding than those for data acquisition to ensure accurate and reliable timing information across the entire DAQ system.

Following this overview of the key features of the PCIe40 and their benefits to the entire DAQ chain, the next section will examine how the hardware characteristics, such as its modularity, has influenced the design decisions for the gateware.

4.1.2 Gateware Design

The gateware of the PCIe40 board is available in three main flavors: the readout supervisor SODIN, the control cards SOL40, and the readout cards TELL40. The first two are utilized in the timing and control part of the DAQ system, while the TELL40 is specifically designed for data acquisition. This section focuses on the TELL40 gateware.

The PCIe40 hardware was designed as a common platform for all sub-detectors. It is modular and configurable to a certain degree by end users, while benefiting from the cost efficiency of large-scale production. Similarly, the TELL40 gateware follows the same philosophy, offering a complete framework that abstracts the board's basic functionality and provides standard components that encapsulate sub-detector-agnostic logic. Due to the constraints of the PCIe Endpoint Intellectual Property (IP), which supports only PCIe Gen3x8, the gateware is structured into two independent data streams as shown in Figure 4.3.

Common Gateware In Figure 4.3, the common elements of the gateware are highlighted in green and blue. The green components belong to the Low Level Interface (LLI), which handles sub-detector link transceivers, GBT decoding, the PCIe Endpoint IP, and TFC PON logic. These elements are specific to the FPGA and board used, as they rely on hard IPs and device-specific functionalities. The LLI interfaces with the rest of the system using two interfaces: an Avalon Streaming Interface for data and an Avalon Memory Mapped interface for registers and control via the ECS.

After passing through the LLI, data is processed by a set of common components operating on individual links. These components include:

- **Raw Data Decoding:** This block processes FE fragments received after GBT protocol decoding. It isolates FE fragments, extracts the Bunch Crossing ID

[1] Configuration of the FPGA hardware behavior, also known as firmware.

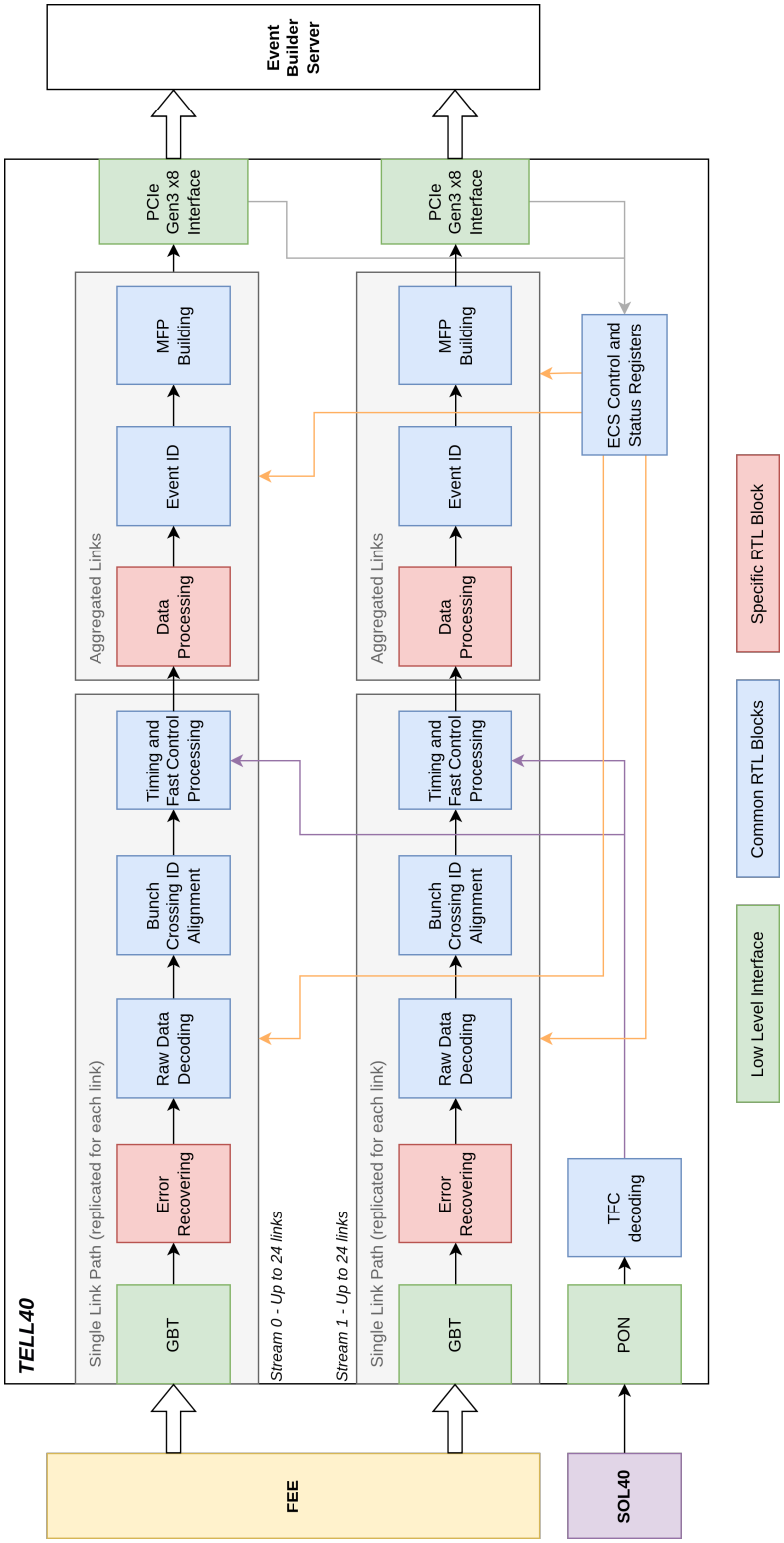


Figure 4.3: High-level overview of the TELL40 gateware: common elements are highlighted in blue and green, while sub-detector-specific components are shown in red.

(BXID) from their headers, and forwards the BXID to the alignment block. It also handles synchronization sequences to reset the BXID counter.

- **Bunch Crossing ID Alignment:** This block ensures BXID consistency across all received FE fragments. It checks that BXIDs increment monotonically and uses buffering and deskewing to align fragments from different links. The block ensures fragments with the same BXID are synchronized for further processing.
- **TFC Processing:** This block verifies and stores timing and control information received from the TFC system. It provides these details to the alignment block and implements data filtering based on the acquisition rate configured via ECS.

Aligned data is now prepared for detector-specific processing and moves to the **Event ID** block. This block assigns a unique Event ID tag to each fragment, ensuring fragments corresponding to the same orbit and BXID in the entire system are associated correctly during event building. To optimize bandwidth usage, not all fragments are transmitted to the EB. This block also handles and corrects errors, such as missing fragments, skipped sequences, or jumps in Event IDs, and performs sanity checks on the Avalon Streaming Interface.

Once processed, the fragments are buffered and prepared for transmission over PCIe. The **MFP Building** block groups 30'000 fragments into a packet format called MFP, which includes headers for decoding. The DMA controller then packages these MFPs into DMA requests and sends them to the EB via the PCIe interface.

Beyond the data path, the TELL40 gateware includes additional essential components:

- **TFC Decoding:** Decodes TFC information received via PON.
- **ECS Control and Status Registers:** Serves as an interface to the ECS, enabling runtime configuration of board parameters and real-time monitoring of error counters and status indicators.

These common elements are parameterizable, allowing them to adapt to the diverse requirements of most sub-detectors. This approach simplifies the design process and streamlines development.

Detector-specific Blocks In Figure 4.3, the red blocks represent components that sub-detector groups can customize to meet their specific requirements. To facilitate this development, the collaboration provides templates with generic algorithms. These customizable blocks are:

- **Error Recovery:** This block is essential when using the GBT protocol in Wide-Bus mode, which removes the standard error correction mechanism to achieve higher bandwidth. In such cases, sub-detector groups are responsible for developing their own error recovery mechanisms, along with the ECS monitoring interface. Additionally, this block is tasked with tagging fragments with error flags when a data transmission error cannot be recovered.

- **Data Processing:** This is the most critical block in the data path, responsible for decoding the sub-detector-specific FEE data format and processing the data further, such as by applying zero suppression to reduce throughput. Sub-detector groups are free to implement any algorithm they require, provided they adhere to the input and output interfaces, the protocols used, and the resource constraints of the device. Some sub-detectors even offload part of their reconstruction algorithms to this block to reduce the computational load on the High Level Trigger (HLT).

Development and Deployment As outlined above, the LHCb Online group provides engineering support and development for the various LHCb subdetectors, covering tasks from readout board gateware to control system integration. With a team of fewer than ten members, the group must address diverse demands from the subsystems at all stages of their evolution, including design, test beams, assembly, and commissioning. This process is labor-intensive, requiring significant effort and time to identify, reproduce, and resolve issues that arise during the development cycle. To streamline these tasks, the team has implemented an automated pipeline for continuous integration of FPGA gateware [29].

This pipeline offers multiple advantages across the integration cycle. During the development phase, both core team members and external collaborators from the LHCb collaboration can contribute using a distributed version control system based on Git, managed through CERN's GitLab infrastructure [37]. Git facilitates issue tracking, such as failed builds and tests, enables open discussions regarding developments, and supports milestone tracking. Developers can contribute to the codebase regardless of their location or time zone.

The gateware repository is organized using logically distinct Git submodules, which are linked repositories that allow independent versioning of code components, as shown in Figure 4.4. A central repository oversees the state of these submodules and serves as a reference for the build system. Developers submit merge requests to propose their releases for review.

When a merge request is submitted, a continuous integration pipeline in GitLab, shown in Figure 4.5, is triggered. This pipeline performs Register Transfer Level (RTL) simulations for predefined gateware configurations and, upon success, proceeds to compile all possible configurations for the readout boards used in LHCb. Both the control and data variants, SOL40 and TELL40, for each board type must be recompiled. This process involves over 30 configurations, with a total compilation time of approximately 100 hours. However, an intelligent pipeline reduces this by tracking changes in components and recompiling specific configurations only when dependencies have been modified. Additionally, compute time is optimized through job distribution across a dedicated cluster of 8 servers. During the later stages of development, gateware revisions often target localized changes, limiting the scope of recompilations, which can typically be completed overnight.

The build process, once successfully completed, generates outputs that are packaged together with associated software and control system components in an RPM

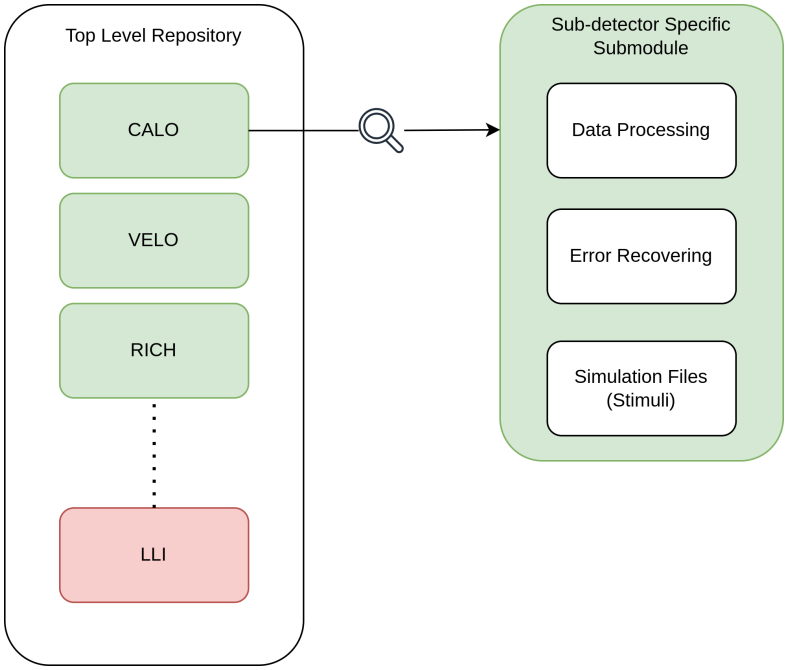


Figure 4.4: High Level Diagram of the Gateware Repository. On the left, the top level repository which contains all subdetector specific (in green) and common submodules (in red). On the right, the main components in a generic subdetector specific submodule.

format. These packages are then distributed through dedicated RPM repositories. Deployment servers then fetch the latest validated release for operational use.

4.1.3 Lessons Learned

The architecture outlined above has been successfully deployed in production, demonstrating the viability of the approach. However, several lessons have been gathered over the years that should be considered for future designs.

Lack of Testing A significant shortcoming of the current design is the limited testing framework available for the gateware. Each sub-detector relies on a full dataflow simulation, which uses a manually generated stimulus file fed into the error recovery block. The output of the MFP building block is then verified against the initial stimuli. Unfortunately, most simulations use a single stimulus file for each gateware variant, meaning only a single case is tested. Additionally, no unit tests are implemented to target individual components within the dataflow.

This results in poor coverage, which lead to labor-intensive debugging during commissioning and risking the loss of valuable beam time. Another drawback of full dataflow simulations is their high computational cost. The complexity of the gateware means that simulating even a few milliseconds of operation requires hours to

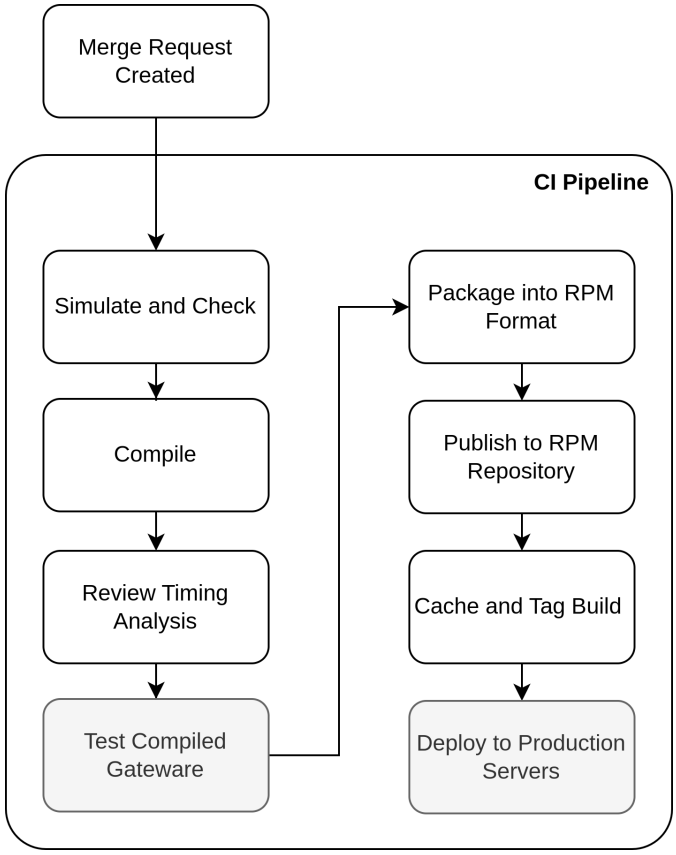


Figure 4.5: The Continuous Integration Pipeline from the PCIe40 Gateware Repository. In grey the steps that are triggered explicitly by manual action.

complete, limiting the frequency and usefulness of such tests during development or debugging.

Vendor Lock-In Both simulations and gateware development were performed using a proprietary software framework, comprising a vendor-specific simulator and compiler. While the use of a proprietary compiler is unavoidable due to its dependency on the hardware, the choice of simulator is more flexible. Proprietary simulators, however, often require licenses that restrict the number of parallel instances, thereby limiting the number of simulations that can be executed simultaneously and reducing overall efficiency.

Additionally, proprietary tools create challenges for portability. Migrating the code to a different simulation or compilation environment requires significant effort, further locking users into the proprietary ecosystem. The reliance on proprietary IPs compounds this issue. While some IPs—such as transceivers and PLLs—must remain proprietary due to technological constraints, others involve basic logic that is broadly available across FPGA platforms. The use of proprietary IPs severely limits the portability of gateware for both simulation and compilation.

Hardware Constraints The readout board successfully demonstrated the feasibility of the PCIe interface and its compactness when integrated directly into the EB servers, which also host GPUs for HLT1. While PCIe remains a widely adopted standard, especially for GPU accelerators, industry trends in High Performance Computing indicate a shift away from PCIe due to challenges related to power, cooling, and space constraints.

This evolution raises concerns about future compatibility. It may become increasingly difficult to source servers with PCIe slots that accommodate the current readout card form factor.

Limited Manpower As discussed, the experiment requires a significant number of different configurations to be able to interface with the different subdetectors' FEE. Even with component reuse between these configurations, a substantial amount of effort is required to maintain the current codebase. Moreover the core team, which counts fewer than ten members, has to support the subdetectors' teams which are composed of designers with diverse levels of expertise, often including students with little knowledge of FPGAs.

4.2 Future Upgrades

Upgrades to the LHCb subdetectors will introduce a higher number of links and increased link speeds to accommodate the demands of higher luminosity. As a result, the readout systems must be enhanced to meet these new requirements. This necessity provides designers and developers an opportunity to address the limitations identified in the current design.

Sub-detector	lpGBT links	Average Bandwidth (Tb/s)	Average Bandwidth per lpGBT link (Gb/s)
VeLo	3400	34	10.00
UT	1888	7	3.71
Magnet Station	1400	5	3.57
Mighty Tracker	9500	30	3.16
RICH	5700	30	5.26
TORCH	4312	27	6.26
PicoCal	2360	21	8.90
MUON	1576	16	10.15
Total	**30136**	**170**	-

Table 4.2: Estimate of lpGBT links and average bandwidth requirements for each sub-detector for LHCb Upgrade II [46].

4.2.1 Hardware

The RICH subdetector will require an upgraded readout system during LS3, with additional subdetectors following in subsequent runs as shown in Table 4.2. This upgrade is primarily driven by the adoption of the lpGBT link protocol, which doubles the current bandwidth and introduces a more complex FE data protocol, requiring significant logic resources for processing. Moreover, the new detectors have more stringent constraints on clock jitter and phase determinism to introduce picosecond-level timing information. While the current readout board can support the lpGBT protocol, the clock performance is not sufficient and the number of available cards is limited, and a new production is infeasible due to obsolete components necessitating a redesign.

To address this, a new board has been designed from the ground up, taking advantage of the advancements in FPGA technology. Its high level diagram is shown in Figure 4.6. The Altera Agilex 7 M-series FPGA has been selected for its cutting-edge features, including a high density of transceivers capable of up to 112 Gbps, hard IPs for 400GbE, and PCIe Gen5x16, providing a four times increase in bandwidth to the host. The chosen FPGA also offers three times the resources of the current PCIe40 and includes 32 GB of integrated High-Bandwidth Memory (HBM). These capabilities enable new possibilities, such as accelerating data processing and reconstruction algorithms. For example, the proposed DoWnstream Tracker (DWT) custom processor [47] will use these boards to perform track primitive reconstruction for the SciFi detector.

4.2.2 Gateware Development

Transitioning to a new FPGA generation, even within the same vendor family, is a labor-intensive process due to the technological differences between devices. A substantial portion of the codebase must be reworked to implement and exploit

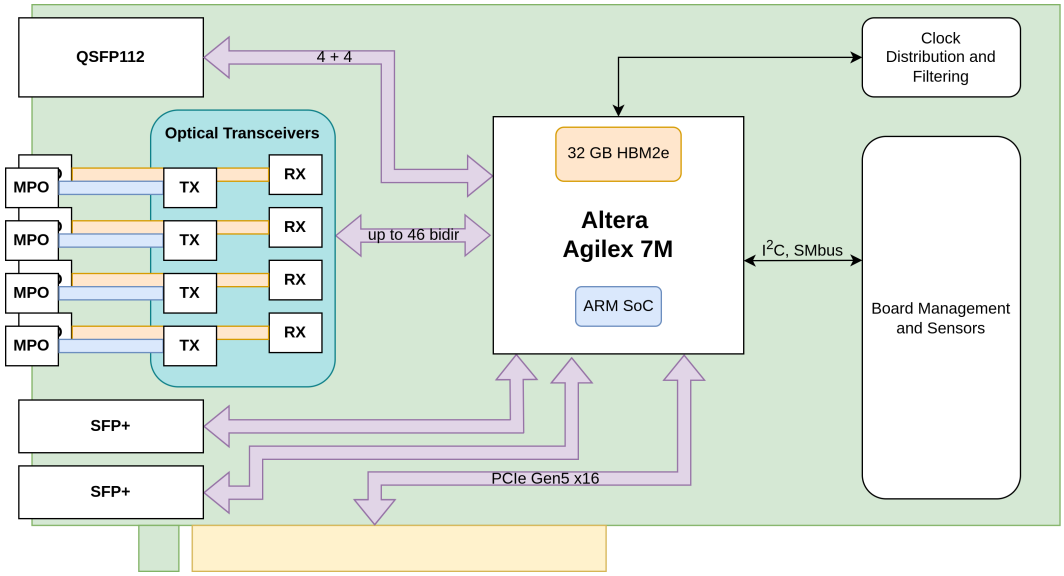


Figure 4.6: High Level Synopsis of the PCIe400 board, the successor of the PCIe40. Purple rows indicate high speed transceivers: Optical transceivers support up to 26Gbps NRZ-encoded speeds, while the ones connected to the QSFP112 are capable of up to 112Gbps in PAM4 encoding.

the new features. To ensure an efficient and effective development process, general gateware development guidelines have been established:

- **Test-Driven Development:** The current development approach is primarily functionality-driven, where developers focus on implementing features first and write tests later. While this might seem expedient initially, it is prone to errors and leads to significant debugging efforts during more critical periods such as the commissioning. Test-driven development reverses this workflow, requiring developers to create testbenches based on application requirements before implementing functionality. This approach ensures comprehensive test coverage, enforces adherence to requirements, and reduces the tendency to cut corners during testing. To facilitate this methodology, a range of modern tools and practices have been evaluated and will be detailed, along with case studies, in subsequent chapters.

- **Vendor Agnosticism:** Much of the existing codebase relies on proprietary IP cores and tools, locking developers into specific environments. Adapting to changes in these environments incurs significant overhead. This issue became evident during simulations of the current gateware and assessments of the effort required to port to new hardware platforms. To address this, open-source simulators have been evaluated and compared with proprietary ones, as detailed in the following chapters. Additionally, open-source tools facilitating multi-simulator support are discussed in Section 5.1.2. On the IP side, efforts have been made to design a collection of vendor-agnostic, open-source components to replace

proprietary equivalents, simplifying the porting process. These developments are presented in Section 5.3.

By adhering to these guidelines, the project aims to enhance code quality, increase test coverage, and improve portability. This will directly reduce complexity, labor, and debugging efforts, ultimately boosting the overall efficiency of the gateware development process.

5 Technologies and Methodologies for FPGA Gateware Design

FPGA designers are tasked with creating components that address specific functionalities, typically defined by a set of requirements outlined in the design specifications. An integral part of this process is verifying that the design operates correctly according to these specifications and ensuring that the verification process is thorough, testing all potential edge cases that the component might encounter in its real-world application. While these objectives may seem straightforward, they represent some of the most critical and challenging aspects of the development workflow. The Wilson Research Group Functional Verification Study 2022 [1] revealed that 50% of the project time on average is spent in verification to achieve satisfactory results, as shown in Figure 5.1.

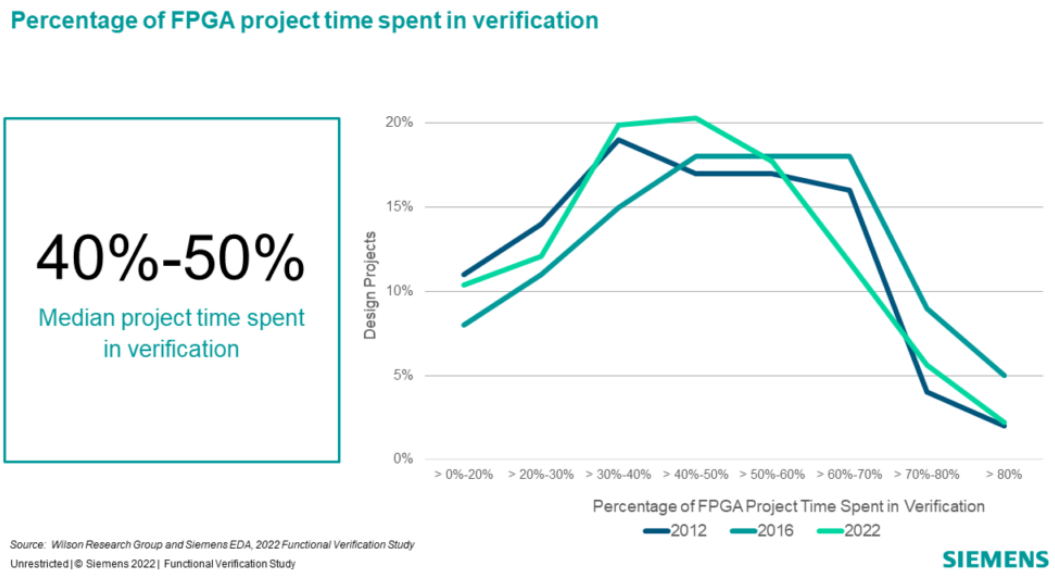


Figure 5.1: Percentage of FPGA project time spent in verification. [1]

Historically, the verification process was often less rigorous due to the smaller size and complexity of FPGA components. Designers would frequently rely on minimal testbenches and then deploy their designs directly to hardware, addressing issues only as they appeared. However, this approach is inherently flawed and does not scale effectively as FPGA designs grow in complexity.

Verification became a crucial step in ASIC development, where identifying and

resolving design issues during simulation could prevent costly post-fabrication fixes that might result in significant financial losses and prolonged delays. The techniques and tools developed for ASICs have since been adapted to FPGA workflows, where robust simulation strategies can substantially reduce the time spent on compilation and hardware validation.

Traditional methodologies for functional verification will be discussed in Section 5.1, along with guidelines to help developers establish effective verification plans, modern open-source verification frameworks will also be examined. Section 5.2 introduces formal methods, a collection of mathematical techniques for determining the correctness of designs, and evaluates their advantages and limitations when applied to FPGA workflows. The chapter also discusses the design aspect of gateware components, presenting in Section 5.3 a vendor-agnostic common core library that reduces verification effort by enabling the reuse of pre-verified IPs. Lastly, Section 5.4 explores the use of HLS tools in gateware development, highlighting their potential to streamline design workflows.

5.1 Verification by Simulation

Simulation remains the traditional and widely used method for verifying hardware design functionality. While simulations can provide insights into various aspects of a design, such as performance and power analysis, this section focuses specifically on functional verification.

Simulations can be divided in three different levels of abstraction, each targeting distinct applications:

- **High-Level Modeling:** Hardware designs can be modeled using high-level languages such as SystemC or C, enabling rapid and straightforward component development. Although not cycle-accurate, these simulations provide valuable insights into system architecture and are often used for testing embedded software and speeding up integration. High-level models also offer high simulation throughput, making debugging more rapid. Additionally, these models can be synthesized into hardware using HLS. However, current HLS tools have yet to reach the maturity required for widespread adoption. This topic will be discussed in detail in Section 5.4.

- **RTL Simulation:** The most commonly used abstraction for functional verification, RTL simulations are based on RTL designs written in Hardware Description Languages (HDLs). These designs can be directly synthesized into logic circuits, making them implementation-ready. RTL simulations provide *cycle-accurate* behavior tracking, accurately representing design functionality at every clock cycle. Although they do not model delays or other implementation-dependent processes, RTL simulations offer a good balance between accuracy and simulation speed, making them ideal for functional verification.

- **Gate-Level Simulation:** Using the design netlist with timing information, gate-level simulations provide the highest level of accuracy, including timing effects. These simulations are used for identifying issues such as incorrect timing constraints, asynchronous path bugs, and clock-domain crossing errors, as well as verifying timing closure. Given their implementation dependence and slower speed, gate-level simulations should be reserved for critical sub-components or debugging specific issues discovered during hardware deployment.

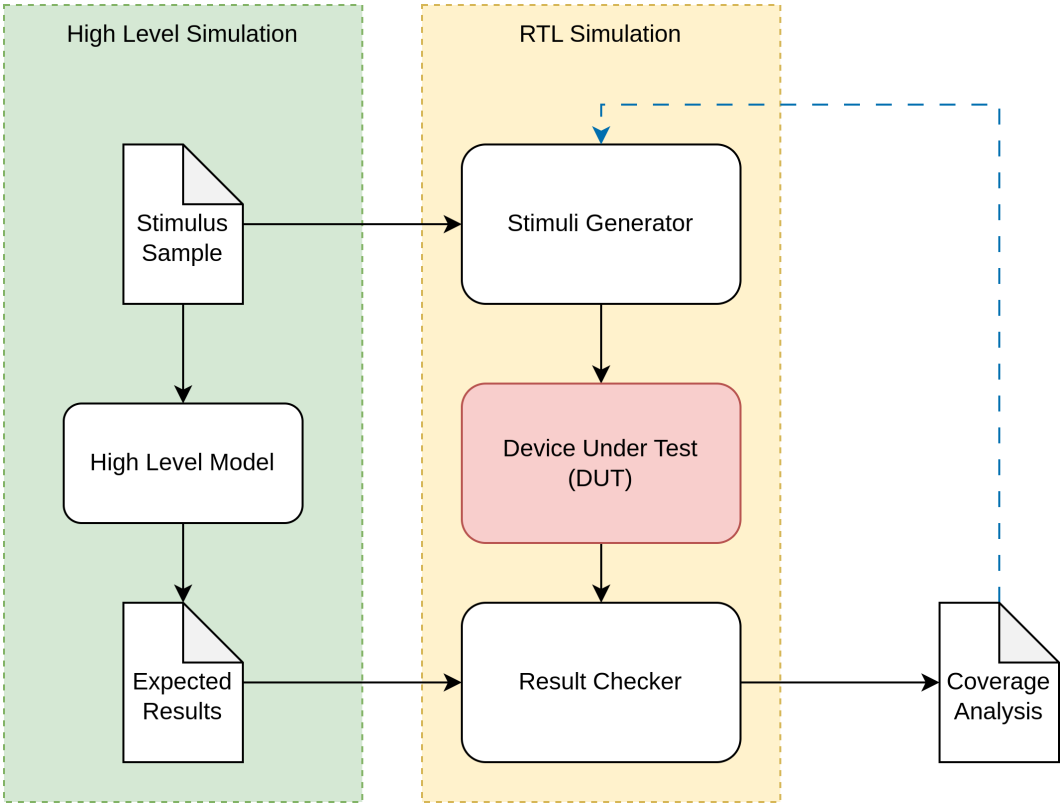


Figure 5.2: An example diagram of a functional verification testbench. The primary components of the simulation are highlighted in yellow. Stimuli generators and result checkers are often implemented in HDL code, while high-level simulation is typically implemented using languages like SystemC or C.

A typical testbench, as depicted in Figure 5.2, consists of a Device Under Test (DUT), which is the component to be verified, a stimuli generator to feed inputs to the DUT, and a checker to compare the DUT's outputs against expected values, thereby confirming its functionality.

For each simulation, the designer must assess the test coverage and create additional stimuli to address uncovered scenarios, particularly focusing on identifying and testing corner cases. Such cases are prone to hide bugs that might remain undetected until deployment, where they could cause undefined behavior or unrecoverable failures.

To tackle these challenges and reduce verification complexity, designers can follow a set of general principles:

- **Test Minimal Components:** Focus on verifying the smallest functional units possible. Testing isolated sub-components simplifies test case development and reduces complexity. Rigorous verification of these smaller units facilitates the subsequent verification of the complete design. Additionally, organizing tests based on specifications improves bug tracking and collaboration by making the tests easier to understand.

- **Cover Normal Operational Modes:** Ensure that the design performs as expected in all operational modes under normal conditions, as specified in the functional requirements. For larger or parameterizable designs, test all combinations of normal modes to verify the correct interaction between modules. The system should always return to an idle state after the tests.

- **Test Exception Conditions:** Validate the system's behavior under abnormal or exceptional conditions. This includes testing the system's recovery capabilities when operating outside normal modes. Tests should cover illegal conditions and protocol violations, ensuring the system can gracefully handle unexpected scenarios.

By adhering to these guidelines, designers can develop an effective verification plan tailored to the DUT's functionality and specifications, leveraging appropriate tools and methodologies to achieve validation closure. These guidelines have been tested and applied during the development of all the work presented in Chapter 6.

5.1.1 Test Coverage

Defining and testing all possible conditions in a design is a highly complex task, making it impractical to fully validate the system by covering every normal and corner case. To streamline this effort, a quantitative metric, known as *coverage*, is employed. Coverage directs developers' attention to untested portions of the design, avoiding redundant tests and ensuring a thorough validation process.

Coverage is expressed as a percentage of verification objectives met, providing a measure of verification progress. Two primary types of coverage metrics are commonly used:

- **Code Coverage**: This metric, derived directly from the code, indicates whether and how frequently specific lines of code are exercised during testing. While useful, it does not provide insights into the context or purpose of the exercised code.

- **Functional Coverage**: Defined by the designer, this metric ties verification to the design's intended functionality. It measures whether specific scenarios, corner cases, design requirements, and other critical conditions of the DUT have been observed, validated, and tested.

Functional coverage significantly enhances design quality and reduces verification time but requires upfront effort from designers to translate specifications into meaningful tests. This challenge is mitigated by leveraging established methodologies that provide standardized functions, metrics, and components to accelerate the development of high-quality tests.

Constrained Random Testing Directed testing involves manually crafting tests for each item in the test plan, which can become infeasible for even small designs due to the large number of tests required. This approach is neither portable nor reusable, leading to segmented, inefficient testbenches that are hard to maintain and trust.

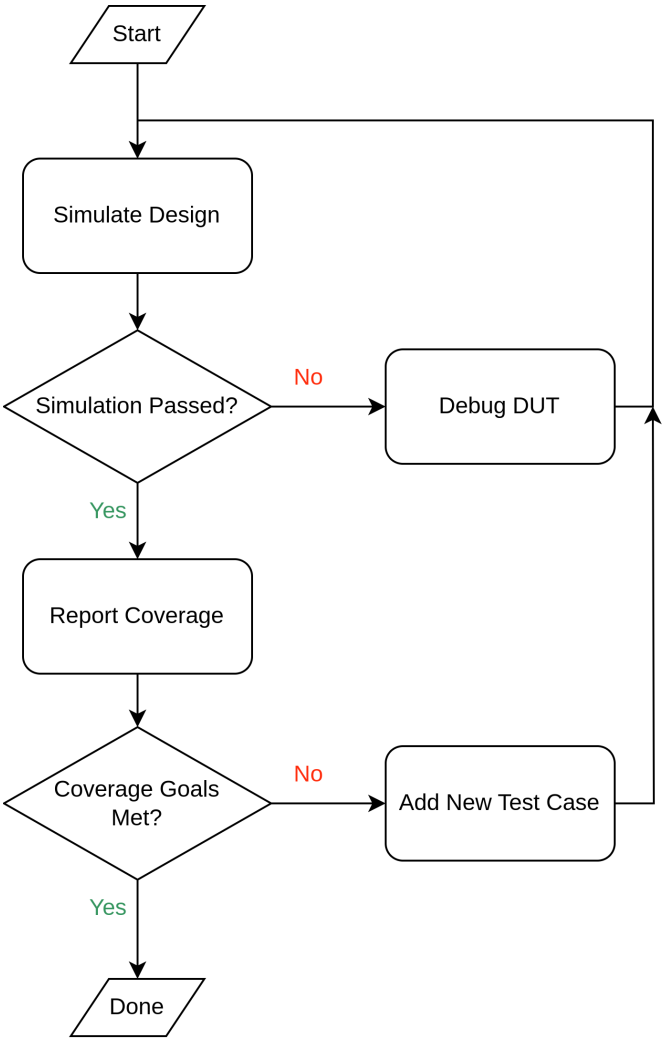


Figure 5.3: Flow chart of the constrained random test methodology.

A more efficient solution is the use of a pseudo-random stimuli generator, which automatically creates diverse test cases within the testbench without developer intervention. The testbench is also responsible for verifying coverage goals, using assertions

to validate the DUT's properties. The stimuli generation continues until all specified properties are tested and validated. The flow chart of this methodology is depicted in Figure 5.3.

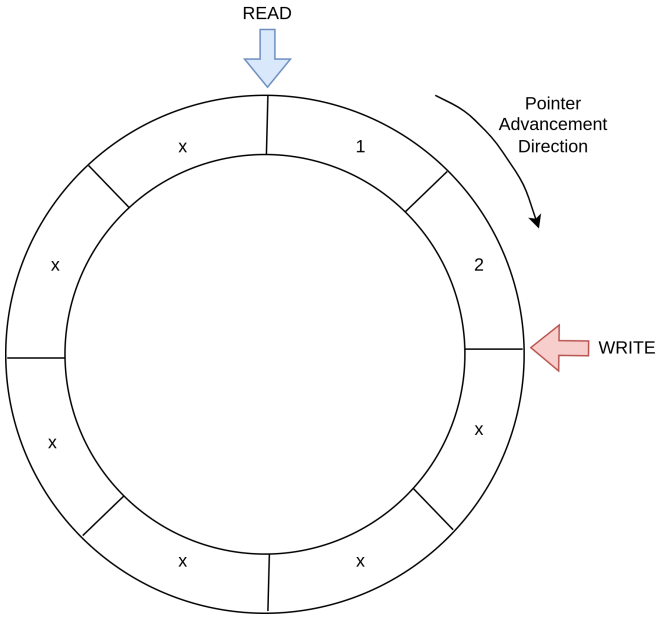


Figure 5.4: Diagram of the ring buffer FIFO. The red arrow indicates the write pointer, the blue arrow indicates the read pointer.

For example, consider a ring First-In First-Out (FIFO) buffer as the DUT. An explicative diagram of the DUT is shown in Figure 5.4. The pseudo-random stimuli generator randomly performs write and read transactions on the FIFO's interfaces. A simple constrained random testbench can validate the following properties:

- Normal-condition writes advance the write pointer.
- Normal-condition reads advance the read pointer.
- Writes to a full buffer do not advance the write pointer.
- Reads from an empty buffer do not advance the read pointer.
- When the buffer is full, the difference between the write and read pointers equals the buffer size.
- When the buffer is empty, the difference between the write and read pointers is zero.

These properties encapsulate all possible behaviors of the DUT in a high-level manner, abstracting the number of specific read or write operations required. The testbench generates enough random stimuli to ensure all properties are covered at least once, providing confidence in the DUT's functionality and pointer logic.

However, this approach does not verify the data path, for example ensuring data is correctly written to memory and read in order. This can be addressed using a scoreboard, a verification component that combines a FIFO's functionality with a checker. In the FIFO example, random stimuli data can be pushed into the scoreboard during valid write transactions and compared against read outputs during valid read transactions. This addition ensures the data path is thoroughly tested, verifying correct and ordered data handling. The constrained random testbench from which this example is derived is available as part of the work presented in Section 5.3.

Transaction Level Modeling FPGA designs typically involve component interactions over interfaces, with signals divided into data and control paths. The behavior of these interactions, defined by a *protocol*, consists of specific *transactions* that dictate component responses based on control signal combinations.

Abstracting these interactions into interfaces enables higher-level modeling, known as Transaction Level Modeling (TLM). TLM separates communication details from the implementation specifics of functional units, simplifying the representation of complex systems.

The FIFO example discussed above is already described in a TLM manner. In the natural language it is immediately clear that FIFO operations such as write and read involve a set of signals representing the data and a set of control signals that indicate to the FIFO control logic whether a write or read operation is requested, as illustrated in Figure 5.5. Details of the implementation of the interface architecture are separated from the communication operations, allowing a simpler description of an otherwise complex transaction based system.

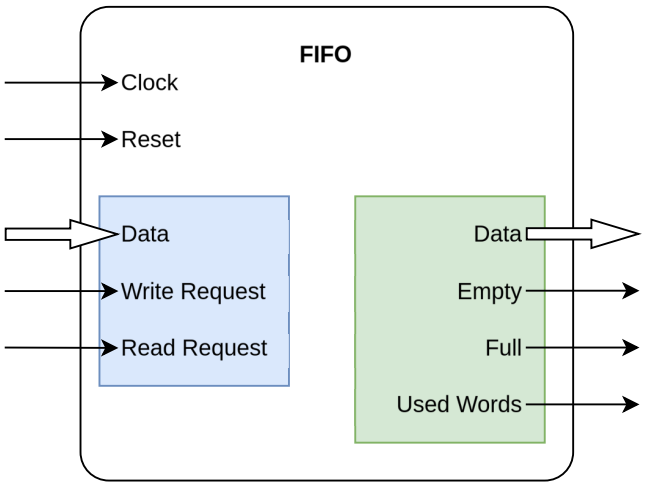


Figure 5.5: Block diagram of the FIFO. The blue rectangle shows the input signals which drive the transactions, while the green rectangle shows the output signals generated by the component.

Most FPGA designs commonly use two categories of data communication mechanisms:

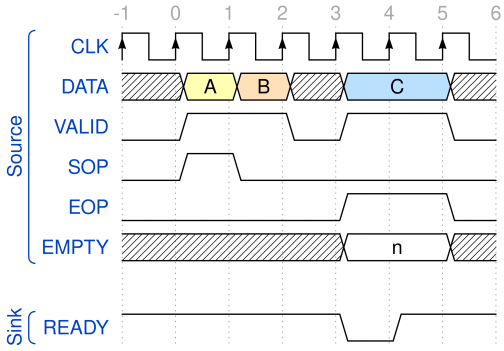
- **Streaming Interfaces**: these are point-to-point links where the transmitter is known as the source or master and the receiver as the sink or slave. Transactions on a streaming interface are defined by a basic *handshake* mechanism: The source drives the data together with a *valid* signal, which informs the sink that data on the link is valid and can be read from that clock cycle onwards. The sink can drive a *ready* signal to notify the source that it is prepared to receive the incoming data. Additional control signals can be used to transform the communication from a cycle-based approach to a packet-based one, where one packet is transmitted using multiple clock cycles.

Different standard streaming interfaces have been developed by FPGA vendors, the most common are AXI-Stream, used by AMD Xilinx, and Avalon Stream, used by Altera. A streaming transaction example is displayed in Figure 5.6a.

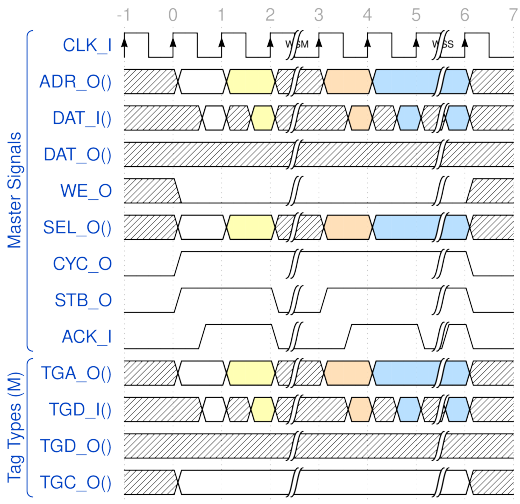
- **Memory-Mapped Interfaces**: Used for bus communication, these interfaces consist of a bidirectional data path, address path, and control signals. A host orchestrates transactions across multiple peripherals. The difference from a streaming interface is the presence of the address path which specifies the physical location of a particular register, memory, or resource. Moreover, the use of a bus allows sharing the same signals to access different peripherals, saving resources.

Basic memory-mapped interfaces allow read and write operations: both operations have to specify the address and the control signals correspondent to the operation. This simple functionality can become more complex when extended features are required, such as pipelined burst transfers or variable latency support.

Different standards have been developed and they can offer different extended functionalities. The most common protocols are AXI4 and AXI4-Lite, Avalon Memory-Mapped, and Wishbone Bus. A memory-mapped transaction example is displayed in Figure 5.6b.



(a) Example of an Avalon Streaming packet transaction. SOP and EOP signal the start and the end of the packet, respectively. EMPTY indicates how many symbols are valid in the last data word.



(b) Example of a Wishbone Block Read transaction. CYC_O high signals the transaction is happening. The operation spans over five clock cycles.

Verification frameworks often provide functional models supporting these standards, allowing developers to import models into testbenches and connect them to the appropriate interfaces. Testing then involves invoking model methods to generate required signals. These frameworks integrate seamlessly with constrained random testing, further simplifying and enhancing the verification process.

5.1.2 Frameworks for Verification

Effective verification testbenches share common traits, including the methodologies discussed earlier. To streamline their development, FPGA designers have consolidated these features into reusable frameworks. These frameworks abstract much of the complexity, reducing development effort and enabling designers to focus on the core verification tasks.

Different HDLs utilize distinct verification frameworks. For instance, SystemVerilog is the corresponding verification language for verilog-based designs. SystemVerilog has been further enhanced by the standardized Unified Verification Methodology (UVM) [42], which provides a robust collection of tools and libraries. These include constrained random generation, TLM, and many other features. UVM is widely supported by major vendors, facilitating the creation of robust, portable, and efficient testbenches.

VHDL-based designs rely on multiple open-source verification frameworks. While these frameworks offer similar functionalities, they often differ in implementation focus and methodology. It is up to the designer to choose the most suitable framework for their application. In some cases, a combination of frameworks may be used within the same testbench. The following sections present an overview of the features provided by these frameworks, along with recommendations for their use, based on the evaluation conducted during the development of the work discussed in Chapter 6.

Universal VHDL Verification Methodology (UVVM) [70] is a free and open-source framework for developing structured, reusable testbenches. Supported by industry and research institutions like the European Space Agency, UVVM is fully implemented in VHDL-2008, making it compatible with a wide range of simulators that support the language.

UVVM's utility library includes basic features such as checkers, but its standout feature is the VHDL Verification Components (VVCs). VVCs extend the TLM concept by providing advanced interfaces that handle both transaction scheduling and component implementation autonomously. This allows simultaneous activity on multiple interfaces orchestrated by a central sequencer. VVCs also support complex protocols, including packet-based streams and out-of-order communication.

UVVM provides a library of pre-built VVCs for common protocols and interfaces, and designers can develop custom VVCs to extend its functionality. However, the complexity of these components results in a steeper learning curve, particularly for customization. Numerous parameters must be configured to model specific behaviors, requiring an in-depth understanding of both UVVM and the interface being verified.

Open Source VHDL Verification Methodology (OSVVM) [58] is another free and open-source alternative to UVVM. Developed by contributors to the IEEE VHDL standard working group, OSVVM offers similar features, including verification components and utility libraries. Its key strengths lie in its functional coverage package and constrained random tools.

OSVVM provides an intuitive interface for pseudo-random generators, histograms, and coverage routines. It also features powerful scoreboards that integrate seamlessly into existing testbenches, enhancing their functionality.

While OSVVM supports advanced methodologies, some aspects, such as its Tcl-based scripting interface, may be seen as outdated. Tcl is commonly used in Electronic Design Automation (EDA) tools but lacks the usability of more modern scripting languages like Python. Transitioning to a more contemporary approach could improve the user experience.

VUnit VUnit [71] offers a modern approach to HDL verification, drawing inspiration from software testing frameworks. Like UVVM and OSVVM, VUnit provides libraries with checkers, loggers, and verification components. However, its primary differentiator is its Python-based scripting interface.

VUnit's test runner automates test discovery, file scanning, and compilation order management. It supports automatic incremental compilation and execution of tests, greatly simplifying test design and organization. Python scripting enables seamless simulator integration, enhancing test portability and simplifying testbench development.

The methodologies and testing frameworks discussed significantly enhance the verification process of FPGA projects, improving code quality and creating a more structured and efficient test harness. This, in turn, reduces debugging time when new bugs are discovered.

However, despite these advancements, they cannot address every possible scenario or behavior. Beyond simulation-based verification, an alternative approach can be applied to FPGA designs: formal methods. These will be explored in the next section.

5.2 Formal Verification

Formal Verification (FV) is a mathematical approach used to analyze the space of possible behaviors of a design. Formal methods can be applied to both software and hardware problems and can be utilized at various stages of the development process.

Similar to simulation, FV employs a set of tools to ensure that a design satisfies the properties and requirements of its specification. However, the way this is achieved is fundamentally different. FV tools explore the entire space of possible simulations rather than specific points. In other words, while simulation evaluates individual test cases, FV aims to cover the entire design space at once.

This does not mean that FV actually executes all possible simulations. Instead, it uses mathematical techniques to simplify and compute the behavior of the design

efficiently.

A notable historical example of FV's importance is the infamous Pentium Floating Point Division (FDIV) bug discovered in 1994. Intel's floating-point unit produced incorrect results for specific divisions. Simulation-based verification missed the test cases that would have exposed the issue before production, resulting in a USD $475 million recall. This incident highlighted the need for formal verification to prevent critical bugs from escaping to production.

Since the 1990s, FV techniques and tools have matured significantly, making them more accessible to FPGA engineers. However, many designers still hesitate to adopt FV, as it requires a shift from traditional point-to-point simulation to an abstract, mathematically-driven methodology.

FPGA designs are well-suited to FV because they are deterministic digital systems defined by boolean logic. Verification in this domain often involves solving Boolean Satisfiability Problems (SATs). In logic, a SAT is defined as a propositional logic formula that contains variables and logical operators. This formula is said to be satisfiable if there is a set of values assigned to its variables that results in the formula to be true. SAT is proven to be an NP[1]-complete decision problem, which means that the time required to solve the problem is at best exponential with respect to the input size. This means that a fully general proof for all conceivable cases can't be implemented. However, using a set of clever strategies can reduce the problem size significantly, making FV a viable approach. Figure 5.6 illustrates this situation.

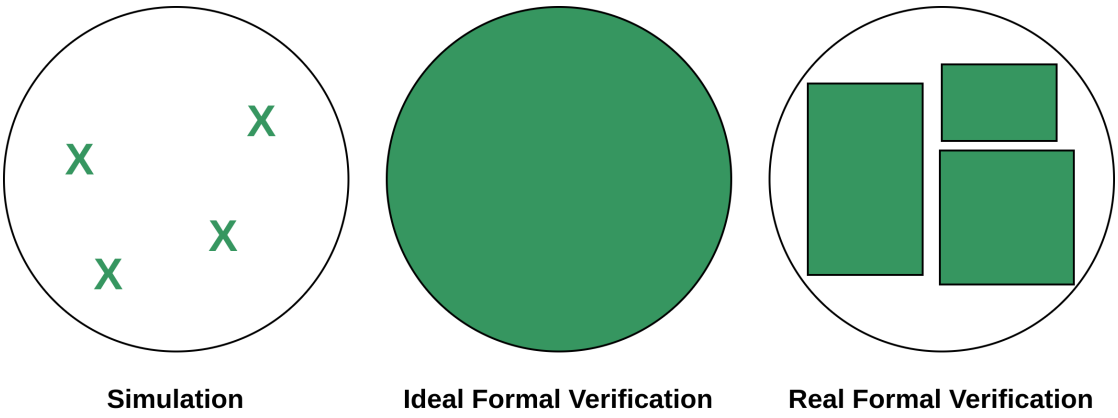


Figure 5.6: Coverage level of formal verification compared to simulation: simulation covers only some spots, ideally formal verification gives full coverage of the design space, but in practice full coverage can be achieved only in some areas.

Bounded Model Checking Bounded Model Checking (BMC) is the most widely used FV technique. In BMC, the design is represented as a finite-state system, and its properties are expressed in temporal logic. The tool translates the design into a SAT

[1]Non-deterministic Polinomial Time

expression that models all possible state transitions from the initial state. The user specifies a finite bound of steps, limiting the scope of verification to a manageable range.

A counterexample is found when a combination of variables falsifies the expression. If no counterexample exists within the specified bound, the tool guarantees that the design satisfies the property for all states within that range. It is up to the user to determine if the chosen bound is sufficient for verification closure.

Property Specification Languages To perform FV, properties of the design must be specified using Property Specification Languages (PSLs). These languages enable precise definitions of the expected behavior and the verification methodology. Properties can be categorized as:

- **Assertions**: Define the allowed behavior of the DUT.
- **Assumptions**: Specify the constraints on the inputs to the DUT.

PSL assertions are more expressive than those in simulation, as they can incorporate temporal logic to describe how conditions evolve over time. The most diffused PSL are SystemVerilog Assertions (SVA) and IEEE PSL [41]. While the first one only applies to verilog designs, IEEE PSL is a generic language that supports many different languages, including VHDL.

A Formal Verification Example Consider the FIFO ring buffer discussed in the sections above. The first step to cover is the reset. In BMC, the DUT should always start from a known state and this is usually the `reset` state. This condition can be expressed as an assumption, since it specifies the behavior of an input signal. In PSL, this can be expressed as:

```
assume reset;
```

After a reset, the property "the buffer should be empty" should be tested. This can be expressed as an assertion in this way:

```
assert always reset |=> empty;
```

This expression shows two constructs of PSL: `always` tells the tool that this property should be satisfied at each step and the `|=>` operator indicates a temporal dependency: its semantics is on each step i that `reset` is true, `empty` must be true in the step $i+1$. A more complex expression is used to model a valid write transaction. As said earlier, the property to prove is "during normal operation, a write transaction should increase the write pointer". This can be expressed as:

```
assert always (not reset and
               write_req and
               not read_req and
               not full) |=>
               (write_ptr = prev(write_ptr) + 1);
```

In the first part of the expression there is the boolean expression correspondent to a valid write operation, which requires the DUT to not be reset and full. In the consecutive expression, the assertion checks that the write pointer has been increased from the previous step. The `prev()` operator return the previous value.

A complete FV testbench for the FIFO can be implemented in fewer than 20 lines of PSL, whereas the equivalent constrained random simulation might require at least ten times as much code without guaranteeing comprehensive coverage.

Optimization Techniques In computational theory, FV is a computational problem that can be considered *NP-hard*. This translates in practice that the time and memory space required to compute the FV tests will grow exponentially with the size of the design. However, a set of techniques can be used to reduce the complexity of the problem and allow the tools to treat it more efficiently. The first and most important one is the reduction of the problem size. FPGA designs often involve multiple memories and data paths that are many bits wide. If no information is provided, the tool will try to model any possible combination of these bits, resulting in a explosion of possible states, impossible to deal with. Reducing data and memories to just a few bits usually will not change the functionality of the design, allowing the tool to catch the vast majority of bugs. If reducing data widths is not possible, it is possible to limit the search space to just some data patterns, such as most bits being 0.

Another useful technique is splitting the tests in smaller simpler tasks. This can be applied both to proofs, where complex properties can be decomposed in simpler independent ones, and to independent behaviors in the DUT. These techniques allow tools to manage complexity effectively, often enabling significant reductions in computation time and memory usage.

Applying Formal Verification FV proved to be a very effective approach to verification, leading to higher quality code and finding critical bugs that are difficult to find with conventional simulations. Its main technique, BMC, enables fast and efficient testing of components, automatically generating counterexamples when properties are violated. FV, however, is better suited for small designs where the possible states are limited, allowing the tools to compute the results in a reasonable amount of time and resources. Exploring multiple clock domains, large data paths, and large memories is not recommended. Even with these limitations, FV has proven its capabilities and it should be used to complement the traditional simulations.

5.3 Common Core Library

FPGA designs are composed of various components, commonly referred to as IPs, which work together to achieve the desired functionality. These components can generally be categorized as either technology-specific or generic.

Technology-specific components, often called hard IPs, are typically provided by FPGA manufacturers. These IPs implement high-speed protocols and interfaces such as PCIe and multi-gigabit Ethernet. They leverage specialized hardware features available on the device to accelerate repetitive tasks, including operations like scrambling, error detection, and error correction. By utilizing these IPs, designers can offload computationally intensive tasks, freeing up resources for other parts of their design.

However, these IPs are proprietary, with their source code encrypted, preventing designers from analyzing or modifying their internal workings. Designers must rely solely on the information provided in the manufacturer's datasheet. This closed-source approach is driven by the use of proprietary technologies protected as trade secrets, which would otherwise risk exposing valuable insights to competitors.

Generic components, on the other hand, provide widely applicable functionalities that can theoretically be implemented on any FPGA. These components are typically written in HDL languages and do not depend on specific technologies used by manufacturers to map them into hardware. Generic components are available from manufacturers, third-party vendors, and the broader FPGA development community.

Manufacturers and third-party vendors often distribute these components under licensing agreements, providing encrypted black boxes with defined interfaces. While this approach is similar to that of technology-specific IPs, the justification for encryption is weaker, as these components do not require the use of proprietary hardware features. Instead, such licensing practices primarily serve to lock developers into a vendor's ecosystem, limiting their flexibility and choice.

Given the widespread and essential nature of generic components, their functionalities are frequently required in multiple parts of a design. Consequently, many designers opt to develop their own implementations tailored to their specific needs, avoiding vendor lock-in. However, these custom implementations often lack thorough verification and proper documentation. While this approach mitigates dependency on proprietary solutions, it introduces its own challenges, such as reduced portability and collaboration.

Without a standardized approach, developers create personal implementations that may differ only in minor details from those of their peers. This results in a fragmented and inconsistent codebase, creating an ideal environment for bugs to proliferate and complicating long-term maintenance and integration.

With the rise of the internet and modern collaboration tools, developers have increasingly shared their efforts in designing FPGA components within the community. One of the most notable platforms for this purpose was OpenCores [57], where digital designers could publish source code for their developments along with version control systems for managing projects. Unfortunately, OpenCores now appears to be abandoned, prompting most designers to migrate to more popular platforms such as

GitHub to share their work.

While these platforms facilitate code sharing, they do not provide any assurances regarding the quality or reliability of the code. Additionally, the components are dispersed across numerous repositories, making it challenging to locate, maintain, and integrate them into larger projects effectively.

In response to these issues, some developers have attempted to consolidate these cores into libraries, adding features such as documentation, standardized styling, and basic automated verification pipelines. Although these efforts represent a significant improvement, they still fall short of providing the robust verification necessary for critical applications. As a result, these collections are not yet considered reliable enough for integration into critical designs, such as those used in projects at CERN.

As part of this thesis, the *colibri* library [24] has been developed to address these limitations, aiming to provide a standard components library for all CERN developers. Its logo is shown in Figure 5.7.

Figure 5.7: The logo of the colibri library.

Library Content The library consolidates the most commonly used components in DAQ applications, drawing inspiration from the selection used in the current PCIe40 gateware. The components are primarily generic and parameterizable to satisfy a wide range of user applications. As of this writing, *colibri* contains 57 components, organized into the following categories:

- **Common**: Entities and packages implementing frequently used functions, such as synchronizers, buffers, and debouncers.
- **Communication**: Entities designed for communication tasks, including scramblers and gearboxes.
- **Encoders**: Components for standard data encoding and decoding, such as Run-Length Encoding.
- **File I/O**: Functions and packages that simplify file operations during simulation.
- **Interfaces**: Components for translating and converting between different interfaces, such as Avalon and AXI.

- **Input/Output**: Basic low-level protocols (I^2C, SPI, UART) for interacting with external sensors and microcontrollers.
- **Memory**: Implementations of FIFO for single and dual clocks, along with RAM entities.
- **Packet**: Common operations for packet-based streaming interfaces.
- **Pipes**: Components for managing streaming interfaces, such as arbiters, broadcasters, and routers.
- **Protocols**: High-level protocols like Ethernet and Aurora. The Aurora implementation is described in detail in Section 6.2.1.

5.3.1 Design Philosophy

The development of this library is guided by three core principles:

- **Open-Source** The library provides its components, including source code, tests, and documentation, under the Weakly Reciprocal CERN Open Hardware License (CERN-OHL-W2). This licensing approach allows developers to use, modify, and contribute to the library, fostering a collaborative and robust community.
- **Vendor Agnosticism** To prevent vendor lock-in, the library is designed to be vendor-neutral, ensuring compatibility across a wide range of FPGA platforms. This flexibility enables developers to adapt and port designs seamlessly between platforms, offering greater freedom in hardware selection.
- **Full Verification** All components included in the library are subjected to rigorous validation and verification processes. By providing pre-verified components, the library builds developer confidence and facilitates seamless integration into new and existing projects.

These foundational principles ensure that the library meets the stringent requirements of FPGA designers working at or collaborating with CERN.

To implement these principles, the colibri development is standardized on a subset of the VHDL 2008 language. This choice balances expressivity and clarity while maintaining compatibility with a variety of FPGA toolchains, including AMD Vivado, Intel Quartus, and open-source tools. VHDL-2008's modern features significantly enhance the maintainability and readability of the codebase.

Verification and simulation are performed using open-source tools like NVC [35] and GHDL [36], which have been extensively tested with the colibri library. These tools, free from licensing restrictions, are used in the library's automated testing pipelines. Their compatibility with open-source testing frameworks further enables the application of state-of-the-art verification methodologies, ensuring that the library achieves the highest standards of reliability.

5.3.2 Validation and Verification

Validation and verification of components are central to the philosophy of the colibri library, requiring meticulous design of test harnesses and procedures. As outlined in previous sections, testing is categorized into two main approaches: simulation-based verification and formal verification.

Simulations Every component in the colibri library has an associated testbench, tailored to the complexity of the component being tested.

For simple components and VHDL functions, a single testbench file is sufficient. These testbenches generate randomized stimuli and verify the results against expected values, leveraging the utility and random functions provided by the OSVVM and UVVM frameworks.

For more complex components, the testbench structure is designed for greater portability and code reuse, following UVVM's guidelines. This design employs a modular architecture comprising three core elements:

- **Test Package** This package contains common constants, functions, and procedures utilized across the testbench. It also defines types and subtypes for custom interfaces. For instance, parameters such as clock periods and data widths are specified within this package.
- **Test Harness** The test harness is a non-synthesizable entity responsible for instantiating and interconnecting all components used in the tests. These components include the DUT and auxiliary entities such as TLM interfaces. Importantly, the harness entity is designed without ports, ensuring all signals sources and sinks are fully contained within the entity.
- **Test Bench** The test bench entity hosts the test procedure itself. It instantiates the test harness and controls its behavior through a single sequencer process. Commands are submitted and dispatched via the framework's internal communication network to the various TLM entities in the test harness.

This modular test architecture, depicted in Figure 5.8, ensures a clear separation of implementation details from the test procedure. High-level TLM commands are managed in the sequencer process, while low-level signal handling resides in a separate file. VVCs translate the commands into low-level signals and directly interface to the DUT. This approach enhances test clarity and facilitates reusability, as both the test harness and test package can be shared across multiple testbenches that implement different independent test scenarios.

Formal Verification In addition to simulation-based testing, many components are validated through a suite of FV tests. These tests are conducted using the open-source synthesis tool Yosys [73] with its FV extension SymbiYosys [72], while VHDL support is provided through the GHDL plugin. However, as of the time of writing,

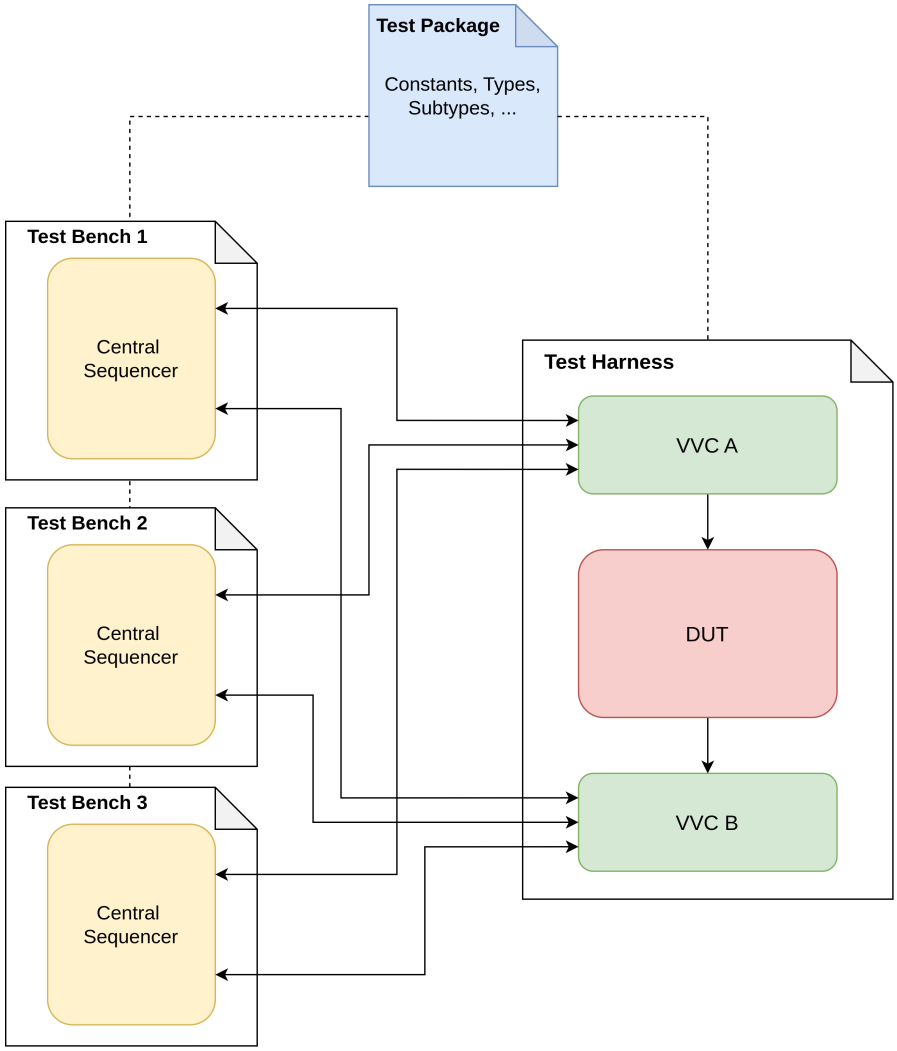


Figure 5.8: Modular testbench structure. This implementation allows the reuse of the test package and the harness across multiple testbenches. For instance, VVC A stimulates the DUT by generating the low-level signals from the sequencer commands. VVC B receives the output signals of the DUT and transmits them to the sequencer for checking.

these tools do not support multi-clock domain analysis using VHDL and IEEE PSL. Consequently, components with multiple clock domains are excluded from FV testing. Instead, their single-clock counterparts implementing equivalent logic are formally verified.

FV tests leverage the optimization techniques discussed in the previous section, such as reducing data widths and memory sizes for parametrizable components to simplify the verification process.

For more complex components that integrate standard interfaces like Avalon Stream, dedicated FV tests ensure protocol compliance. Verifying adherence to these protocols is considered sufficient to establish functionality. This approach confines any potential erroneous behavior to the failing component itself, enabling neighboring entities to monitor and correct errors effectively.

Continuous Integration The colibri library is hosted and maintained on GitLab [37], provided by the CERN IT department. GitLab's distributed version control and collaborative features streamline development. Among its capabilities, the automated pipeline system is integral to colibri's workflow.

The pipelines track every contribution and automate testing in three main stages:

- **Style Checking** Each contribution must adhere to a standardized VHDL coding style defined by the main developers. The open-source VHDL Style Guide (VSG) [45] tool enforces this style by automatically analyzing and, if necessary, correcting the code. While usually underestimated, style consistency ensures a homogeneous codebase, serves as implicit documentation, and simplifies the integration of library components into user projects.

- **Simulation** Automated testbenches are executed using VUnit, which is configured to generate JUnit XML outputs [43]. These outputs are compatible with GitLab's interface, providing clear feedback on test results, including error tracking and coverage analysis. This step is crucial for ensuring the functionality and reliability of the components.

- **Formal Verification** FV tests are conducted using the open-source toolchain described earlier. Although this toolchain lacks native GitLab-compatible reporting, it provides a simple pass/fail status.

For both simulation and FV stages, GitLab pipelines capture detailed test artifacts, such as logs and waveforms, in the event of test failures. These artifacts can be downloaded and analyzed for debugging without rerunning the tests.

The pipelines run on a distributed computing cluster provided by CERN IT, accessible to all FPGA developers at CERN. Each pipeline stage executes within a Docker container preloaded with the necessary tools and frameworks. Pipelines are triggered by new commits to the `devel` and `master` branches, as well as merge requests. To optimize resource usage, tests are executed selectively, only when relevant source or simulation files have been modified.

By automating continuous integration, the system ensures that developers and users always have up-to-date information about the library components. This transparency and reliability improves the community's trust in the project.

5.4 High Level Synthesis

HLS is a design methodology that enables developers to describe hardware designs using high-level programming languages like C and C++, rather than traditional hardware description languages such as Verilog and VHDL.

The primary advantage of HLS is its ability to decouple design specification from hardware implementation. This approach allows a single design to be configured for specific performance or power-efficiency goals directly by the compiler, without requiring modifications to the source code or user intervention.

HLS relies on specialized tools to automate the translation of high-level specifications into RTL representations that implement the desired functionality in hardware. Although introduced in the early 2000s, HLS has only gained significant traction in the past decade. This growth has been driven by several factors, including the exponential increase in silicon capacity, the rise of hardware-based compute accelerators, and advancements in HLS toolchains.

Despite these advancements, modern HLS tools still face notable limitations. They lack the abstraction level necessary for developers without hardware design experience to fully utilize them. At the same time, hardware engineers often find HLS tools restrictive due to their inability to specify critical hardware design features, such as bit-level accuracy, precise timing, and synchronization.

Additional challenges include the inferior quality of verification and debugging tools, particularly the lack of robust formal verification capabilities.

To address some of these challenges, Intel introduced the Intel OneAPI FPGA toolkit, aiming to bridge the gap between high-level design and hardware-specific requirements while improving usability and verification support.

Intel OneAPI FPGA Toolkit The Intel OneAPI FPGA Toolkit [33] is an integral part of the OneAPI framework [55], which is built on the SYCL language [68]. SYCL is a platform-agnostic abstraction layer based on standard C++17, providing a single-source programming model that allows developers to write code capable of targeting CPUs, GPUs, and FPGAs. By using templated functions to differentiate platform-specific implementations, SYCL offers flexibility and ease of use. Its similarity to NVIDIA's CUDA Runtime API further simplifies the transition for GPU developers to the OneAPI environment.

The toolkit is designed for compatibility across Intel's full product line, including CPUs, GPUs, and FPGAs. the toolkit also benefits from SYCL's open-source nature, which enables adaptation to devices from other vendors. With the FPGA Toolkit, OneAPI extends its capabilities to support the latest generation of Altera FPGAs, the Agilex 7 family.

The development cycle enabled by the FPGA Toolkit, depicted in Figure 5.9, is one of its key strengths. Developers can begin by sketching an algorithm in SYCL and testing it on the CPU to validate its basic functionality. Successively, the developer can translate the code to an FPGA compatible code by encapsulating the algorithm in a device routine, commonly known as *kernel*. This step should only require minimal modifications to the source.

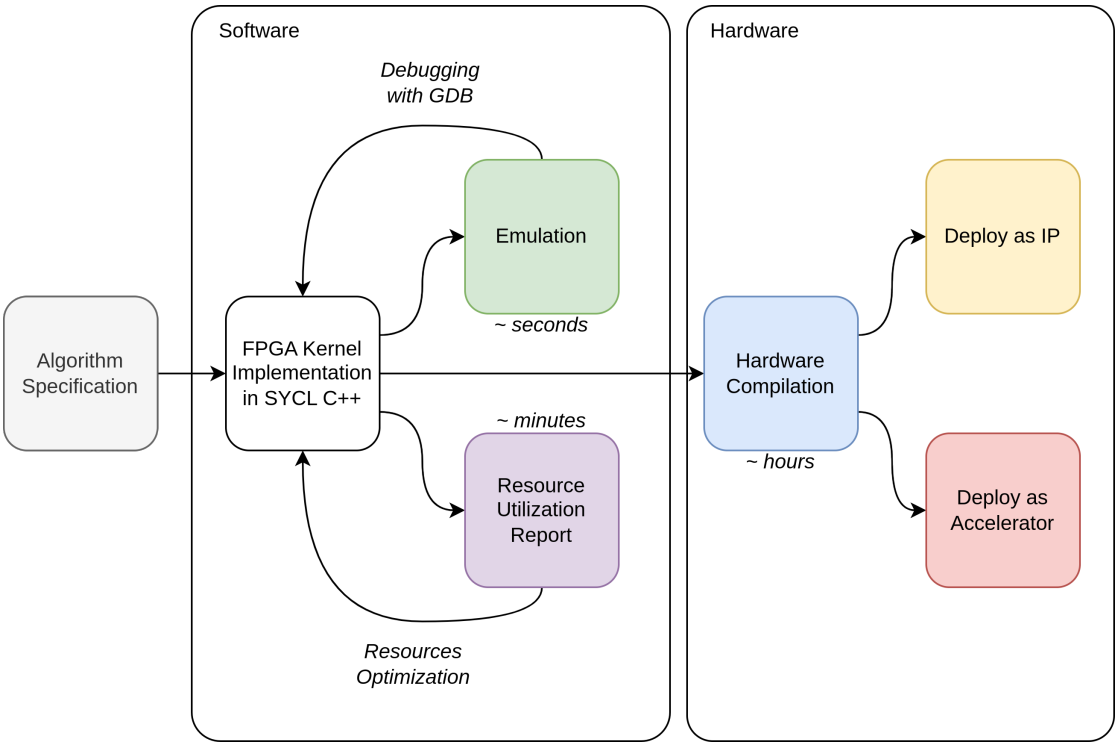


Figure 5.9: Overview of the Intel OneAPI FPGA Toolkit workflow. The process is divided into software-based steps, which do not require physical hardware, and hardware-specific steps, where a target device must be defined. Compilation times are indicated for each stage.

Before deploying to hardware, the FPGA Toolkit allows developers to emulate their designs on a CPU, which offers fast and effective debugging using familiar tools like GDB and Valgrind. Resource usage estimates can then be generated in minutes without requiring a full hardware compilation, enabling rapid feedback and early-stage optimization of the design. When the implementation is ready for hardware, a full compilation is performed, translating the design into a configuration for the FPGA. This process is the most time consuming, usually taking hours. Two hardware deployments options are available:

- **Acceleration Flow**: Produces a complete design where the FPGA serves as a compute accelerator for the host. This requires an FPGA with a PCIe interface connected to the host. The toolkit automatically configures the device and the software required to communicate with it.

- **IP Flow**: Packages the kernel into a reusable component that can be integrated in an existing FPGA design. The kernel design will require specific adaptations of its interfaces towards the rest of the design.

The final hardware implementation can be equipped with a performance profiling tool, allowing developers to monitor and benchmark the kernel directly on the FPGA.

However, despite its innovative approach and features, the development process is not as seamless as it initially appears. A detailed case study in Section 6.3 highlights some of the toolkit's critical challenges and limitations. While the Intel OneAPI FPGA Toolkit represents a significant step forward in HLS tools for FPGA development, it is not yet mature enough for widespread production use. Moreover, creating effective designs with the toolkit still requires prior understanding of FPGA design principles.

5.5 Highlights

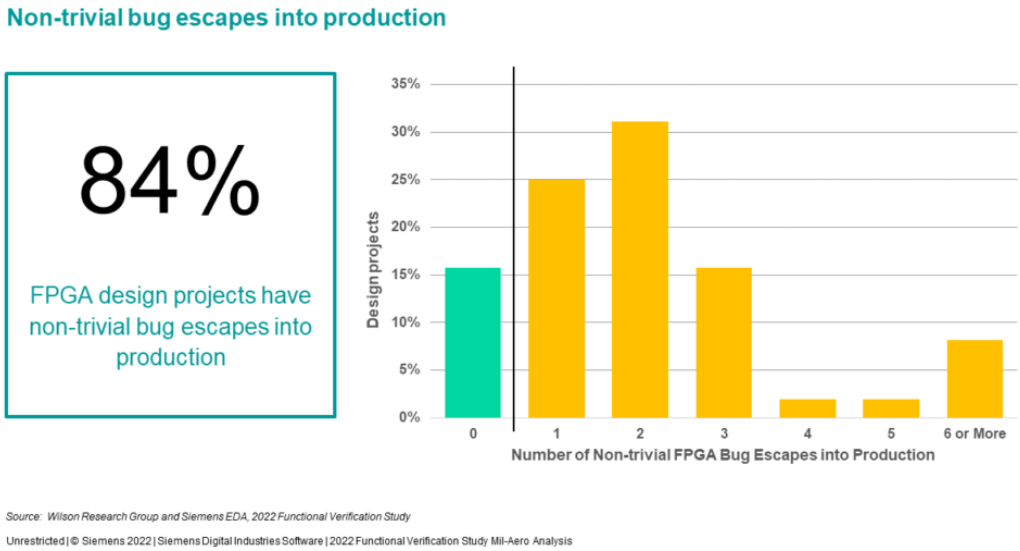
Over the past decade, FPGA designs have significantly increased in size and complexity due to technological advancements. However, this growth demands greater effort to ensure quality and verify functionality. The 2022 Wilson Research Group Functional Verification Study [1] revealed that 84% of FPGA projects contain undetected bugs that make it into production, as shown in Figure 5.10a. Additionally, Figure 5.10b suggests that projects with more mature verification methodologies tend to experience fewer bug escapes, emphasizing the importance of a rigorous verification process.

Despite its importance, verification remains a challenging and time-consuming task, often consuming over half of a designer's time. Techniques such as constrained random testing and functional coverage, along with open-source verification frameworks, can improve efficiency and effectiveness. The strengths and limitations of these frameworks were identified through the evaluation and tests presented in Chapter 6 and are summarized in Table 5.1.

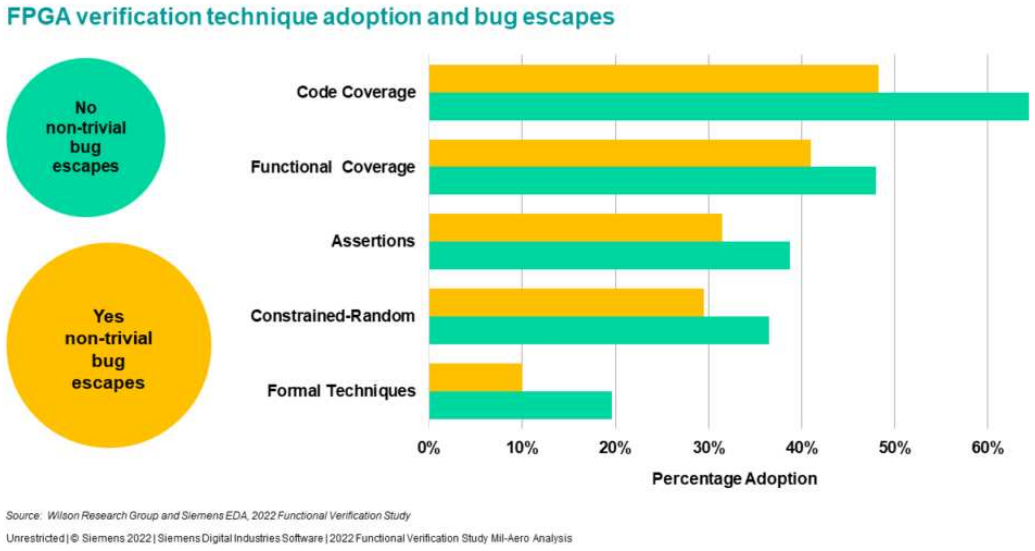
	UVVM	**OSVVM**	**Vunit**
Checkers and Loggers	yes	yes	yes
Verification Components	excellent	good	basic
Randomized Coverage	basic	excellent	not available
Scoreboards	good	excellent	not available
Scripting	not available	poor (Tcl)	excellent (Python)

Table 5.1: Comparison of key features among the three major open-source VHDL verification frameworks.

FV provides an alternative approach, significantly enhancing code quality by identifying complex bugs that traditional simulations may miss. However, due to current tool limitations, FV cannot yet replace simulation-based verification entirely. Instead, it is recommended as a supplementary technique.



(a) Percentage of FPGA projects with undetected bugs in production. A concerning 84% of projects are released with at least one bug.



(b) Impact of verification maturity on bug escapes, categorized by methodology. The data indicate that projects with well-structured verification processes experience fewer undetected bugs.

Figure 5.10: Analysis of FPGA verification effectiveness. (a) Occurrence of undetected bugs in production. (b) Relationship between verification maturity and bug escapes.

IP reuse can simplify verification and maintenance, but vendor-provided IPs often impose strict licensing agreements, limiting flexibility. Conversely, open-source IPs frequently lack sufficient verification and support, making integration challenging. The colibri library addresses these issues by offering a vendor-agnostic, fully verified, open-source collection of FPGA components. Thanks to these design principles, the library is already being adopted in CERN experiments, including LHCb, ALICE, and CMS.

HLS techniques have also been explored, enabling developers to describe designs using high-level languages while shifting hardware design complexities to automated tools. Intel OneAPI FPGA Toolkit represents a promising implementation, featuring a unique development cycle that integrates software debugging tools and fast compilation workflows. However, critical limitations identified during evaluation suggest that the toolkit is not yet mature enough for production use.

The following chapter presents case studies developed as part of this thesis, illustrating the practical applications, advantages, and challenges of each methodology.

6 Case Studies

This chapter presents a series of case studies that demonstrate the application of the techniques and methodologies outlined in previous sections. Section 6.1 explores the use of FV methods for gateware debugging and edge-case detection. Section 6.2 details the implementation of FastRICH decoding using the colibri library, along with the design of a prototype Ethernet-based readout system. Lastly, Section 6.3 examines the deployment of HLT algorithms utilizing the Intel oneAPI HLS toolkit.

6.1 Bug Finding with Formal Verification

This section presents an important case study where the application of formal verification (FV) techniques effectively isolated and corrected elusive corner-case behaviors.

During commissioning, sporadic data corruption was observed in events from the RICH Back-End (BE). The issue was traced to the readout gateware, but its rarity made isolation and reproduction extremely challenging. Traditional debugging relied on SignalTap, a logic analyzer integrated in the gateware, to capture incoming data. This method proved time-consuming, particularly for rare edge cases, and susceptible to false negatives in the presence of timing violations.

After weeks of investigation, the RICH and LHCb Online teams identified the bug. Fortunately, the issue surfaced during commissioning rather than data-taking, preventing potential data loss during beam time.

This scenario provided an ideal opportunity to evaluate FV tools. Initially, the work focused on the development FV test suites for the entire data processing block. However, the block's complexity, with multiple clock domains and extensive logic, exceeded the capabilities of current open-source FV tools. Consequently, the focus shifted to smaller, single-clock internal components.

The first target was the *compressor* block, which processes a packet-based Avalon Stream input split across four 85-bit channels. Its output is a 256-bit wide Avalon Stream with additional signals conveying BXID, packet type, and packet size. A parallel 64-bit stream carries TFC information. The block diagram is illustrated in Figure 6.1.

The first step involved defining the DUT specifications, treating it as a black box with known input and output formats. The Avalon Stream Protocol Specification was translated into IEEE PSL properties, such as:

- Packets start with a Start of Packet (SOP) signal and end with an End of Packet (EOP) signal.
- SOP and EOP are de-asserted during intermediate cycles.

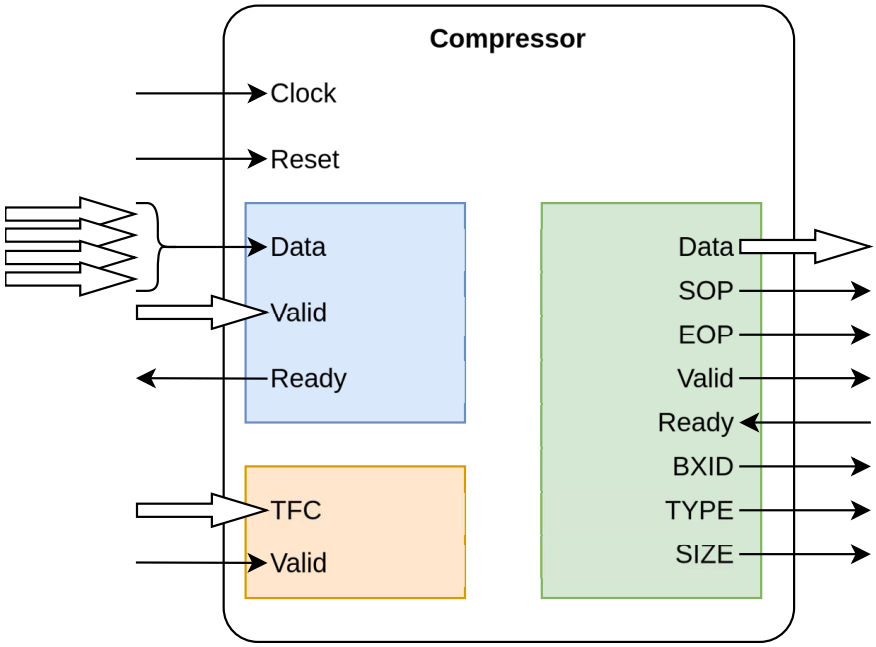


Figure 6.1: Block diagram of the RICH compressor block. In blue, the four 85 bit inputs with their correspondent control signals. In orange, the TFC information. In green, the packet Avalon Stream output with the extended signals.

- Valid signals can only de-assert after the ready signal is asserted.
- Data is captured when both valid and ready signals are asserted.
- Data and control signals remains unchanged if valid is asserted and ready is not.
- The empty signal, sampled at EOP, must be within the valid byte range.

The DUT uses a simplified input protocol, where fixed packet sizes eliminate the need for an empty signal. A Finite State Machine (FSM) manages protocol states and keeps track of transmission cycles. The properties are translated to PSL assumptions:

```
assume always (valid and ready) -> (sop or INTERMEDIATE);
assume always (valid and ready and eop) -> (words = FIXED_SIZE);
assume always (valid and ready and words = FIXED_SIZE) -> (eop);
assume always (valid and not ready) -> (valid);
assume always (valid and not ready) -> (data = prev(data));
```

For the TFC input, a fixed value is used and its valid signal is asserted when SOP and valid are asserted. This acurately creates a set of inputs to the DUT and guarantees they respect the specifications.

The output interface follows similar rules with additional PSL assertions for control signals and extra metadata:

```
assert always (valid and ready and IDLE) ->
              (sop);
assert always (valid and ready and INTERMEDIATE) ->
              (not sop);
assert always (valid and ready and sop and eop) ->
              (SINGLE_CYCLE);
assert always (valid and ready and sop and eop) ->
              (size <= DATA_BYTES_WIDTH);
assert always (valid and ready and not sop and eop) ->
              (size > N_WORDS*DATA_BYTES_WIDTH &&
               size <= (N_WORDS+1)*DATA_BYTES_WIDTH);
```

These assertions verify that single-word packets have appropriate sizes and multi-word packet sizes fall within valid ranges.

Modeling the specifications, while seemingly straightforward, requires time and practice, with numerous iterations to accurately express the properties in PSL. A common challenge for beginners is leaving signals uninitialized, which can lead to unexpected behaviors and false positives. Additionally, determining when the test harness is complete and whether it aligns with the specification can be difficult. Gaining experience with the tools and the language significantly reduces the time required to design an effective verification harness.

The FV tool successfully identified a counterexample, illustrated in Figure 6.2, that reproduced the issue found during traditional debugging: an output packet size exceeding the valid range, which caused downstream component failures. Developing and testing the FV specifications took only a few days, with the tool generating a counterexample within minutes.

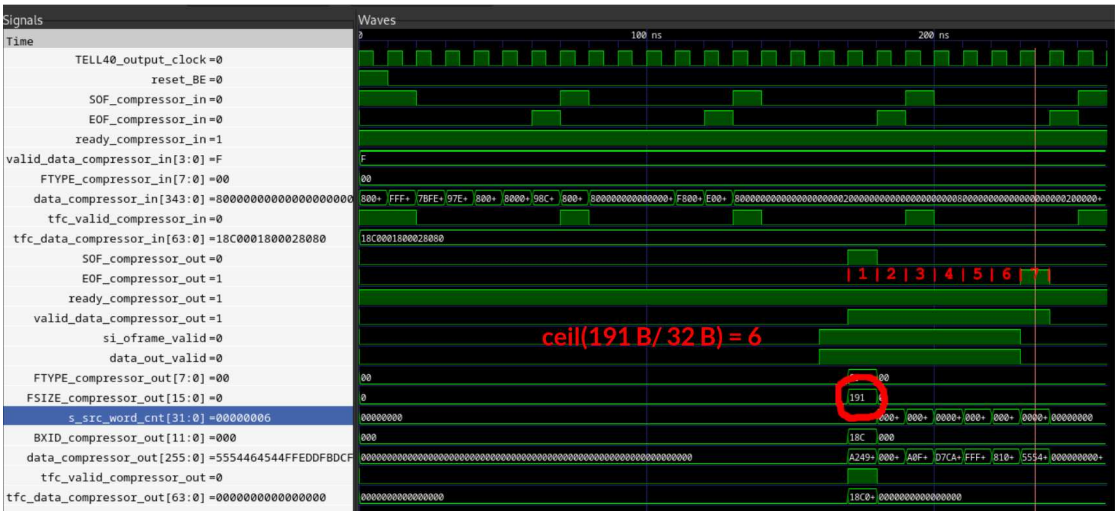


Figure 6.2: Counterexample waveform generated by the FV tool showing a mismatch between the number of words and the packet size.

6.2 Common Core Library Applications

As discussed in the previous chapters, the current FPGA codebase is populated with proprietary IPs, which require designers to use only supported proprietary tools. Additionally, there are numerous handmade IP variations that implement the same functionality but lack proper verification, leading to codebase fragmentation.

The *colibri* library was created as part of this thesis to address some of these issues by offering open-source, fully verified, vendor-agnostic components. In this section, two important case studies are presented where the library plays a key role. The case study in Section 6.2.1 covers the implementation of the Run 4 RICH FEE ASIC protocol decoding. The case study in Section 6.2.2 presents a proof of concept for future readout systems based on Ethernet.

6.2.1 FastRICH ASIC Decoding

The FastRICH [44] is a readout chip designed for the upgrade of the LHCb RICH detectors during LS3. The chip is compatible with the current multi-anode photomultiplier tubes, which will be used throughout Run 4, while also supporting silicon photomultipliers being evaluated for Run 5.

This new chip allows for accurate timestamping of Cherenkov photons with a precision of 25 ps, facilitating the reconstruction process. However, this generates a large volume of data. To manage this, the FastRICH uses a highly optimized data-driven packet format with zero suppression and relies on the Aurora protocol [13] for framing and lpGBT for the physical link.

Using Aurora enables the FastRICH to dynamically configure the number of output links, allowing the ASIC to adapt to the occupancy of different subdetector regions. Consequently, the corresponding decoding of the Aurora protocol in the BE FPGAs was developed. The FastRICH uses a subset of the Aurora protocol, specifically the simplex version with 64b66b encoding. Therefore, the BE decoder will only implement this part. Moreover, the electrical specifications and the Physical Medium Attachment (PMA) are unnecessary since the protocol is encapsulated within lpGBT. The work focuses solely on implementing the Physical Coding Sublayer (PCS) for both the encoder and the decoder, in order to simplify testing and emulation.

Aurora Encoding and Decoding The Aurora protocol defines the transfer of user data across a *channel*, consisting of one or more *lanes*. Each lane is a serial data connection. Aurora channels have the following features:

- Data is transferred in frames.
- Data frames and control words share the same channel.
- The protocol specifies only the delineation of frames, with no constraints on format or length.

- Frames can be interrupted by control words.

Figure 6.3 illustrates the connection between the chip, which acts as the transmitter, and the BE, which acts as the receiver.

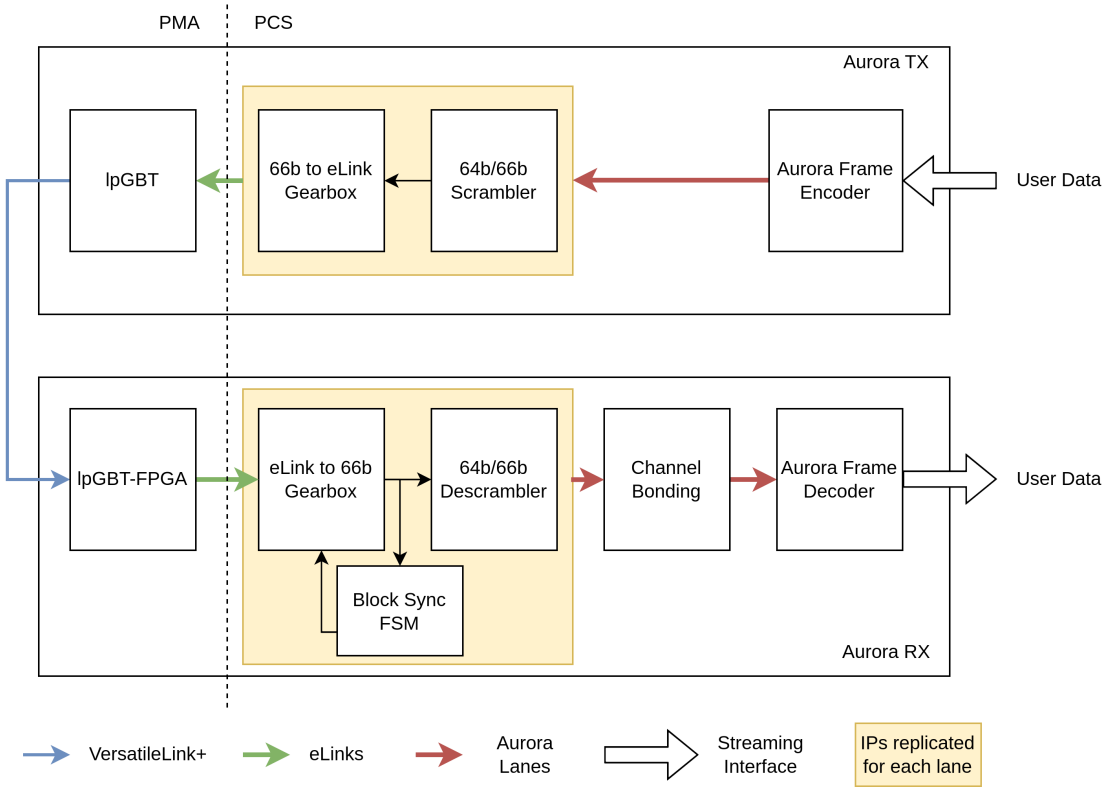


Figure 6.3: Connection between the FastRICH and the BE. The main components of the Aurora protocol are highlighted. IP in the yellow section are replicated for each lane.

Following the data path, user data is formatted in the user-defined format and uses a user-chosen interface. In the case of *colibri* components, this is a packet-based Avalon Streaming Interface with a width of 64 bits. The first component encountered is the **Aurora Frame Encoder**, which converts data words into Aurora *blocks*. Each Aurora block consists of a 64-bit word and two additional bits used for synchronization and distinguishing between data and control blocks. In data blocks, these two bits are encoded as `0b01`, while control blocks start with `0b10`. The values `0b11` and `0b00` are not allowed.

The remaining 64 bits encode ten different block types, but only three are used in this simplex version. In order of priority, these blocks are illustrated in Figure 6.4:

- **Channel Bonding**: A control block sent every n cycles, used by the receiver's channel bonding component to align multiple lanes. This block starts with `0x27840`. The channel bonding interval is user-configurable, with a default of 64 cycles.

- **Data, Separator, Separator-7**: These blocks contain user data. Data frames start with `0b01`, with all 64 bits as user data. Separator and Separator-7 blocks delimit frames, starting with `0b10`. Separator blocks begin with `0x1E`, and the second byte indicates how many of the remaining six bytes contain valid data. Separator-7 blocks start with `0xE1`, with all remaining bytes containing valid data.

- **Idle**: The lowest priority control block, sent only when no other data is available. Idle blocks maintain synchronization between transmitter and receiver when no data is being transferred. They start with `0x27800`. Typically, the transmitter sends a predefined number of idle blocks before assuming the link is ready for data transmission.

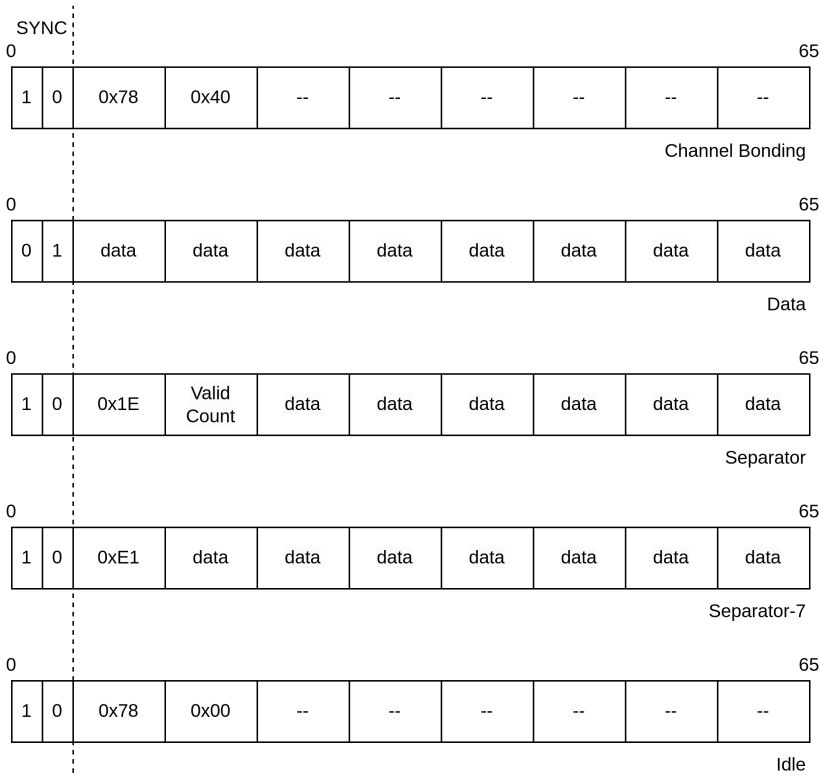


Figure 6.4: Types of Aurora blocks used in simplex communications.

Encoded blocks are then forwarded to the **64/66 Scrambler**, which encodes the 64-bit part of the block into a new sequence using a pseudo-random algorithm. Scrambling helps with clock recovery, synchronization, DC balance, and reduces inter-lane interference. The scrambling algorithm, defined in IEEE 802.3ae [40], uses an LFSR with the polynomial:

$$G(x) = 1 + x^{39} + x^{58} \tag{6.1}$$

The coefficients of the polynomial 6.1 represents the position of the Flip-Flops (FF) taps which are added to the input using XOR operators, as shown in Figure 6.5.

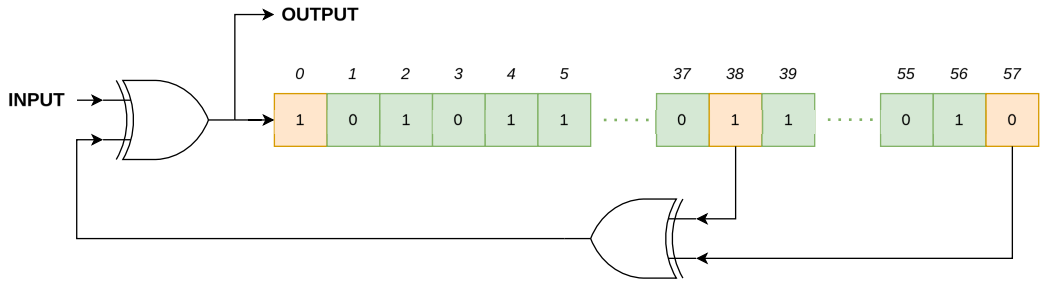


Figure 6.5: A diagram of the LFSR-based scrambler used in IEEE 802.3ae. The FF chain length is defined by the degree of the equivalent polynomial. The taps are defined by the coefficients of the polynomial.

. The final step in the PCS is the **66b to eLink Gearbox**, which converts the 66-bit Aurora word to a user-defined width. For FastRICH, the chip uses the lpGBT eLinks, which are independent links of configurable size and bit rate. Possible eLinks widths are 8, 16, and 32 bits.

The scrambler and gearbox are replicated for each Aurora lane, with each lane corresponding to one eLink. These eLink streams are then fed to the lpGBT for encapsulation and transmission over the Versatile Link+.

On the receiver side, the lpGBT link is decoded by the lpGBT-FPGA IP provided by CERN's Microelectronics group (EP-ESE). Once decoded, each eLink is fed to a gearbox that reconstructs the 66-bit Aurora words. However, a simple gearbox is insufficient because serialization causes the loss of bit position information, misaligning the reconstructed word boundaries and preventing further decoding.

To recover the correct bit alignment, the gearbox is equipped with a `slip` signal. When pulsed, this signal shifts the gearbox output by one cycle, adjusting the word boundaries. The slip signal is controlled by the **Block Sync FSM**, which implements the state machine described in IEEE 802.3ae.

The FSM examines the SYNC header to verify if it holds an allowed value (`0b01` or `0b10`). This verification is repeated over a configurable number k of words, counting valid and invalid headers. If all headers are valid, the test continues on the next k words. If invalid headers are detected, the system triggers the gearbox slip signal. After a sufficient amount of cycles, the system is capable of aligning the incoming words automatically and keeping the alignment.

Once the words are aligned, they are fed into the **64b/66b Descrambler**, which reverses the scrambling process performed by the transmitter-side scrambler, restoring the original user data. The descrambler is implemented using an LFSR based on the same polynomial defined in Equation 6.1. These components are replicated for each independent lane.

If multiple lanes are used, an additional **Channel Bonding** component is required. Since each lane operates independently, FastRICH allows lanes to be transmitted over separate lpGBT links. Thus, de-skewing is necessary to reconstruct frames successfully.

The channel bonding process involves a set of FIFOs, one for each lane, which buffer

a configurable number of Aurora words. An FSM reads each FIFO until it detects the first channel bonding block, then pauses reading from that FIFO and continues with the others. This process repeats until all lanes are aligned at the channel bonding block. Once synchronized, lanes can be read in parallel. The FSM continuously monitors channel bonding blocks and resets the system if alignment is lost.

After synchronization, the **Aurora Frame Decoder** reconstructs the frames, removing the SYNC headers and control information. The frames are then transmitted to the user logic through a streaming interface, such as the packet-based Avalon Streaming Interface.

Functional Verification The components described above have been developed together with matching testbenches to ensure correct functionality. In particular, blocks that are common throughout different protocols, such as the scramblers and the gearboxes, have been designed to be as versatile as possible to help with reusability. Since most components implement some processing which is then reverted, the testbench approach instantiates the mirrored components connected back-to-back in a loopback. This design simplifies the development of the behavioral models and streamlines the test harnesses.

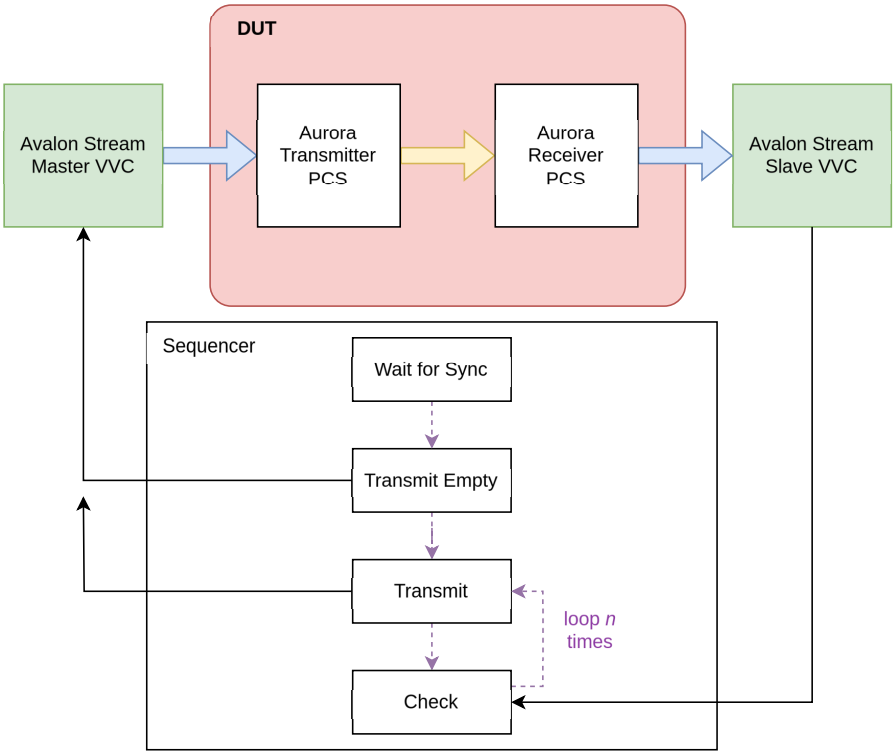


Figure 6.6: Diagram of the Aurora testbench. Transmitter and Receiver are connected in loopback, indicated with the yellow arrow. Avalon interfaces are connected to the VVCs.

For instance, in the main test harness, the transmitter PCS and the receiver PCS are

connected in a loopback, as depicted in Figure 6.6. The input to the transmitter is driven by the UVVM Avalon Stream Master VVC, while the receiver output is connected to the Avalon Stream Slave VVC. The test harness allows the configuration of the number and width of lanes for DUT using generic ports. The test bench sequencer proceeds as follows:

1. It waits for an amount of time that is sufficient to allow the DUT to synchronize and lock the link.
2. It sends a first packet through the transmitter VVC that will be discarded by the receiver. This is required by the decoder to align the framing window.
3. A user-defined set of packets with random sizes are sent by the transmitter VVC.
4. The receiver VVC then checks that the packets received match the ones sent.

With a sufficient number of packets, it is possible to test many different cases, indicating the level of coverage. Moreover, the test harness allows setting delays on the different aurora lanes to check the functionality of the channel bonding component. The tests are run using the VUnit framework, which allows easy scripting for the generation of test cases for each possible lane number and width. VUnit is able to directly set the generic ports in the VHDL code and automatically parallelizes the execution.

With this setup, it was possible to perform testing both during development and deployment, streamlining debugging. Moreover, it simplifies the addition of new test cases thanks to its modular approach.

Vendor Portability At the time of development, there was no PCIe400 board available, which is the target board for the FastRICH BE. Therefore, the Aurora decoder IP had to be designed in a way that allowed early testing on the available hardware, which included the PCIe40 board and multiple AMD Xilinx Ultrascale+ Development Kits. As a result, the design needed to be portable across different vendors and adaptable to multiple devices with varying specifications.

The first tests involved integrating the Aurora encoder and decoder into the test gateware for the PCIe40. In this configuration, the system operated in loopback mode, directly connecting the encoder to the decoder over a GBT link. The inclusion of Aurora in the loopback was seamless, with no errors or failures observed. However, lpGBT functionality could not be tested at this stage, as it had not yet been ported to the PCIe40.

The second set of tests involved a system composed of two FPGAs:

- **FEE Emulator**: This device, based on the Zynq Ultrascale+ ZCU102 Development Kit, generates a stream of detector data from a file preloaded in memory, encodes the data using Aurora, and transmits it over five lpGBT links. This design was developed by Mitja Vodnik.

- **Back-End**: This device, based on the Opal Kelly XEM8320 Artix Ultrascale+ Development Kit, receives the lpGBT links, decodes the Aurora-encoded data, and forwards it to a PC over a 10 Gigabit Ethernet (10GbE) link. This design is part of the work described in Section 6.2.2.

The tests confirmed successful data transmission, reception, and decoding. The system is now considered ready for integration with the FastRICH once the chip and its specifications become available.

Porting the Aurora decoder and encoder IPs between vendors did not require any modifications to the source code, demonstrating the vendor-agnostic nature of the *colibri* library. Both IPs are supported by comprehensive automated constrained random testbenches, which validate every possible FastRICH configuration to ensure the system meets its intended performance requirements.

Additionally, a resource utilization analysis was conducted for both the Altera Arria 10GX used in the PCIe40 and a Zynq Ultrascale+ device. This analysis provides valuable insights for BE developers and FE designers, helping to optimize detector layouts, determine the number of required links, and estimate the FPGA resources needed for data readout. Figure 6.7 illustrates the Look-Up Table (LUT) utilization for all possible configurations of a single decoder instance, while Figure 6.8 shows the total resource utilization in the worst case scenario as the number of lpGBT links varies. The worst-case scenario considered assumes a single-lane Aurora decoder with an eLink width of 8 bits, resulting in 28 decoder instances per lpGBT link.

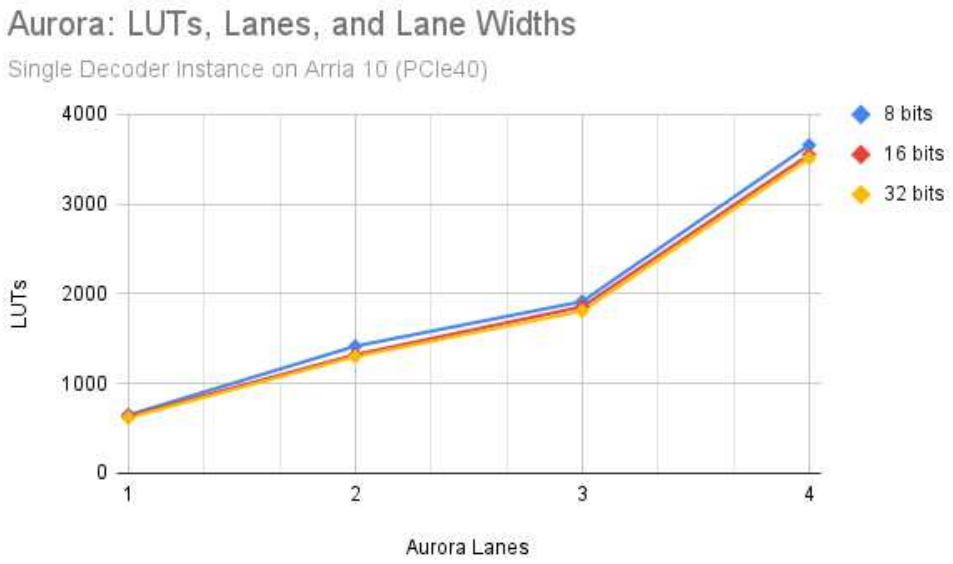
6.2.2 lpGBT to Ethernet Media Converter

This section presents a PoC of a cost effective and power efficient readout system for future HEP experiments.

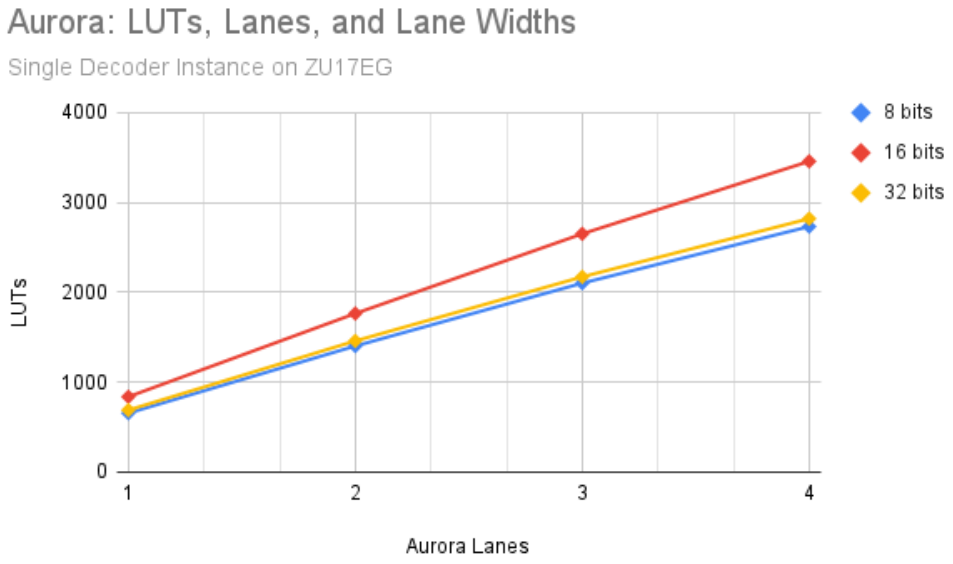
The FPGAs currently in use in HEP experiments are equipped with many transceivers in order to process and aggreggate a sufficient number of detector links. As discussed in section 4.2, the current hardware can not be replicated due to component obsolescence, thus making it impossible to make new boards, and its capabilities are limited compared to the requirements of future detectors. Consequently, efforts are underway to develop new boards with upgraded FPGAs. However, state-of-the-art FPGAs are trending towards fewer transceivers with higher data bandwidths [7], and features optimized for artificial intelligence, which have limited applications in DAQ systems.

In contrast, FEEs prioritize radiation hardness and low power consumption, resulting in high number of low-bandwidth links. This discrepancy leads to underutilization of high-end FPGA capabilities, diminishing cost-effectiveness. As part of the work on future back-end architectures, this case study explores an alternative DAQ architecture emphasizing cost efficiency and evaluates the feasibility of lower-end Ethernet-based FPGA readout boards for this purpose.

Design Choices The proposed design [61], named *NetGBT*, is based on the following principles:

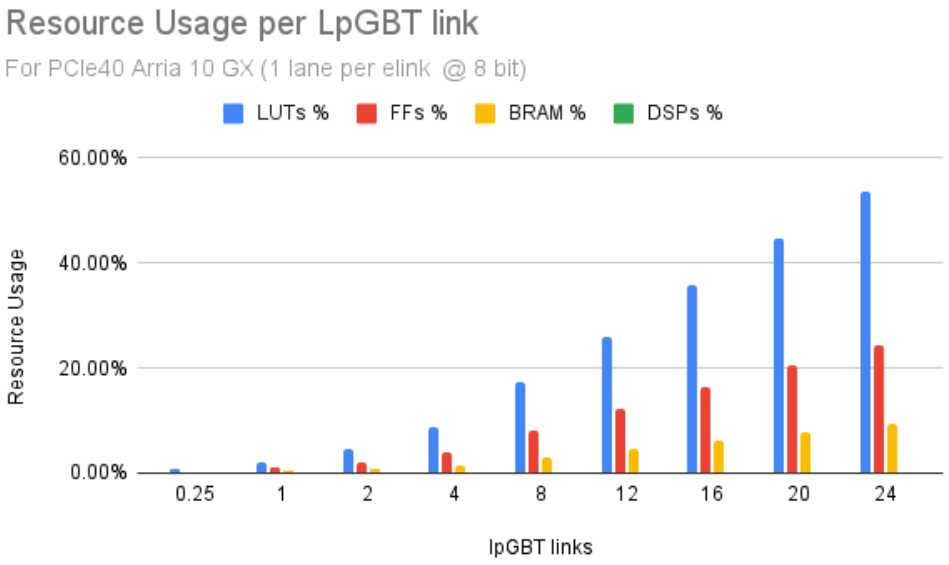


(a)

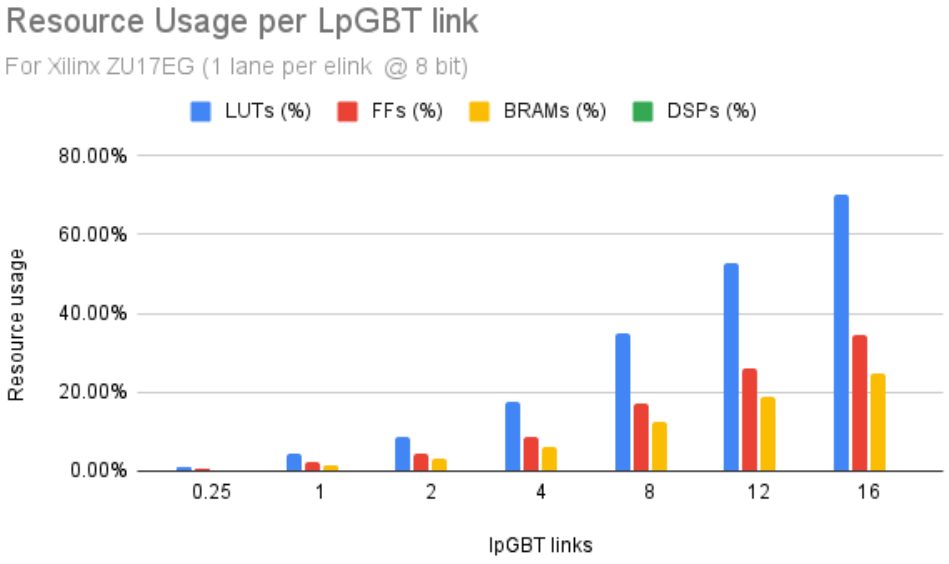


(b)

Figure 6.7: LUT utilization scan for all possible configurations of a single decoder instance on a PCIe40 (a) and a Zynq Ultrascale+ (b). The higher LUT utilization in (b) is caused by the technology mapping of the compiler on the FPGA. Optimized code should reduce this effect.



(a)



(b)

Figure 6.8: Resource utilization scan of multiple single-lane, 8-bit-wide decoders on a PCIe40 (a) and a Zynq Ultrascale+ (b). Each lpGBT link requires 28 decoders.

1. Direct conversion of custom radiation-hard protols, such as lpGBT, into standard Ethernet-based network protocols.
2. FPGA selection optimized for price-to-transceiver ratio, favoring devices with numerous transceivers for efficient aggregation.
3. Integration of hard IP support for 10 GbE, with preference for hardware capable of 100 GbE.
4. Adequate resources for decoding, aggregating, and packaging current and future FEE data formats.

Compared to the current architecture, the system leverages Ethernet instead of the PCIe interface. This choice makes the solution highly flexible, supporting a wide range of applications, from test beams and small setups where it can be directly connected to a host through a Network Interface Card (NIC), to large scale deployments where high levels of aggregation can be obtained by employing COTS Ethernet switches. Furthermore, the presented design is less dependent on space and power constraints dictated by the host server . The network stack employs Internet Protocol version 4 (IPv4) and User Datagram Protocol (UDP)-Lite protocols, both universally supported by COTS devices.

IPv4 facilitates packet addressing and routing, while UDP-Lite offers a simple message protocol without built-in Cyclic Redundancy Checking (CRC), simplifying gateware design and streamlining data processing at the receiver.

Proof of Concept The PoC is designed around an COTS development kit based on the AMD Xilinx Artix Ultrascale+ AU25P FPGA [56]. This FPGA offers 12 transceivers, named GTYs, capable of speeds up to 16.3 Gbps. The development kit breakouts the transceivers through two SYZYGY XCVR connectors hosting 4 transceivers each, two SFP+ cages, and the remaining two using SMA connectors. A mezzanine [69] with a QSFP+ connector is mounted on one of the SYZYGY XCVR connectors and it is used to implement up to 4 10GbE links. Another mezzanine can be used to receive 4 lpGBT links on the other SYZYGY XCVR port. Additionally, a 1 Gigabit Ethernet (1GbE) link connected to one of the SFP+ cages is used for board management and configuration. Unfortunately, the Artix line-up does not offer 25 Gbps capable transceiver, so it is not possible to use the board to aggregate multiple links into a single 100 Gigabit Ethernet (100GbE) link.

In the gateware illustrated in Figure 6.9, an lpGBT link is received using the provided transciever hard IP and decoded using the lpGBT-FPGA IP provided by the CERN Microelectronics group (EP-ESE). In this PoC, the lpGBT link is configured for its highest data rate of 10.24 Gbps with FEC5 error correction mechanism. After decoding, the lpGBT core outputs a 224-bit wide word every 40 MHz, yielding in an effective throughput of 8.96 Gbps. The stream is packeted using a mixed-width dual clock FIFO which stores the incoming data, convert the stream size to 64 bits and clock to 156.25 MHz, to match the upcoming network IPs.

An FSM manages the FIFO reads to create large packets containing multiple lpGBT words. The packet size has to be sufficiently large to ensure efficient use of the network bandwidth. Once the packets are formed, they are forwarded to the network stack, which appends all the necessary headers for UDP, IPv4, and Media Access Control (MAC) address. The packets are then processed by the proprietary 10G/25G Ethernet Subsystem IP and sent on the wire towards the host.

The system can be configured through registers exposed over JTAG and via a proprietary microcontroller integrated in the gateware which manages the 1GbE interface. Configuration over the network uses Message Queue Telemetry Transport (MQTT), a lightweight protocol for device communication, remote monitoring, and telemetry.

The PoC does not constrain the future developments to a specific vendor ecosystem, leaving the designers to evaluate different solutions to find the most cost effective. Therefore, the full data path from the FIFO to the network stack has been designed using vendor agnostic components from colibri. A proof of the effectiveness of this agnostic design was demonstrated by the CERN Microelectronics group which adapted successfully the gateware to a Microchip FPGA development kit with minimal adjustments.

An FPGA-based FEE emulator, designed by Mitja Vodnik, complements the PoC. The design provides up to 5 lpGBT links, one using the lpGBT chip mounted on the VLDB+ evaluation board and the remaining ones are emulated in the FPGA and transmitted using COTS SFP+ transceivers. A diagram of the gateware is shown in Figure 6.10. The data sent on the links is read from the DDR4 RAM available on-board. The user can load predefined sequences in memory to emulate multiple kinds of FE data formats.

Measurements Testing utilized the FEE emulator as a data generator, with the NetGBT connected directly to a host NIC via a Direct Attach Copper cable. The objective was to determine the minimum packet size needed for maximum throughput without data loss or backpressure.

The test system featured an AMD Threadripper 2990WX CPU, 64 GB DDR4 RAM, and a Mellanox ConnectX-6 NIC, running Alma Linux 9.2 with kernel 5.14. To achieve the best UDP performance jumbo packets were enabled setting the Ethernet MTU to 9000, kernel buffer sizes were increased to 10 MB, and the kernel firewall was set to not track and filter any packet for the UDP destination port.

Throughput and packet rate were measured by taking 10 samples of 1 second to reduce the impact of external factors. The results in Figure 6.11 show that the point-to-point connection reaches its peak throughput of (9064 ± 2) Mbps with a packet size of 3584 B. The correspondent packet rate is (312490 ± 54) PPS. Therefore, 4 kB can be assumed as the lower bound for packet sizes in this application.

Another set of measurements considers the resource utilization of the design. The PoC gateware uses a small amount of resources, as shown in Table 6.1. This enables the implementation of some detector-specific data processing blocks directly on the FPGA, offloading some decoding tasks for the HLT stages. Therefore, resource utilization of different current and future data processing components has been collected and normalized to the specified device.

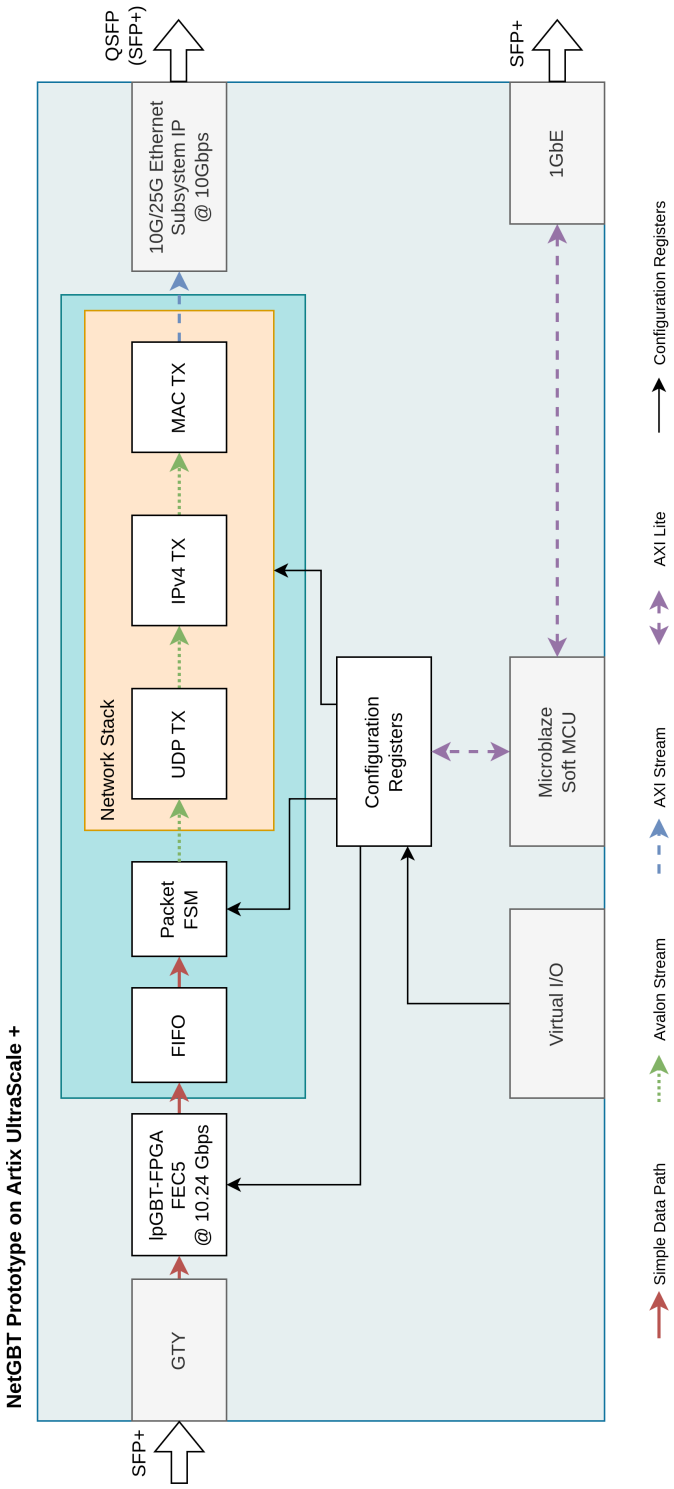


Figure 6.9: Gateware diagram of the NetGBT PoC. The diagram shows a single lpGBT link to one 10GbE link conversion.

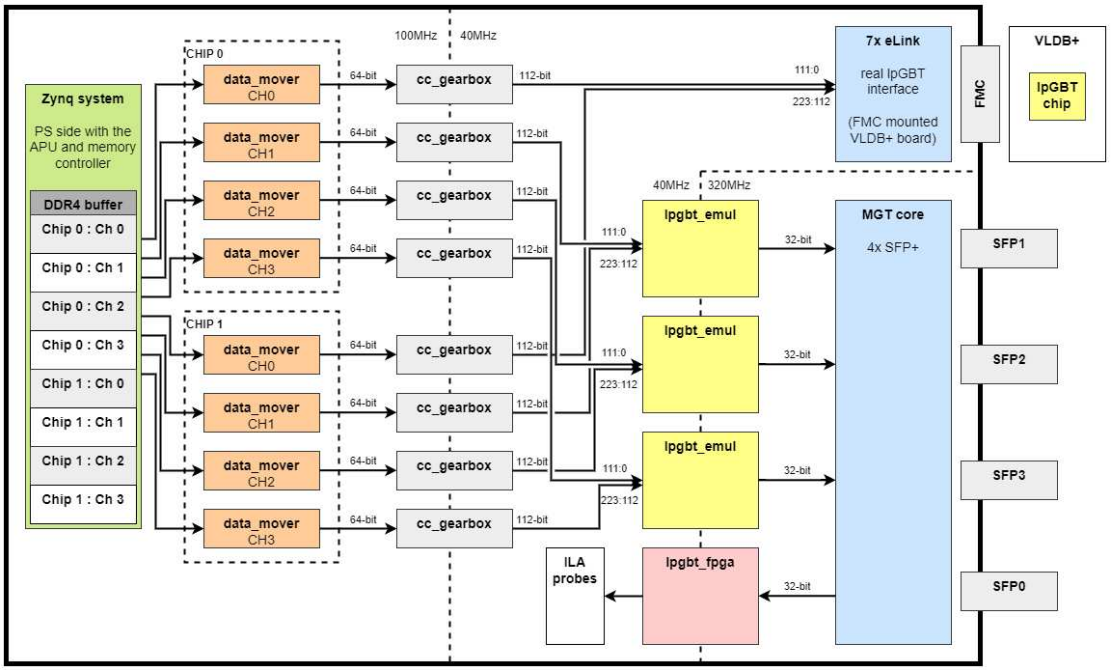


Figure 6.10: Gateware diagram of the lpGBT FE emulator. Courtesy of Mitja Vodnik.

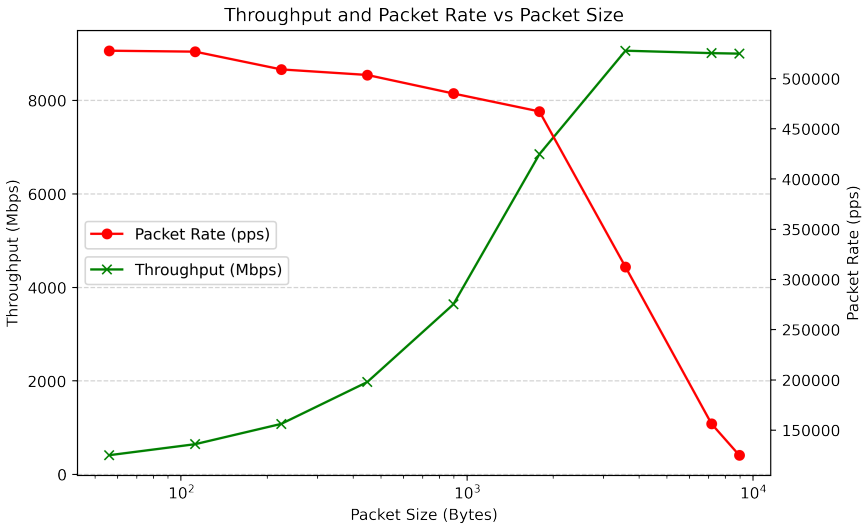


Figure 6.11: Throughput and packet rate measurements for a single lpGBT link converted into a single 10GbE UDP stream.

From Figure 6.12 it is clear that simpler data processing tasks involving bit manipulation and packing, represented by the CALO, can be inserted easily in the data path. More complex tasks which do some degree of reconstruction, represented by the VeLo, could also fit within the available resource. However, these measurements only consider a fraction of the complete data processing as the complete dataflow is not yet implemented.

	LUTs	LUT RAM	FF	Block RAM
Utilization	20056	298	34500	34
Utilization (%)	14.22	0.30	12.23	11.33

Table 6.1: Resource utilization for 4 lpGBT links on AMD AU25P

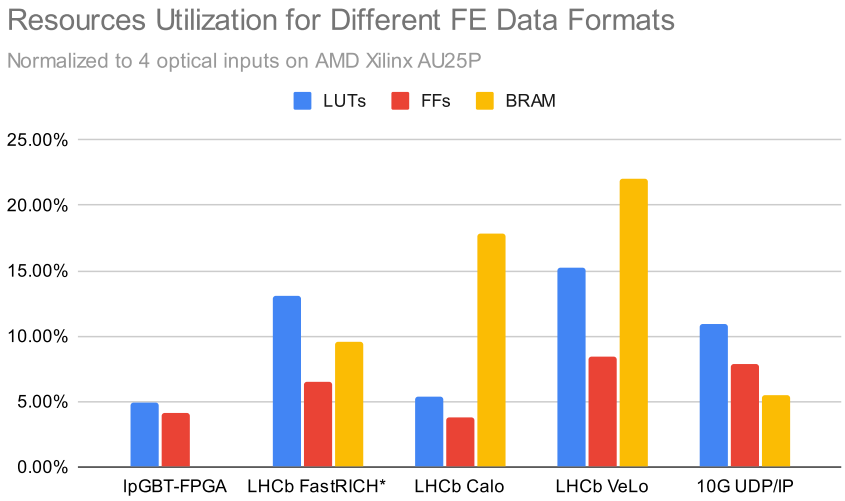


Figure 6.12: Resource utilization in percentage for different FE data format processing blocks normalized to the 4 optical links implemented on the AMD Xilinx Artix Ultrascale+ AU25P. Flip Flops (FFs), Look-Up Tables (LUTs), and Block RAM (BRAM) are common resources available in the FPGAs to implement gateware functionality.

Future Directions The positive outcomes of the PoC have driven the advancement of the project. A more comprehensive solution has been designed to meet the full requirements of the LHCb DAQ. This solution utilizes a System-on-Module (SoM) based on the AMD Zynq Ultrascale+ ZU17EG, capable of converting up to 48 lpGBT links and supporting 100GbE. The use of a SoM not only reduces costs but also simplifies the carrier board design. Additionally, it offers flexibility, allowing users to upgrade to a more powerful FPGA simply by replacing the module with a pin-compatible one. Moreover, multiple SoMs can be integrated into a single 19 inch rack unit, enabling very high link density.

The NetGBT gateware was successfully ported to the SoM evaluation kit within a few hours. This kit facilitated the testing of 100GbE UDP streams. However, the standard UDP implementation in the Linux kernel is not fast enough to efficiently process these packets, leading to significant packet loss. This issue can be addressed by using the open-source Data Plane Development Kit (DPDK) project [28], which allows packet processing in userspace by bypassing the traditional kernel, thereby reducing costly data copies and interrupt overheads. Preliminary results demonstrate the ability to handle full 100GbE UDP streams without backpressure when using DPDK. Nevertheless, these findings are initial, and further testing is required.

6.3 HLS for HLT Acceleration

This section presents the application of novel HLS methodologies to accelerate HEP reconstruction tasks using FPGAs.

The first selection stage in the current LHCb DAQ architecture is the HLT1, as described in Section 1.3. This trigger stage processes incoming data using inclusive one- and two-track algorithms for event selection. The goal of HLT1 is to efficiently reduce the 30 MHz raw data input rate by a factor of 30 to 60, depending on the configuration, which is achieved through full track reconstruction.

HLT1 algorithms are embarrassingly parallel problems, making them highly suitable for implementation on many-core architectures such as GPUs. This approach supports a leaner LHCb DAQ architecture by integrating GPUs directly into the EB servers, thereby reducing infrastructure costs.

Many trigger tasks involve extensive combinatorial and bitwise operations, which are well-suited for FPGAs. However, the adoption of FPGAs as accelerators for computational workloads is often limited by the specialized expertise required and the steep learning curve of traditional development environments. The HLS methodology addresses this challenge by leveraging higher-level programming languages and providing a development environment similar to that used in software engineering.

The case study evaluates the capabilities and maturity of the latest HLS workflow developed by Intel for Altera's high-end FPGAs, specifically the Agilex 7 Family.

For this evaluation, the Upgrade I VeLo decoding and clustering algorithms have been selected.

VELO Raw Data Format The Upgrade I VeLo is the silicon detector described in Section 1.3, and its primary function is to enable the reconstruction of tracks and displaced vertices. The subdetector consists of 52 modules positioned on either side of the beamline and oriented perpendicular to it. Each module contains 4 pixel sensors, with each sensor comprising 3 chips, each with 256×256 pixels. Thus, the total number of pixels can be calculated as follows:

$$(52 \text{ modules}) \times (4 \text{ sensors}) \times (3 \text{ chips}) \times (256 \times 256 \text{ pixels}) \simeq 40\,\text{M}$$

If each pixel were encoded individually, the required bandwidth would be approximately ~ 1600 Tb/s, which is 50 times greater than the total bandwidth of the entire experiment. Therefore, a data reduction technique is essential to make the readout feasible.

This reduction method leverages the fact that the average detector occupancy is below 0.1%. The data format groups pixels into arrays called Super Pixels (SPs), with each SP consisting of 8 pixels. Only the SPs containing active pixels are transmitted, which necessitates encoding the location coordinates relative to the sensor.

The VeLo raw data is organized into 208 *raw banks*, with each bank corresponding to a distinct sensor. Each raw bank contains an *SP header* followed by a sequence of *SP words*. The number of SP words is encoded in the 16 Least Significant Bits (LSBs) of the header. An SP word, shown in Figure 6.13, consists of 9 bits to encode the sensor columns, 6 bits for the rows, and 8 bits representing the pixel hitmap.

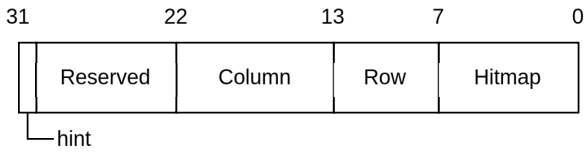


Figure 6.13: Data format of an SP word.

6.3.1 Clustering Algorithm

When particles interact with the subdetector pixels, they produce signals that often activate multiple pixels. To accurately determine the actual hit coordinates, an additional reconstruction step called *clustering* is required. In this process, neighboring active pixels are grouped into *clusters*, and the coordinates and size of each cluster are subsequently computed.

Several approaches can be applied to this problem. The HLT VeLo employs a variant of the *Connected Component Analysis* known as *Mask Clustering*. In this algorithm, a neighbor is defined as any pixel that shares an edge or corner with another pixel, following a pattern called *8-connectivity*. A preprocessing step, described in [21], identifies a set of candidates—active pixels that are likely to form clusters. After this, the mask clustering algorithm proceeds through the following steps:

1. The SP containing the candidate is loaded into memory along with its neighboring SPs. The pixels are arranged in a two-dimensional matrix consisting of three columns and four rows of SPs, totaling 96 pixels. The candidate SP is positioned at (1,1).

2. A bit mask is created around the cluster candidate and its 8-connected neighboring pixels.

3. The bit mask is applied to the matrix using a logical AND operation, which selects the active neighboring pixels. This operation defines the initial cluster boundaries.

4. A new mask is generated around the forming cluster, encompassing all pixels within the cluster and their 8-connected neighbors. This mask is then applied to the SP matrix. This step is repeated until there are no changes in the cluster boundaries between successive iterations, indicating that all cluster pixels have been identified.

The bit mask is computed using eight bitwise shift operations combined with logical OR operations, allowing the mask to be generated in constant time.

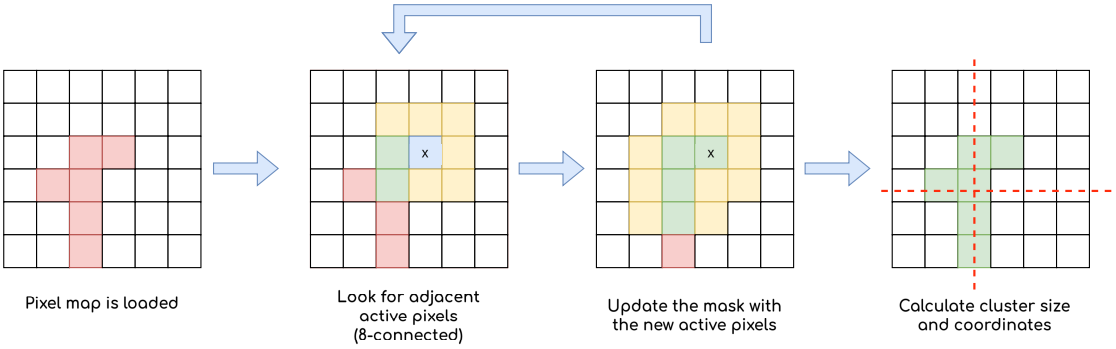


Figure 6.14: Illustration of the masked clustering algorithm.

The described algorithm is highly parallelizable, illustrated in Figure 6.14, as there are no physical dependencies between sensors and events. This enables the storage of only a subset of active SPs in local registers, allowing for extremely fast access. In the GPU implementation, the algorithm is encapsulated within a *kernel*, which serves as the fundamental computational unit. Multiple compute kernels are deployed using a block-and-thread approach, with each kernel operating independently to process candidates, banks, and events. For example, when iterating over events with blocks and banks with threads, kernel 1 would process candidates in bank 1 of event 1, and so on.

6.3.2 FPGA Architecture

The described algorithm is available in three implementations: CPU, CUDA, and SYCL, with the latter two specifically optimized for GPUs. The oneAPI framework enables execution across different platforms, including FPGAs. However, additional modifications are required to adapt the code to the distinct architecture of FPGA-based accelerators. This section will detail the work involved in porting the algorithm to the FPGA and present the benchmark results for this implementation.

Firstly, FPGAs do not utilize the concept of blocks or threads, which are specific to GPUs. As a result, there is no direct equivalent to the GPU-specific `parallel_for` instruction, which automatically deploys kernels iterating over user-specified

indices and arguments. Instead, deployment must be handled using the `single_task` approach, where kernels are launched independently, and iteration is manually implemented by the user. This fundamental difference necessitates slight modifications to the original source code to correctly manage memory regions and kernel arguments.

After making these adjustments, the algorithm was successfully executed on a Bittware IA-840f FPGA-based PCIe accelerator. The results were compared with those produced by the GPU implementation, confirming the correctness of the output. The entire adaptation process took less than a month, despite the author's limited prior experience with SYCL or oneAPI. This initial PoC was functional, demonstrating the feasibility of the approach, but its performance was inferior to the CPU-only version, as shown in Figure 6.17.

Performance issues were identified using the built-in profiler and compilation reports. Notably, the profiler revealed that the PCIe throughput was only a few MB/s, far below the expected $\mathcal{O}(10)$ GB/s. This is a critical issue since FPGAs have limited on-board memory and slower external RAM compared to GPUs. Thus, accelerators rely on the main system memory, performing transactions over PCIe via DMA.

An initial optimization involved separating the computational part of the algorithm from memory transactions into distinct kernels. This was facilitated by built-in *pipes*, which act as hardware-implemented FIFOs, enabling unidirectional communication between kernels.

The updated design consisted of:

- A **producer** kernel that reads from the host memory,
- A **worker** kernel responsible for computations, and
- A **consumer** kernel that writes results back to the host memory.

Dividing the kernels reduced complexity and allowed the compiler to apply optimizations, such as recompiling only modified kernels, thereby improving compilation times.

However, the performance gains were minimal, though this restructuring helped isolate the bottlenecks. Compiler reports indicated that the hardware design was inferring an excessive number of Load-Store Units (LSUs)—components responsible for handling memory load and store operations. Multiple LSUs on the same bus cause contention, potentially leading to deadlocks. To mitigate this, the compiler instantiates a bus arbiter to manage bus access among peripherals, but this introduces significant latency and degrades performance.

Addressing this issue required a complete redesign of the buffer architecture to consolidate all raw data into a single continuous memory region. This entailed interleaving candidates with raw banks, following the format shown in Figure 6.15, which necessitated preprocessing on the host. The producer kernel was redesigned to accommodate this new format, incorporating a FSM to manage memory operations. With these changes, the compiler inferred a single LSU and eliminated the bus arbiter, effectively removing the bottleneck.

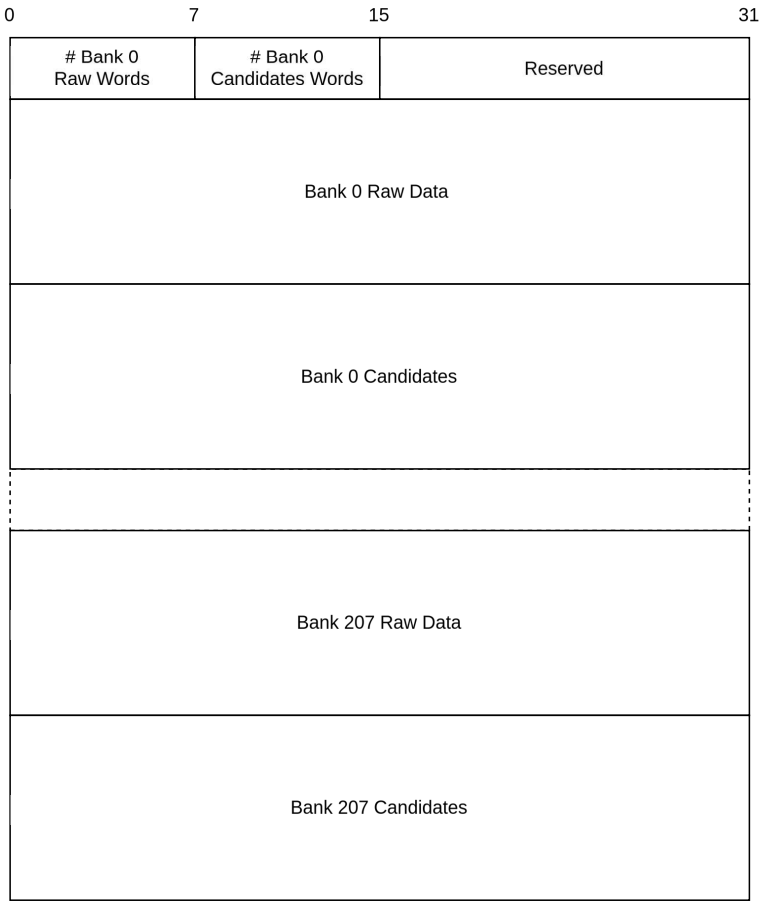


Figure 6.15: Data format of the input buffer, interleaving candidates and raw banks to enable sequential reads and infer a single LSU.

At this stage, computation became the new performance bottleneck. The design employed a single worker, leaving significant unused FPGA resources. Optimizing performance required instantiating multiple workers, akin to the GPU block-and-thread model—though this is considerably more complex on FPGAs.

To achieve this, two new kernels were introduced:

- **Distributor**: Receives output from the producer and distributes it to a pool of workers in a round-robin fashion. Separate distributors handle candidates and raw banks.
- **Collector**: Aggregates results from the worker pool into a single pipe connected to the consumer.

All kernels communicate through appropriately sized pipes to prevent backpressure. Parallel computation not only requires multiple workers but also necessitates synchronization mechanisms to prevent data corruption. To ensure this, a synchronization kernel was added to coordinate memory operations with event boundaries. This kernel connects to the producer, consumer, and worker kernels via dedicated pipes, regulating the following flow:

1. The producer reads an event from host memory and dispatches it to workers via the distributors. It then waits for a release signal from the synchronization kernel before reading the next event.
2. Each worker processes its assigned task and sends a completion signal to the synchronization kernel, then waits for the release signal.
3. The consumer collects results from the worker pool through the collector and sends a completion signal to the synchronization kernel, then waits for the release signal.
4. The synchronization kernel waits for all completion signals and broadcasts a release signal to all kernels, allowing the system to process the next event.

This mechanism, known as a *central barrier*, ensures that each event is processed within its boundaries, preventing data corruption.

The complete design, incorporating all the features described, is illustrated in Figure 6.16.

6.3.3 Results

The multi-kernel pipelined design was evaluated using the Bittware IA-840f FPGA Acceleration Card, based on an Altera Agilex 7 F-Series FPGA. To assess platform stability and compatibility, tests were conducted on two servers: one equipped with an Intel Xeon Gold 6326 CPU, and the other with a newer Intel Xeon Gold 6426Y CPU. Both hosts were connected to the accelerator via a PCIe 4.0 x16 interface.

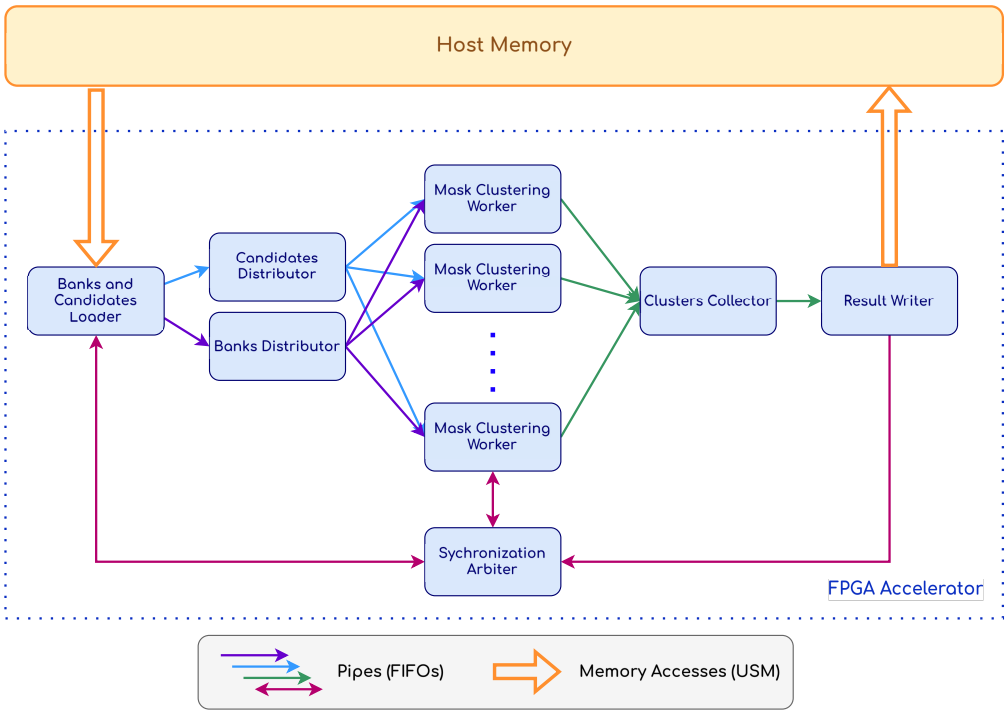


Figure 6.16: Diagram of the FPGA implementations. From the left, the producer reads from the input buffer in the system memory. Candidates and clusters are distributed to the worker pool in a round-robin manner. Multiple workers apply the clustering algorithm. The collector gathers the results, which are written back to host by the consumer. The whole pipeline is kept synchronized by the synchronization kernel.

The first set of tests aimed to determine the optimal number of worker kernels deployed on the accelerator. Each test involved processing 100,000 events, using a batch size of 100 events repeated 1,000 times. The results, presented in Table 6.2, measured both the event processing rate and the input PCIe throughput. The system demonstrated effective scalability, achieving a peak event rate of 107.9 kHz with 16 worker kernels. The speedup from 4 to 8 workers was nearly linear, while the increase from 8 to 16 workers resulted in only a 50% improvement. Moreover, the kernel utilized only one-fifth of the available PCIe bandwidth, indicating that computation, rather than data transfer, was the primary bottleneck.

# workers	Event Rate (kHz)	PCIe Throughput (Gbps)	Speedup
4	37.5	15.8	1.0
8	71.4	30.0	1.9
16	107.9	45.3	2.9

Table 6.2: Worker scaling analysis results. The event rate, PCIe throughput, and corresponding speedup are presented for configurations with 4, 8, and 16 workers.

The second set of tests provided a comparative analysis across different platforms and architectures. Each test again processed 100,000 events with the same batch configuration. The platforms evaluated were as follows:

- **CPU:** Execution on an Intel Xeon Gold 6326 CPU utilizing 16 compute threads.
- **GPU:** Execution on an NVIDIA RTX A5000 GPU hosted on a production server used for HLT1.
- **FPGA PoC:** The initial FPGA implementation, based on the GPU algorithm without architectural optimizations, executed on the IA-840f card in the Xeon Gold 6326 host.
- **FPGA Optimized:** The optimized, pipelined FPGA implementation described in this work, executed on the IA-840f card.

As shown in Figure 6.17, the PoC implementation performed the worst, achieving only 300 Hz—40 times slower than the CPU implementation. In contrast, the optimized design outperformed the GPU implementation, achieving a 36% higher event rate. Despite this promising result, performance was affected by a runtime bug in oneAPI that caused kernel panics when the accelerator attempted to write data back to the host memory. This issue was isolated and reported to both the hardware vendor and Intel, who confirmed its reproducibility but have not yet provided a solution.

To circumvent the problem, a modified consumer kernel was implemented to mimic the behavior of the original while avoiding host memory transactions. This workaround is assumed to have a negligible impact on performance, as the PCIe bus remained underutilized, consistent with the input PCIe throughput measurements.

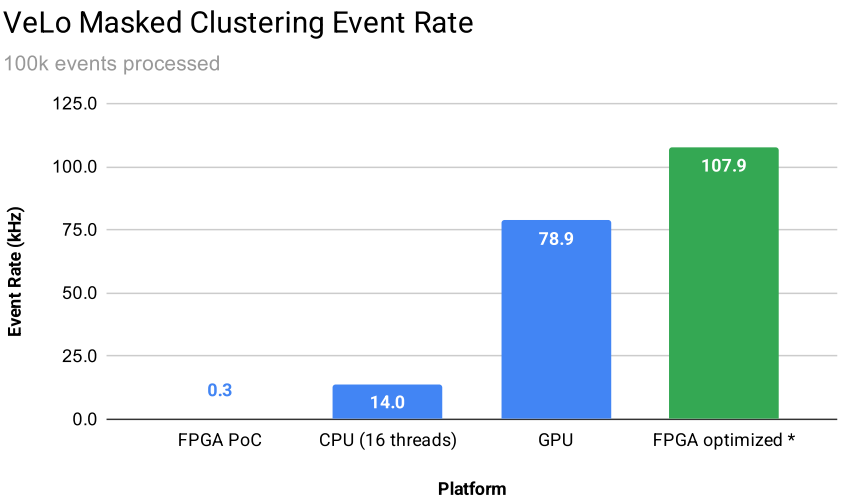


Figure 6.17: Results of the multi-platform benchmark. Note that the FPGA optimized result is not writing back to host because of the kernel panic issue.

7 Conclusions

The work presented in this thesis has systematically evaluated modern technologies and methodologies for FPGA gateware development. The findings emphasize the necessity of adopting a verification-driven approach for HDL projects to enhance reliability, maintainability, and efficiency.

7.1 Achievements

All aspects of this work rely, at least in part, on open-source software. This is a crucial factor, as open-source tools enable broader accessibility, allowing new users to develop FPGA-based systems without the constraints of proprietary licensing. Below is a summary of the key outcomes across the discussed topics.

Functional Verification Functional verification frameworks significantly enhanced testbenches by introducing constrained random stimulus generation and automated checking. Both UVVM and OSVVM demonstrated advanced capabilities, meeting all verification requirements.

A key advantage of UVVM is its extensive library of VVCs, which provide TLM-based behavioral models for widely used protocols. These VVCs are highly adaptable and can be extended for custom applications.

OSVVM excels in constrained random testing and test coverage functionalities. Integrated scoreboards and random generators improve test quality through detailed reporting and analysis.

VUnit proved to be the most extensively used framework due to its Python scripting interface. It supports multiple simulators through a unified Python script, streamlining both development and debugging. This capability allows developers to accelerate iterations using fast proprietary simulators while leveraging slower open-source alternatives in Continuous Integration environments without requiring additional licenses. Additionally, VUnit facilitates parameterized test generation via configurable generic ports, enabling efficient parameter sweeps and comprehensive test reporting.

The application of these frameworks to the existing testbenches for the PCIe40 enhanced both the quality and portability, significantly extending the test coverage.

Formal Verification Although open-source FV tools are less mature than the functional verification frameworks, applying formal methods demonstrated their significant value in FPGA design and testing. FV tools currently lack support for multiple

clock domains, limiting their applicability. However, when deployed on suitable components, FV techniques isolated an unexpected edge case within days—a process that would have taken weeks using traditional debugging approaches.

Given its effectiveness, FV should be applied wherever possible, even if limited to specific design components. Experience with IEEE PSL and the ability to express design specifications as logic equations require practice, but the investment substantially reduces debugging time.

Applications in the current PCIe40 gateware helped isolate hidden bugs that had escaped into production after commissioning, highlighting the importance of this methodology.

Common Core Library The development of the colibri library applied all the tools and methodologies discussed in this thesis. Standardized components significantly reduced development time for new projects, while an extensive suite of automated testbenches ensured correctness and reliability. The vendor independence was validated by porting FPGA designs across multiple vendors, requiring only minor timing constraint adjustments.

The colibri library has already been adopted by developers from ALICE, LHCb, and CMS, with additional teams at CERN expressing interest in integrating the library into their workflows.

High Level Synthesis The oneAPI FPGA Toolkit proved to be a robust development tool, particularly due to its advanced debugging and profiling capabilities. SYCL facilitated cross-platform adaptation with minimal source code modifications. However, optimizing performance required significant effort due to architectural differences, particularly in memory operations.

The final optimized implementation fully utilized available FPGA resources, achieving a 36% higher event processing rate compared to a high-end GPU. Nonetheless, several limitations were identified, the most critical being a kernel panic bug during host memory writes. Consequently, oneAPI is not yet considered mature enough for production deployments in high-throughput data processing applications, necessitating further development to improve reliability.

7.2 Future Directions

For LHCb Upgrade II, all tools and techniques described earlier will be adopted as standard for all designs. Functional verification is being implemented in the new developments for the PCIe400, while part of the work will go into upgrading the current tests for the PCIe40 to the new frameworks. The colibri library is going to be integrated in all future design giving a common base to all engineers, while simplifying simulation and verification. FV will be extensively used to complement the verification efforts and it will be used in the early development phase to avoid bug escapes.

For future Ethernet-based readout systems, the efforts will focus on developing a second prototype of netGBT, designed to aggregate more links over higher-speed Ethernet streams. This work will also focus on establishing some guidelines on future FE data formats which make more efficient use of the resources available in both hardware and software.

Additionally, HLS tools will continue to be evaluated to assess their suitability for DAQ development, even if there is no plan of deployment for this technology.

Bibliography

[1] *2022 Wilson Research Group Functional Verification Study*. en-US. Oct. 2022. URL: https://blogs.sw.siemens.com/verificationhorizons/2022/10/10/prologue-the-2022-wilson-research-group-functional-verification-study/ (visited on 01/30/2025) (cit. on pp. 85, 106).

[2] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020. URL: https://www.sciencedirect.com/science/article/pii/S037026931200857X (visited on 05/16/2022) (cit. on p. 35).

[3] R. Aaij et al. "The LHCb Upgrade I". en. In: *Journal of Instrumentation* 19.05 (May 2024), P05065. ISSN: 1748-0221. DOI: 10.1088/1748-0221/19/05/P05065. URL: https://iopscience.iop.org/article/10.1088/1748-0221/19/05/P05065 (visited on 10/29/2024) (cit. on pp. 43, 46, 47, 49, 50, 52).

[4] Roel Aaij et al. "Measurements of prompt charm production cross-sections in pp collisions at \sqrt s=13 TeV". In: *JHEP* 03 (2016), p. 159. DOI: 10.1007/JHEP03(2016)159 (cit. on p. 39).

[5] Roel Aaij et al. "Allen: A high level trigger on GPUs for LHCb". In: *Comput. Softw. Big Sci.* 4.1 (2020), p. 7. DOI: 10.1007/s41781-020-00039-7 (cit. on p. 59).

[6] M. Adinolfi et al. *Performance of the LHCb RICH detector at the LHC*. Tech. rep. LHCB-DP-2012-003. CERN-LHCb-DP-2012-003, Nov. 2012, p. 2431. DOI: 10.1140/epjc/s10052-013-2431-9. URL: https://cds.cern.ch/record/1495721 (visited on 06/30/2022) (cit. on p. 50).

[7] *Agilex™ 7 FPGA and SoC FPGA M-Series*. en. URL: https://www.intel.com/content/www/us/en/products/details/fpga/agilex/7/m-series.html (visited on 02/10/2025) (cit. on p. 118).

[8] Johannes Albrecht et al. "New approaches for track reconstruction in LHCb's Vertex Locator". In: *EPJ Web of Conferences* 214 (Jan. 2019), p. 01042. DOI: 10.1051/epjconf/201921401042 (cit. on pp. 43, 48).

[9] G. B. Andresen et al. "Trapped antihydrogen". In: *Nature* 468.7324 (Dec. 2010), pp. 673–676. ISSN: 1476-4687. DOI: 10.1038/nature09610. URL: https://www.nature.com/articles/nature09610 (visited on 05/16/2022) (cit. on p. 35).

[10] G. Arnison et al. “Experimental observation of isolated large transverse energy electrons with associated missing energy at s=540 GeV”. In: *Physics Letters B* 122.1 (Feb. 1983), pp. 103–116. ISSN: 0370-2693. DOI: 10.1016/0370-2693(83)91177-2. URL: https://www.sciencedirect.com/science/article/pii/0370269383911772 (visited on 05/16/2022) (cit. on p. 35).

[11] ATLAS Collaboration. *ATLAS Phase-II Upgrade Scoping Document*. Tech. rep. CERN-LHCC-2015-020. CERN, 2015. DOI: 10.17181/CERN.7CRX.AJHP. URL: https://cds.cern.ch/record/2055248 (visited on 12/04/2024) (cit. on p. 67).

[12] ATLAS Collaboration. *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*. Tech. rep. ATLAS-TDR-029. CERN, 2017. DOI: 10.17181/CERN.2LBB.4IAL. URL: https://cds.cern.ch/record/2285584 (visited on 12/04/2024) (cit. on pp. 67, 68).

[13] *Aurora 64B/66B*. en. URL: https://www.xilinx.com/products/intellectual-property/aurora64b66b.html (visited on 02/06/2025) (cit. on pp. 64, 112).

[14] J. R Batley et al. “A precision measurement of direct CP violation in the decay of neutral kaons into two pions”. In: *Physics Letters B* 544.1 (Sept. 2002), pp. 97–112. ISSN: 0370-2693. DOI: 10.1016/S0370-2693(02)02476-0. URL: https://www.sciencedirect.com/science/article/pii/S0370269302024760 (visited on 05/16/2022) (cit. on p. 35).

[15] M. Bellato et al. “A PCIe Gen3 based readout for the LHCb upgrade”. In: *Journal of Physics: Conference Series* 513.1 (June 2014), p. 012023. ISSN: 1742-6596. DOI: 10.1088/1742-6596/513/1/012023. URL: https://doi.org/10.1088/1742-6596/513/1/012023 (visited on 05/20/2022) (cit. on p. 57).

[16] Sean Benson et al. “The LHCb Turbo Stream”. In: *Journal of Physics: Conference Series* 664.8 (Dec. 2015), p. 082004. ISSN: 1742-6596. DOI: 10.1088/1742-6596/664/8/082004. URL: https://doi.org/10.1088/1742-6596/664/8/082004 (visited on 06/21/2022) (cit. on p. 61).

[17] S. Borghi. “Novel real-time alignment and calibration of the LHCb detector and its performance”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. Proceedings of the Vienna Conference on Instrumentation 2016 845 (Feb. 2017), pp. 560–564. ISSN: 0168-9002. DOI: 10.1016/j.nima.2016.06.050. URL: https://www.sciencedirect.com/science/article/pii/S0168900216305927 (visited on 11/12/2024) (cit. on p. 60).

[18] “LHC Design Report Vol.1: The LHC Main Ring”. In: (June 2004). Ed. by Oliver S. Bruning et al. DOI: 10.5170/CERN-2004-003-V-1 (cit. on p. 36).

[19] Emma Buchanan. “The LHCb Vertex Locator (VELO) Pixel Detector Upgrade”. In: *JINST* 12 (Jan. 2017), p. C01013. DOI: 10.1088/1748-0221/12/01/C01013. URL: https://cds.cern.ch/record/2243156 (visited on 07/05/2022) (cit. on p. 44).

[20] S. Cadeddu et al. *LHCb trigger system : Technical Design Report*. Tech. rep. CERN-LHCC-2003-031. CERN, 2003. URL: https://cds.cern.ch/record/630828 (visited on 07/03/2022) (cit. on p. 54).

[21] Daniel Hugo Campora Perez. "Optimization of high-throughput real-time processes in physics reconstruction". PhD thesis. Seville U., 2019. URL: https://cds.cern.ch/record/2718278 (visited on 09/11/2024) (cit. on p. 127).

[22] A. Caratelli et al. "The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments". en. In: *Journal of Instrumentation* 10.03 (Mar. 2015), p. C03034. ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/03/C03034. URL: https://dx.doi.org/10.1088/1748-0221/10/03/C03034 (visited on 01/30/2025) (cit. on p. 73).

[23] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (Sept. 2012), pp. 30–61. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581 (visited on 05/16/2022) (cit. on p. 35).

[24] *colibri*. Feb. 2025. URL: https://gitlab.cern.ch/colibri/colibri (visited on 02/24/2025) (cit. on p. 99).

[25] DUNE Collaboration et al. "The DUNE Far Detector Interim Design Report Volume 1: Physics, Technology and Strategies". In: (July 2018). DOI: 10.48550/arXiv.1807.10334. URL: http://arxiv.org/abs/1807.10334 (visited on 12/04/2024) (cit. on p. 67).

[26] LHCb collaboration. "Letter of Intent for the LHCb Upgrade". In: (Mar. 2011). URL: https://cds.cern.ch/record/1333091 (visited on 07/03/2022) (cit. on p. 55).

[27] D. Decamp et al. "A precise determination of the number of families with light neutrinos and of the Z boson partial widths". In: *Physics Letters B* 235.3 (Feb. 1990), pp. 399–411. ISSN: 0370-2693. DOI: 10.1016/0370-2693(90)91984-J. URL: https://www.sciencedirect.com/science/article/pii/037026939091984J (visited on 05/16/2022) (cit. on p. 35).

[28] *DPDK/dpdk*. Feb. 2025. URL: https://github.com/DPDK/dpdk (visited on 02/10/2025) (cit. on p. 126).

[29] Paolo Durante et al. "An automated pipeline for continuous integration of FPGA firmware and software for the LHCb Run3 upgrade". en. In: *Proceedings of Topical Workshop on Electronics for Particle Physics — PoS(TWEPP2018)*. Vol. 343. SISSA Medialab, July 2019, p. 069. DOI: 10.22323/1.343.0069. URL: https://pos.sissa.it/343/069 (visited on 01/17/2025) (cit. on p. 78).

[30] Christian Elsasser. *LHCb bb production angle plots*. URL: https://lhcb.web.cern.ch/speakersbureau/html/bb_productionangles.html (visited on 06/30/2022) (cit. on p. 40).

[31] Stephane Fartoukh et al. “LHC Configuration and Operational Scenario for Run 3”. In: (Nov. 2021). URL: https://cds.cern.ch/record/2790409 (visited on 05/18/2022) (cit. on p. 37).

[32] Mauricio Feo et al. “The Real-Time System for Distribution of Clock, Control and Monitoring Commands With Fixed Latency of the LHCb Experiment at CERN”. In: *IEEE Transactions on Nuclear Science* 70.6 (June 2023), pp. 985–992. ISSN: 1558-1578. DOI: 10.1109/TNS.2023.3273086. URL: https://ieeexplore.ieee.org/document/10115510 (visited on 12/29/2024) (cit. on pp. 73, 74).

[33] *FPGA Support Package for Intel® oneAPI DPC++/C++ Compiler*. en. URL: https://www.intel.com/content/www/us/en/developer/tools/oneapi/fpga.html (visited on 01/30/2025) (cit. on p. 104).

[34] Luis Miguel Garcia et al. “Tracking performance for long-lived particles at LHCb”. In: *Journal of Physics: Conference Series* 1525.1 (Apr. 2020), p. 012095. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1525/1/012095. URL: https://doi.org/10.1088/1742-6596/1525/1/012095 (visited on 07/04/2022) (cit. on p. 46).

[35] Nick Gasson. *nickg/nvc*. Jan. 2025. URL: https://github.com/nickg/nvc (visited on 01/30/2025) (cit. on p. 100).

[36] *ghdl/ghdl*. Jan. 2025. URL: https://github.com/ghdl/ghdl (visited on 01/30/2025) (cit. on p. 100).

[37] *GitLab*. en-us. URL: https://about.gitlab.com/ (visited on 01/30/2025) (cit. on pp. 78, 103).

[38] V. Gromov et al. “Development of a low power 5.12 Gbps data serializer and wireline transmitter circuit for the VeloPix chip”. en. In: *Journal of Instrumentation* 10.01 (Jan. 2015), p. C01054. ISSN: 1748-0221. DOI: 10.1088/1748-0221/10/01/C01054. URL: https://dx.doi.org/10.1088/1748-0221/10/01/C01054 (visited on 01/30/2025) (cit. on p. 73).

[39] F. J. Hasert et al. “Observation of neutrino-like interactions without muon or electron in the gargamelle neutrino experiment”. In: *Physics Letters B* 46.1 (Sept. 1973), pp. 138–140. ISSN: 0370-2693. DOI: 10.1016/0370-2693(73)90499-1. URL: https://www.sciencedirect.com/science/article/pii/0370269373904991 (visited on 05/16/2022) (cit. on p. 35).

[40] “IEEE Standard for Information technology - Local and metropolitan area networks - Part 3: CSMA/CD Access Method and Physical Layer Specifications - Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation”. In: *IEEE Std 802.3ae-2002 (Amendment to IEEE Std 802.3-2002)* (Aug. 2002), pp. 1–544. DOI: 10.1109/IEEESTD.2002.94131. URL: https://ieeexplore.ieee.org/document/1040118 (visited on 02/07/2025) (cit. on p. 114).

[41] "IEEE Standard for Property Specification Language (PSL)". In: *IEEE Std 1850-2005* (Oct. 2005), pp. 1–143. DOI: 10.1109/IEEESTD.2005.97780. URL: https://ieeexplore.ieee.org/document/1524461 (visited on 01/30/2025) (cit. on p. 96).

[42] "IEEE Standard for Universal Verification Methodology Language Reference Manual". In: *IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017)* (Sept. 2020), pp. 1–458. DOI: 10.1109/IEEESTD.2020.9195920. URL: https://ieeexplore.ieee.org/document/9195920 (visited on 01/30/2025) (cit. on p. 93).

[43] *junit-team/junit5*. Jan. 2025. URL: https://github.com/junit-team/junit5 (visited on 01/30/2025) (cit. on p. 103).

[44] F. Keizer. "The FastRICH ASIC for the LHCb RICH enhancements". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1067 (Oct. 2024), p. 169664. ISSN: 0168-9002. DOI: 10.1016/j.nima.2024.169664. URL: https://www.sciencedirect.com/science/article/pii/S0168900224005904 (visited on 02/06/2025) (cit. on p. 112).

[45] Jeremiah Leary. *jeremiah-c-leary/vhdl-style-guide*. Jan. 2025. URL: https://github.com/jeremiah-c-leary/vhdl-style-guide (visited on 01/30/2025) (cit. on p. 103).

[46] LHCb Collaboration. *Framework TDR for the LHCb Upgrade II: Opportunities in flavour physics, and beyond, in the HL-LHC era*. Tech. rep. LHCB-TDR-023. CERN, 2021. URL: https://cds.cern.ch/record/2776420 (visited on 01/17/2025) (cit. on pp. 39, 63, 66, 82).

[47] LHCb Collaboration. *LHCb Data Acquisition Enhancement TDR*. Tech. rep. LHCB-TDR-025. CERN, 2024. URL: https://cds.cern.ch/record/2886764 (visited on 01/16/2025) (cit. on p. 82).

[48] LHCb Collaboration. *Primary Vertex resolution with early 2024 data*. Tech. rep. LHCB-FIGURE-2024-011. CERN, 2024. URL: https://cds.cern.ch/record/2898820 (visited on 11/06/2024) (cit. on p. 45).

[49] LHCb Collaboration. *Track reconstruction efficiency mass plots and efficiency plots for 2024 data and simulation*. Tech. rep. LHCB-FIGURE-2024-032. CERN, 2024. URL: https://cds.cern.ch/record/2913624 (visited on 10/29/2024) (cit. on p. 45).

[50] LHCb Collaboration et al. "Observation of $J/\psi\, p$ Resonances Consistent with Pentaquark States in $\Lambda_b^0 \to J/\psi\, K^- p$ Decays". In: *Physical Review Letters* 115.7 (Aug. 2015), p. 072001. DOI: 10.1103/PhysRevLett.115.072001. URL: https://link.aps.org/doi/10.1103/PhysRevLett.115.072001 (visited on 05/16/2022) (cit. on p. 35).

[51] LHCb Collaboration et al. “Measurement of the b-Quark Production Cross Section in 7 and 13 TeV pp Collisions”. In: *Physical Review Letters* 118.5 (Feb. 2017), p. 052002. DOI: 10.1103/PhysRevLett.118.052002. URL: https://link.aps.org/doi/10.1103/PhysRevLett.118.052002 (visited on 05/18/2022) (cit. on p. 39).

[52] LHCb Collaboration et al. “Observation of CP Violation in Charm Decays”. In: *Physical Review Letters* 122.21 (May 2019), p. 211803. DOI: 10.1103/PhysRevLett.122.211803. URL: https://link.aps.org/doi/10.1103/PhysRevLett.122.211803 (visited on 05/16/2022) (cit. on p. 35).

[53] Guoming Liu et al. “DAQ Architecture for the LHCb Upgrade”. In: *Journal of Physics: Conference Series* 513.1 (June 2014), p. 012027. ISSN: 1742-6596. DOI: 10.1088/1742-6596/513/1/012027. URL: https://doi.org/10.1088/1742-6596/513/1/012027 (visited on 05/20/2022) (cit. on p. 54).

[54] P. Moreira et al. “The GBT Project”. In: (2009). DOI: 10.5170/CERN-2009-006.342. URL: https://cds.cern.ch/record/1235836 (visited on 07/03/2022) (cit. on pp. 56, 72).

[55] *oneAPI: A New Era of Heterogeneous Computing*. URL: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html (visited on 01/30/2025) (cit. on p. 104).

[56] opalkelly.com. *XEM8320*. en-US. URL: https://opalkelly.com/products/xem8320/ (visited on 02/10/2025) (cit. on p. 121).

[57] *OpenCores*. URL: https://opencores.org/ (visited on 01/30/2025) (cit. on p. 98).

[58] OSVVM. *Open Source VHDL Verification Methodology*. URL: https://osvvm.org/ (visited on 01/30/2025) (cit. on p. 94).

[59] Ioannis Papakonstantinou et al. “A Fully Bidirectional Optical Network With Latency Monitoring Capability for the Distribution of Timing-Trigger and Control Signals in High-Energy Physics Experiments”. In: *IEEE Transactions on Nuclear Science* 58.4 (Aug. 2011), pp. 1628–1640. ISSN: 1558-1578. DOI: 10.1109/TNS.2011.2154364. URL: https://ieeexplore.ieee.org/document/5876285 (visited on 01/30/2025) (cit. on p. 73).

[60] Paulo Moreira et al. “The lpGBT: a radiation tolerant ASIC for Data, Timing, Trigger and Control Applications in HL-LHC”. en. In: *TWEPP 2019 Topical Workshop on Electronics for Particle Physics*. URL: https://indico.cern.ch/event/799025/contributions/3486153/ (visited on 10/08/2024) (cit. on p. 64).

[61] A. Perro et al. “A low-cost, low-power media converter solution for next-generation detector readout systems”. en. In: *Journal of Instrumentation* 20.02 (Feb. 2025), p. C02027. ISSN: 1748-0221. DOI: 10.1088/1748-0221/20/02/C02027. URL: https://dx.doi.org/10.1088/1748-0221/20/02/C02027 (visited on 02/13/2025) (cit. on pp. 3, 118).

[62] Alberto Perro. "The New Event Builder of the LHCb Experiment: Network Optimisations for the Online Cluster". MA thesis. Università degli Studi di Torino (IT), 2022. URL: https://cds.cern.ch/record/2834628 (visited on 11/12/2024) (cit. on p. 59).

[63] Alberto Perro et al. *Guidelines for FPGA Gateware Development in LHCb*. 2025. URL: https://cds.cern.ch/record/2929526 (visited on 04/08/2025) (cit. on p. 3).

[64] Florian Reiss. *Real-time alignment procedure at the LHCb experiment for Run 3*. Tech. rep. LHCb-PROC-2023-001. 2023. URL: http://cds.cern.ch/record/2846414 (visited on 11/12/2024) (cit. on pp. 60, 61).

[65] J Serrano et al. "The White Rabbit project". In: 2013. URL: https://cds.cern.ch/record/1743073 (visited on 12/09/2024) (cit. on pp. 65, 69).

[66] Roland Sipos. "The Ethernet Readout of the DUNE DAQ System". In: *IEEE Transactions on Nuclear Science* (2024), pp. 1–1. ISSN: 0018-9499, 1558-1578. DOI: 10.1109/TNS.2024.3486059. URL: https://ieeexplore.ieee.org/document/10735136/ (visited on 12/04/2024) (cit. on pp. 67, 70).

[67] C. Soós et al. "The Versatile Transceiver: towards production readiness". en. In: *Journal of Instrumentation* 8.03 (Mar. 2013), p. C03004. ISSN: 1748-0221. DOI: 10.1088/1748-0221/8/03/C03004. URL: https://dx.doi.org/10.1088/1748-0221/8/03/C03004 (visited on 01/30/2025) (cit. on p. 72).

[68] *SYCL - C++ Single-source Heterogeneous Programming for Acceleration Offload*. en. Jan. 2014. URL: https://www.khronos.org/sycl/ (visited on 01/30/2025) (cit. on p. 104).

[69] *SZG-QSFP*. en-US. URL: https://docs.opalkelly.com/syzygy-peripherals/szg-qsfp/ (visited on 02/10/2025) (cit. on p. 121).

[70] UVVM. *UVVM/UVVM*. Jan. 2025. URL: https://github.com/UVVM/UVVM (visited on 01/30/2025) (cit. on p. 93).

[71] *VUnit/vunit*. Jan. 2025. URL: https://github.com/VUnit/vunit (visited on 01/30/2025) (cit. on p. 94).

[72] *YosysHQ/sby*. Jan. 2025. URL: https://github.com/YosysHQ/sby (visited on 01/30/2025) (cit. on p. 101).

[73] *YosysHQ/yosys*. Jan. 2025. URL: https://github.com/YosysHQ/yosys (visited on 01/30/2025) (cit. on p. 101).