Implementation of embedded artificial intelligence algorithms in the readout system of the ATLAS liquid argon calorimeter

Nemer Chiedde

Aix*Marseille université
Socialement engagée

NNT : 2023AIXM0399

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 21 novembre 2023 par

Nemer CHIEDDE

Implementation of embedded artificial intelligence algorithms in the readout system of the ATLAS liquid argon calorimeter

Discipline
Physique et Sciences de la Matière

Spécialité
Instrumentation

École doctorale
ED 352 Physique et Sciences de la Matière

Laboratoire/Partenaires de recherche
- Centre National de la Recherche Scientifique (CNRS)
- Centre de Physique des Particules de Marseille (CPPM)

Composition du jury

Tetiana HRYN'OVA Laboratoire d'Annecy de Physique des Particules	Rapporteuse
Philippe SCHWEMLING Université Paris Cité et Irfu/CEA-Saclay	Rapporteur
Jean-Baptiste SAUVAN Laboratoire Louis Leprince-Ringuet	Examinateur
Emmanuel MONNIER CPPM	Directeur de thèse
Georges AAD CPPM	Co-directeur de thèse
Cristinel DIACONU CPPM	Président du jury

cnrs

CENTRE DE PHYSIQUE DES PARTICULES DE MARSEILLE
CPPM

AFFIDAVIT

I, undersigned, Nemer CHIEDDE, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Emmanuel MONNIER and Georges AAD, 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 the Aix-Marseille University 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, 21 novembre 2023

LIST OF PUBLICATIONS AND CONFERENCE ATTENDED

List of publications done as part of this thesis project:

1. AAD, G.; BERTHOLD, A.-S.; CALVET, T. P.; CHIEDDE, N.; FORTIN, E.; FRITZSCHE, N.; HENTGES, R. G.; LAATU, L. A. O.; MONNIER, E.; STRAESSNER, A.; VOIGT, J. C. for *Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters*. Geneva, 2021. Available: https://cds.cern.ch/record/2775033

2. CHIEDDE, N. for *Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs*. Geneva, 2021. Available: https://cds.cern.ch/record/2789016

3. CHAMBERY, P.; DE BONU DE LAVERGNE, M.; PRACCHIA, M.; VERNET, K.; CHICHE, S; et.al for *JRJC 2021- Journ'ees de Rencontres Jeunes Chercheurs*. La Rochelle, 2021. Available: https://hal.science/hal-03832762v2/file/JRJC2021_Proceedings.pdf

4. AAD, G.; CALVET, T.; CHIEDDE, N.; FAURE, R.; FORTIN, E. M.; LAATU, L.; MONNIER, M; SUR, N. for *Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment*. Marseille, 2023. Available:https://arxiv.org/pdf/2302.07555.pdf

Conferences and summer schools attended during this thesis period:

1. TWEPP 2021 Topical Workshop on Electronics for Particle Physics; On-line, September 2021

2. Journées de Rencontre des Jeunes Chercheurs 2021: Village La Fayette; La Rochelle - France, October 2021

3. ANF d'électronique numérique IN2P3 "Raw2Smart DATA": IJCLab; Orsay - France, November 2021

4. DIS2023: XXX International Workshop on Deep-Inelastic Scattering and Related Subjects; East Lansing - USA, march 2023

ABSTRACT

The Large Hadron Collider (LHC) is currently the world's leading particle accelerator. In 2012, it allowed the discovery of the Higgs boson, the last unobserved particle in the Standard Model of particle physics. The High Luminosity phase of the LHC (HL-LHC), starting in 2029, will allow further study of the Higgs boson properties and improve new particle discovery potential. Up to 200 simultaneous proton-proton collisions are expected at the HL-LHC, which puts stringent requirements on the ATLAS detector electronic. To prepare for these conditions, the readout electronics of the Liquid Argon (LAr) calorimeter of the ATLAS detector will be replaced. The new electronics will contain high-end Field-Programable Gate Arrays (FPGAs) capable of computing the energy deposited in the calorimeter in real time at 40 MHz. The performance optimal filtering algorithms, currently used to compute the energy, will significantly degrade due to the increased rate of overlapping pulses in high luminosity conditions. Neural networks (NNs) were shown to perfectly recover this performance loss. This thesis presents the implementation of neural network firmware embedded in FPGAs. The development is carried out first in High Level Synthesis (HLS), which offers a convenient solution for adapting and optimizing the firmware parameters to facilitate its implementation on FPGA platforms in the prototyping phase. The resource usage and the performance of the firmware are carefully analyzed. The developed tools are implemented in the HLS4ML library, which is open-source software designed to automatically translate a trained NN to a firmware. A test firmware was also developed to be able to validate the neural network implementation on the hardware. This test firmware allows to inject input values into the neural network and extract the computed energies to be compared with simulation. The computed energies in the FPGA were shown to perfectly match the expected values from simulation.

Keywords : LHC, ATLAS, liquid argon calorimeter, FPGA, digital signal processing, energy reconstruction.

RÉSUMÉ

Le Grand Collisionneur de Hadrons (LHC) est actuellement le plus grand accélérateur de particules au monde. En 2012, il a permis la découverte du boson de Higgs, la dernière particule non observée dans le Modèle Standard de la physique des particules. La phase de Haute Luminosité du LHC (HL-LHC), qui débutera en 2029, permettra d'étudier davantage les propriétés du boson de Higgs et d'améliorer le potentiel de découverte de nouvelles particules. Jusqu'à 200 collisions proton-proton simultanées sont prévues au HL-LHC, ce qui impose des exigences strictes sur l'électronique du detecteur ATLAS. Pour se préparer à ces conditions, l'électronique de lecture du calorimètre à argon liquide (LAr) du détecteur ATLAS sera remplacée. La nouvelle électronique contiendra des réseaux de portes programmables sur site ou Field Programmable gate arrays (FPGAs) haut de gamme capables de calculer en temps réel l'énergie déposée dans le calorimètre à une fréquence de 40 MHz. Les performances des algorithmes de filtrage optimal, actuellement utilisés pour calculer l'énergie, se dégraderont significativement en raison de l'augmentation du taux d'empilement des impulsions dans des conditions de haute luminosité. Il a été démontré que les réseaux neuronaux (NNs) peuvent compenser cette perte de performance. Cette thèse présente la mise en œuvre de firmware de réseaux neuronaux intégrés dans des FPGA. Le développement est d'abord réalisé en High-Level Synthesis (HLS), ce qui offre une solution pratique pour adapter et optimiser les paramètres du firmware afin de faciliter leur mise en œuvre sur les plateformes FPGA en phase de prototypage. L'utilisation des ressources et les performances du firmware sont soigneusement analysées. Les outils développés sont intégrés dans la bibliothèque HLS4ML, qui est un logiciel open-source conçu pour traduire automatiquement un réseau neuronal entraîné en firmware. Un firmware de test a également été développé pour valider la mise en œuvre du réseau neuronal sur le matériel. Ce firmware de test permet d'injecter des valeurs d'entrée dans le réseau neuronal et d'extraire les énergies calculées pour les comparer à la simulation. Il a ensuite ete demontre que les energies calculees dans le FPGA correspondent parfaitement aux valeurs obtenues en simulation.

Mots-clés : LHC, ATLAS, calorimètre à argon liquide, FPGA, traitement numérique du signal, reconstruction de l'énergie.

ACKNOWLEDGEMENTS

When I started my postgraduate academic degree back in 2020, I could not imagine how much I would grow personally and professionally. As a Brazilian in France, I want to express my gratitude to God for having the opportunity of be in contact with wonderful people throughout my graduation period. I am glad that I have this moment and space to thank many people who helped me in some way. I am grateful to my supervisor Emmanuel Monnier and Georges Aad, not only for the ongoing support but also for having accepted to guide me. I deeply thank them for believing in me during this time and for the opportunities they provided for my growth.

It was a pleasure to be part of the CPPM team and meet people who I will remember forever with a lot of nostalgia and affection. In particular, I thank Neelam Kumari, Peter Matta, Frederic Hachon and Franck Salomon for the deep discussions and sharing of valuable knowledge and experiences. To Lauri Laatu and Nairit Sur, thank you for your technical insights and feedback, which helped me greatly in my professional growth. Special thanks to Brigitte Pantat and Véronique Roux for the support and assistance provided with administrative matters. They went above and beyond their duties and always helped me with all the difficulties I faced. I appreciate you making my life easier and more peaceful and for making me feel at home even when I was far away. I want to convey my appreciation to the reviewers of this manuscript, as well as the other jury members, Tetiana Hryn'ova, Philippe Schwemling and Jean-Baptiste Sauvan. Thank you for taking the time to read my work and attend my defense. I would like to extend a special thank you to the laboratory director, Cristinel Diaconu, as well as to the CPPM services to have accompanied me throughout my thesis. Your commitment to academic excellence and professionalism helped me work in a professional and international environment.

I want to say how much I appreciate my friends Angela Ayobi and Diego Arias. Thank you for all the support you have given me over the years. Your kind words, encouragement, and emotional support were essential for me to face the challenges that arose in my path. Your constant support allowed me to move forward, even in the toughest times.

I want to conclude by thanking my mother, Magda Macera Gomes. She has always been by my side and encouraged me to pursue my dreams and goals, even when they seemed impossible. She is a warrior, a constant source of inspiration, and an example of unconditional love and dedication. I also thank my sister, Marcela Chiedde, for the love, support, and encouragement she has always offered me. Your presence in my life has been a blessing. To my grandmother Joana Roldana Macera for the support, care, and love. I thank you for being part of my journey and for always being there when I needed you. May we continue to journey together with joy and gratitude in this life. This accomplishment would not have been possible without you.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

SYNTHÈSE EN FRANÇAIS

0.1 Introduction

En 2012, un moment décisif en physique des particules a eu lieu lorsque les équipes ATLAS et CMS du Grand Collisionneur de Hadrons (LHC) ont révélé la découverte du boson de Higgs, la dernière pièce du puzzle du Modèle Standard (MS) de la physique des particules, enrichissant fondamentalement notre compréhension de la façon dont des particules telles que les bosons W et Z, les quarks et les leptons acquièrent leur masse grâce à leurs interactions avec le champ de Higgs. Le MS sert de tremplin vers une théorie plus complète capable d'élucider des énigmes telles que la matière noire, l'énergie sombre et l'asymétrie matière-antimatière. La prochaine étape de fonctionnement du Grand Collisionneur de Hadrons (LHC), appelée LHC à haute luminosité (HL-LHC), offrira une opportunité unique d'examiner de près le boson de Higgs mais aussi de découvrir des phénomènes rares qui vont au-delà du MS. Ceci est essentiel pour notre quête d'une compréhension plus profonde des particules fondamentales.

Le HL-LHC est prévu pour fonctionner jusqu'en 2040. Il représentera la phase finale de l'évolution du LHC, visant à augmenter considérablement sa luminosité par un facteur de 5 à 7 par rapport à sa conception initiale. Ceci se traduira par un taux de 140 à 200 collisions proton-proton toutes les 25 ns. Cela impliquera également une augmentation considérable d'un ordre de grandeur de la quantité de données à enregistrer par rapport à toute les données combinées déjà recueillies lors des précédentes campagnes du LHC depuis 2011. Pour accommoder cette luminosité accrue, la deuxième phase de mise à niveau du détecteur ATLAS, dite Phase-II, prévue pour 2026-2028, est essentielle. Le détecteur ATLAS servira à la fois à scruter le Modèle Standard de la physique des particules et surtout à sonder les domaines au-delà.

Le calorimètre à argon liquide (LAr) d'ATLAS mesure principalement les énergies des particules interagissant de manière électromagnétique produites lors des collisions proton-proton au LHC. L'électronique de lecture du calorimètre LAr sera remplacée lors de la mise à niveau Phase-II du détecteur ATLAS pour préparer l'ère du HL-LHC. La nouvelle électronique de lecture comprendra des FPGA (Field Programmable Gate Array ou réseaux de portes programmables sur site) haut de gamme pour le traitement des données en temps réel. Cette mise à niveau vise à permettre des taux de déclenchement et de lecture plus élevés ainsi qu'une granularité et une efficacité améliorées au niveau du déclenchement. Un algorithme de filtrage optimal (OF) est appliqué sur le signal électronique en impulsion pour reconstruire l'énergie déposée dans chaque cellule du calorimètre LAr. Cependant, l'augmentation anticipée de la lumi-

nosité du LHC entraînera une augmentation de l'activité de « pileup », c'est-à-dire de collisions proton-proton multiples simultanées et hors temps. Cette activité de « pileup » peut dégrader significativement la résolution en énergie obtenue à partir de l'algorithme OF. Cette électronique Phase-II de traitement aura une capacité de calcul bien supérieure à la précédente. Cela permettra la mise en œuvre dans les FPGA d'algorithmes plus élaborés basés sur des réseaux neuronaux (NN) pour reconstruire les énergies déposées dans le calorimètre LAr. Il a été démontré que les NN peuvent surpasser l'OF, notamment en présence de taux de « pileup » élevés. Les FPGA sont choisis pour leur capacité à traiter de grandes quantités de données avec une latence minimale, ce qui est une exigence cruciale pour le système de déclenchement d'ATLAS. L'implémentation du code en langage matériel, dit firmware, nécessite la conversion d'un modèle de réseau neuronal, écrit en langage informatique évolué, en un langage matériel spécifique pour les FPGA, tel que "High-Level Synthesis" (HLS) ou "VHSIC Hardware Description Language" (VHDL).

Cette thèse présente d'abord une description succincte du contexte scientifique théorique sous-tendant le programme HL-LHC. Le complexe d'accélérateur du CERN, le LHC, et le détecteur ATLAS sont ensuite présentés. Un accent particulier est mis sur le système de calorimétrie à argon liquide d'ATLAS qui produit les données traitées par un système électronique novateur s'appuyant sur des FPGA de dernière génération.

Cette thèse examine ensuite en profondeur le développement du firmware NN dans le langage HLS. Le HLS produit est incorporé dans la bibliothèque "High-Level Synthesis for Machine Learning" (HLS4ML), qui est une bibliothèque open source conçue pour automatiser le processus de production du firmware. Différents types de réseaux neuronaux pour les FPGA INTEL utilisés pour la reconstruction de l'énergie dans le calorimètre LAr ont été ajouté à HLS4ML. Ainsi, cette thèse introduit les réseaux neuronaux récurrents (RNN) utilisés pour la reconstruction de l'énergie, la bibliothèque HLS4ML et l'intégration des NN dans HLS4ML. L'utilisation des ressources des FPGA et les performances du firmware y sont également soigneusement analysées.

Cette thèse présente également un firmware de test conçu pour valider l'implémentation du NN dans le firmware du FPGA. Ce firmware de test permet l'injection de valeurs d'entrée dans le réseau neuronal embarqué sur le FPGA et la récupération des valeurs d'énergie calculées. Ces valeurs sont ensuite comparées aux résultats de simulation pour valider l'implémentation du NN. Le dispositif expérimental avec un kit de développement du FPGA INTEL Stratix 10, utilisé pour les tests en firmware, est décrit. Le firmware de test est également présenté, avec un aperçu de sa structure globale et une description des méthodologies et des techniques employées dans chaque composant de son architecture.

Les parties suivantes de cette synthèse vont maintenant décrire plus en détail le calorimètre, les méthodes de reconstruction en énergie et les développements et études effectués sur l'implémentation et les performances obtenues avec ces réseaux de neurones.

0.2 Les calorimètres LAr et la reconstruction d'énergie

Le calorimètre à argon liquide est utilisé pour mesurer les énergies et éventuellement identifier les particules le traversant, ciblant spécifiquement les particules ionisant l'argon liquide qui sert d'environnement actif. La collecte des électrons d'ionisations par un circuit électrique d'amplification est obtenue par des électrodes soumises à une haute tension. Comme illustré dans la figure 1, le détecteur présente une réponse caractérisée par une impulsion de forme triangulaire induite par la création instantanée d'électrons d'ionisation et leur dérive dans le champs électrique produit entre les électrodes et les absorbeur en plomb/inox, cuivre ou tungstène. Cette réponse présente une montée rapide suivie d'une descente subséquente d'environ 500ns dépendant du temps de dérive dans l'argon liquide. Dans le contexte du traitement du signal, l'utilisation d'un filtre analogique est efficace pour accomplir la tâche de filtration du signal. Après la mise en place du filtre, la réponse résultante présente une impulsion bipolaire, permettant de déterminer l'amplitude du dépôt et le temps nécessaire à sa formation sur le premier lobe. La réponse présente un haut degré de cohérence, la seule variation étant l'amplitude de son pic, directement liée au dépôt d'énergie et à la durée du processus de dérive. Ce signal est ensuite numérisé à la fréquence de collision de 40 MHz. Le processus de reconstruction implique de partir de nombreux points échantillonnés sur le signal et d'utiliser un algorithme de OF pour obtenir l'énergie transverse reconstruite.

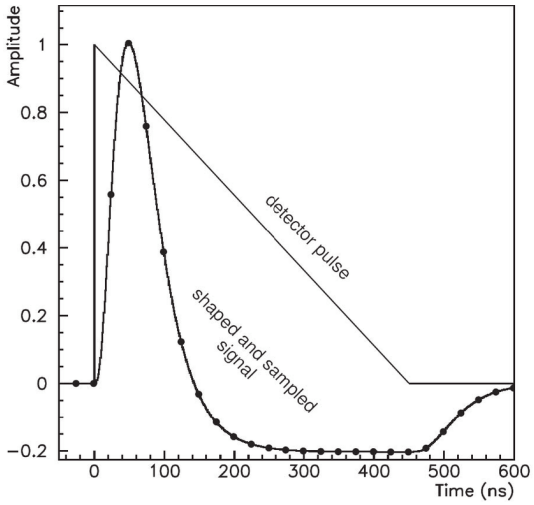


Figure 1 – Formes de l'impulsion actuelle du calorimètre LAr dans le détecteur et de la sortie du signal après le façonnage bipolaire. Les points représentent les échantillons séparés de 25 ns. [1]

0.3 La technique du filtre optimal

Les dispositifs de lecture électronique actuels utilisent une technique connue sous le nom de Filtre Optimale pour calculer la quantité d'énergie dans chaque cellule. La reconstruction

d'énergie à partir de l'OF est représentée par la formule 1.

$$E(t) = \sum_{i=t}^{t+n} a_i \cdot s_i, \tag{1}$$

L'algorithme en question utilise un processus de filtrage sur les échantillons provenant d'un convertisseur analogique-numérique (A/D), ce qui résulte en la production d'une série temporelle d'énergie. Les coefficients du filtre sont optimisés en fonction des caractéristiques des signaux physique obtenues en faisceau test, du bruit et des signaux de calibration. La sélection de la taille de l'échantillon (N), pour tenir compte de la précision nécessaire et des ressources informatiques, impacte sur la précision de l'estimation de l'énergie. Actuellement, OF utilise trois échantillons au niveau du déclenchement et quatre échantillons au niveau de la lecture principale pour réduire le bruit et le l'empilement des signaux. Les dépôts d'énergie réels sont associés à un paquet de collision (BCID) et corrélés à des intervalles de temps spécifiques. Cependant, la résolution du OF diminue lorsque de courts intervalles de temps sont utilisés en raison de la distorsion des formes de pulse causées par les empilements successifs des impulsions, comme représenté dans la figure 2. Ceci est particulièrement pertinent dans le contexte de HL-HLC, où il y a une augmentation à la fois du taux d'empilement hors et en temps et du nombre d'impulsions.

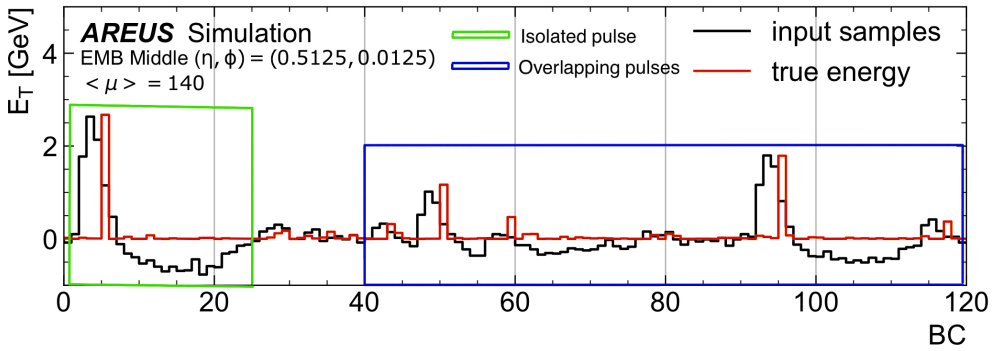


Figure 2 – Représentation d'une séquence d'échantillons (en noir) simulée par le logiciel AREUS. Le rectangle vert indique les impulsions d'une cellule dans les conditions du HL-LHC, tandis que le rectangle bleu met en évidence la présence d'impulsions crées à différents moments et se superposant. Pour améliorer la lisibilité, les vrais dépôts d'énergie transversale sont décalés de cinq cycles de croisement vers la droite et sont affichés en rouge. [2]

0.4 Reconstruction de l'énergie avec des réseaux de neurones récurrents

Afin de reconstruire les énergies, une architecture dite à fenêtres glissantes, représentée par la Figure 3, applique un Réseau de Neurones Récurrents (RNN) sur les échantillons. La séquence complète d'échantillons est divisée en séquences plus courtes qui se chevauchent, chaque séquence représentant un dépôt d'énergie. La point de calcul d'énergie pour chaque sous-séquence peut être choisie de manière à ce que la séquence contienne des informations

avant le dépôt considéré pour corriger les effets d'empilement et des échantillons après le dépôt pour reconstruire l'amplitude du signal. L'architecture de réseau choisie comporte une couche RNN et un neurone dans la couche de sortie. Quatre échantillons sont utilisés après le point de calcul de l'énergie, plus quelques autres dans le passé. Deux types de RNN ont été évalués, "Long Short-Term Memory" (LSTM) et Vanilla-RNN, qui sont respectivement une structure complexe et une simple. Les réseaux Vanilla-RNN et LSTM ont présenté des performances supérieures par rapport au réseau OF, tant en termes de biais moyen que de résolution, comme illustré dans la Figure 4. La plage contenant 98% des données a également été étudiée, et sa non symétrie indique la présence de caractéristiques non gaussiennes aux extrémités de la résolution, notamment pour des niveaux d'énergie faibles. La méthode OF a montré une tendance à mal récupérer les faibles énergies déposées, tandis que les réseaux neuronaux (NN) ont démontré une capacité plus efficace à reconstruire ces énergies.

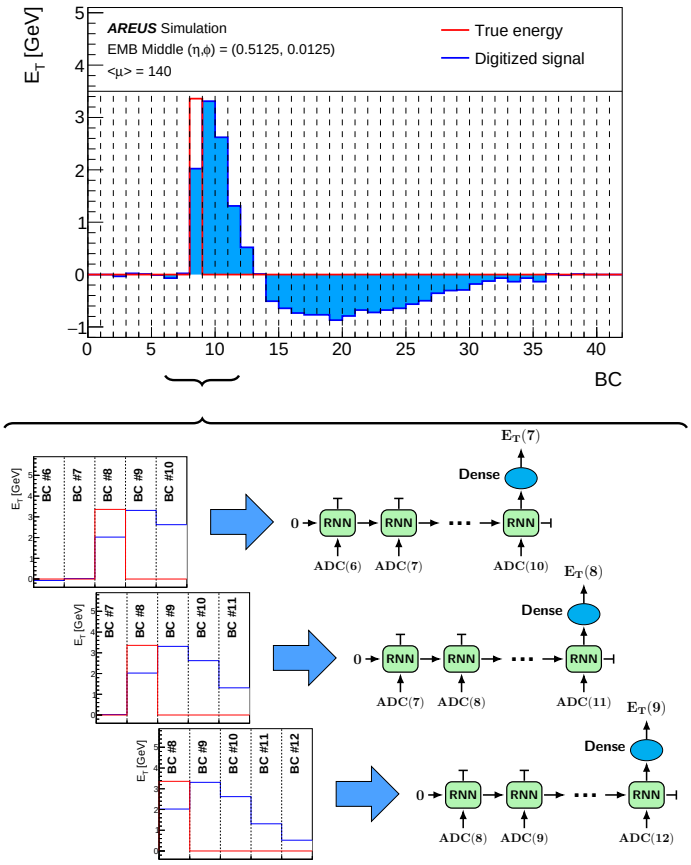


Figure 3 – Application de la fenêtre glissante des réseaux récurrents basés sur LSTM. À chaque instant, l'amplitude du signal des quatre BC passés et présents est entrée dans une couche LSTM. La dernière sortie de cellule est concaténée avec une opération dense comprenant un seul neurone et fournissant la prédiction de l'énergie transverse. [3]

Le modèle LSTM, notamment sa variante à cellule unique, possède une quantité substantielle d'unités de multiplication-accumulation (MAC), ce qui le rend inapplicable pour une implémentation sur un FPGA. La raison de cette limitation est que les FPGAs ont une capacité limitée

d'unités de multiplication-accumulation (MAC) par réseau, souvent de l'ordre de quelques centaines. Le Vanilla-RNN a présenté des performances comparables à celles du modèle LSTM, bien qu'il ait un nombre plus réduit de paramètres. En général, les performances des réseaux LSTM étaient quelque peu supérieures à celles du Vanilla-RNN. Cependant, l'implémentation LSTM (à cellule unique) nécessitait un nombre plus élevé de paramètres, environ cinq fois plus, par rapport au Vanilla-RNN.

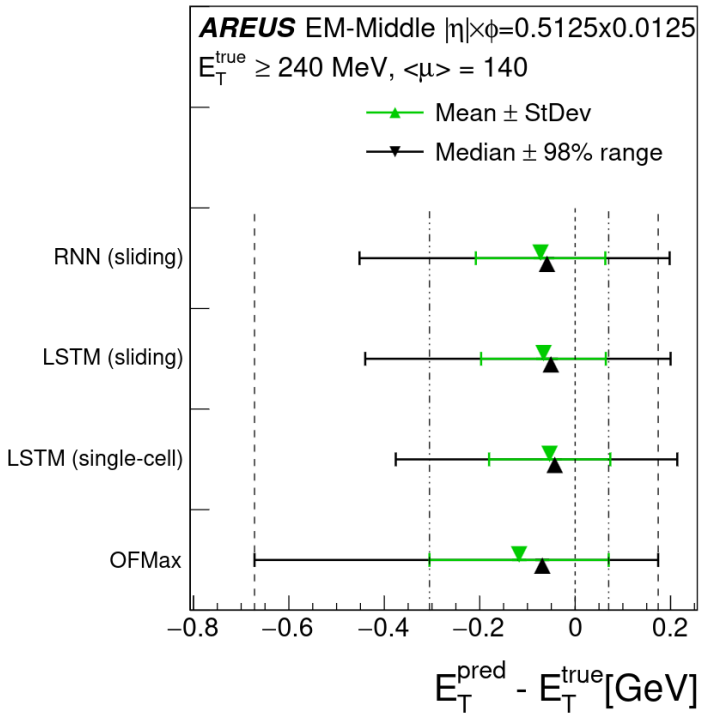


Figure 4 – Résolution de l'énergie transversale pour un filtre optimal et les différents algorithmes RNN. Les performances sont mesurées en comparant l'énergie transverse réelle déposée dans une cellule LAr du milieu du calorimètre électromagnétique tonneaux (EMB) ($\eta = 0,5125$ et $\phi = 0,0125$) à la prédiction faite par le RNN après avoir simulé l'impulsion échantillonnée avec le logiciel AREUS et en supposant $\mu = 140$. Seules les énergies supérieures à 3σ du niveau de bruit sont prises en compte. La moyenne, la médiane, l'écart-type et la plus petite plage qui inclut 98% des événements sont affichés. [3]

0.5 RNN dans HLS4ML pour la reconstruction d'énergie

Le HLS4ML est un outil puissant pour accélérer les applications d'apprentissage machine en transformant les modèles de haut niveau en implémentations matérielles ou firmware. Le logiciel est construit sous la forme d'une bibliothèque Python open-source qui peut traduire efficacement les modèles d'apprentissage machine, développés à l'aide de cadres populaires tels que TensorFlow, PyTorch ou Keras, en code HLS. Cette approche permet la génération automatique de circuits matériels optimisés qui peuvent être déployés sur des FPGA ou des ASIC pour obtenir de haute performance. La flexibilité de HLS4ML permet aux développeurs d'explorer facilement différentes options de conception et compromis, sans avoir besoin de

modifier le code. Le HLS4ML ne prenait pas en charge les réseaux de neurones récurrents (RNN) pour les FPGA de la société INTEL. Dans le cadre de cette thèse, afin de prendre en charge les modèles RNN dans HLS4ML, un code C++ générique a été utilisé pour implémenter LSTM et Vanilla-RNN. De nouveaux paramètres ont également été ajoutés pour personnaliser la fonctionnalité du programme. Les modèles RNN dans HLS4ML ont été exécutés pour tester la flexibilité, les performances et la résolution du firmware implémenté. Un LSTM avec des fonctions non linéaires tanh et sigmoïde et un Vanilla-RNN avec des fonctions d'activation ReLU (Rectified Linear unit) ont été examinés. Les figures 5 et 6 montrent la disparité entre la valeur obtenu à partir de Keras et la simulation dans Quartus HLS4ML en utilisant le LSTM et le Vanilla-RNN. Le LSTM et le Vanilla-RNN ont des valeurs efficaces, ou racine de la moyenne des carrés (RMS), de 0,0033 et 0,0031 [GeV], respectivement, avec un pic maximum à zéro. Cela prouve que l'implémentation RNN a fonctionné.

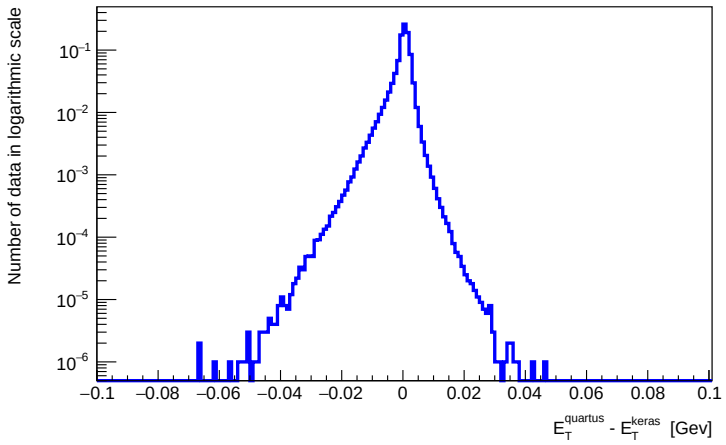


Figure 5 – Différence entre l'énergie transverse calculée avec Quartus et celle calculée avec Keras. Les valeurs de Quartus sont obtenues en utilisant l'implémentation LSTM dans HLS4ML.

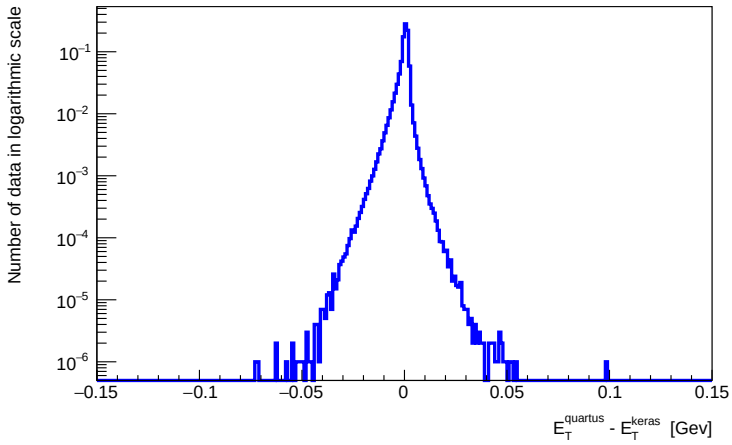


Figure 6 – Différence entre l'énergie transverse calculée avec Quartus et celle calculée avec Keras. Les valeurs de Quartus sont obtenues en utilisant l'implémentation Vanilla-RNN dans HLS4ML.

0.6 Évaluation des réseaux neuronaux récursifs embarqués sur le Stratix 10 DevKit d'INTEL

Le firmware dans HLS permet un prototypage rapide et une amélioration de l'architecture réseau. Mais cette thèse montre que l'implémentation HLS ne parvient pas à atteindre la fréquence nécessaire pour traiter dans un FPGA tous les 384 canaux, correspondant à trois cartes d'électronique de lecture du calorimètre électromagnétique, aux niveaux de latence requis. Le langage de description matériel VHDL associé au logiciel de conception QUARTUS, apportant beaucoup plus de fonctionnalités bas niveau et de flexibilité, a donc été utilisé pour améliorer les performance par rapport à l'implémentation HLS, répondant ainsi aux critères du firmware de la Phase II. Cette thèse décrit également comment tester le firmware NN in situ. Pour vérifier l'efficacité de l'implémentation NN sur la plateforme matérielle, un firmware de test a donc été développé.

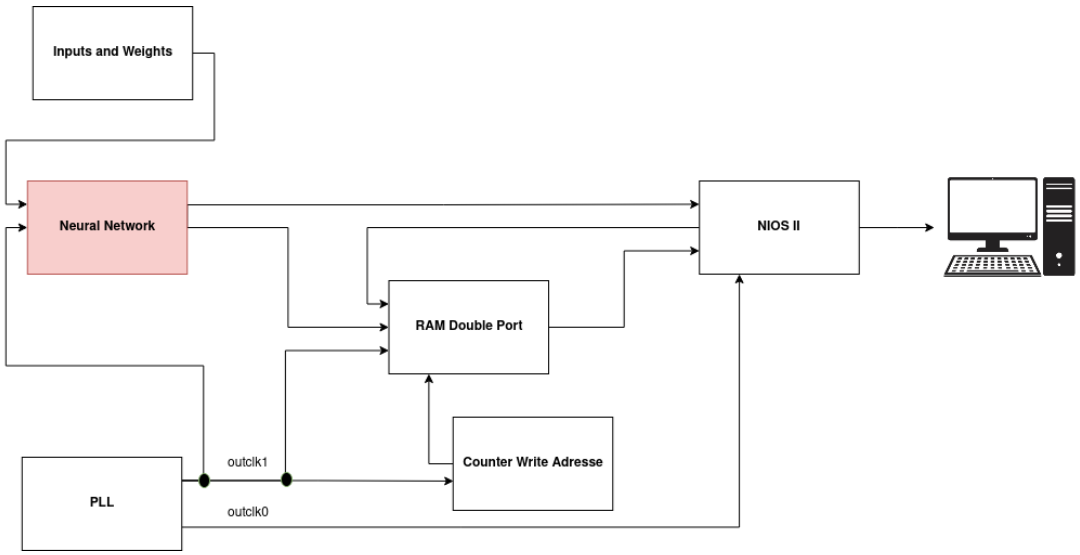


Figure 7 – Schéma simplifié des connexions entre le RNN et le firmware de test.

Ce firmware de test est un cadre qui permet l'injection de données dans le NN et la récupération des énergies transverses reconstruites par l'implémentation FPGA du firmware RNN. Comme le montre le diagramme simplifié dans la Figure 7, la sortie du firmware RNN passera par plusieurs composants critiques avant d'atteindre la sortie vers l'ordinateur. Les données d'entrée et les poids sont transférés de la RAM vers le réseau neuronal, qui calcule ensuite l'énergie et la transfère vers une autre RAM. Le processeur logiciel embarqué, NIOS, lit cette deuxième RAM et envoie les données correspondantes à l'ordinateur. Le firmware de test est conçu avec la modularité à l'esprit afin de faciliter le test de diverses formes de firmware de réseau neuronal. Le firmware NN est validé, et la différence entre Keras et NN, representée sur les figures 8 et 9, s'est avérée être inférieure à 0,1% pour les énergies supérieures au seuil de bruit, ce qui est cohérent avec les valeurs prédites par simulation. Il convient de noter que la validation du firmware de test a été effectuée en utilisant un ensemble contraint de valeurs. Chaque exécution

est limitée à 256 extractions de valeurs. Pour valider la stabilité du firmware sur des périodes prolongées à 40 MHz, un test supplémentaire essentiel est requis. Cela peut être accompli en comparant les énergies calculées dans le FPGA aux valeurs attendues contenues dans la RAM. Avant de pouvoir être testés sur la carte LASP finale, le NN et le firmware de test doivent également être transférés sur les FPGA Agilex.

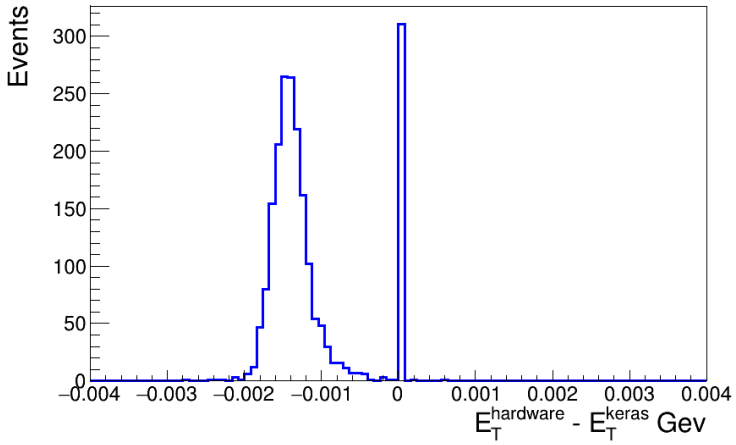


Figure 8 – Différence entre l'énergie transverse calculée avec le firmware et celle calculée avec Keras.

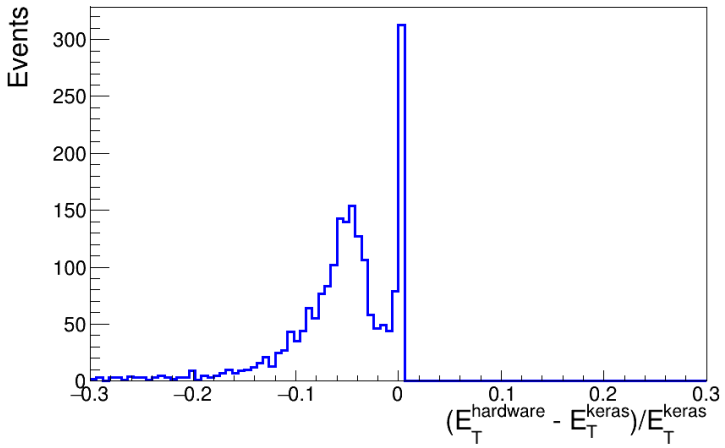


Figure 9 – SDifférence relative entre l'énergie transverse calculée avec le firmware et celle calculée avec Keras.

0.7 Conclusion

La haute luminosité du HL-LHC entraînera des collisions avec un empilement élevé, ce qui se traduira par une réduction de la précision de la reconstruction des énergies déposées dans le calorimètre LAr de l'ATLAS avec l'utilisation de l'algorithme de filtrage optimal. Pour atteindre les objectifs de l'expérience ATLAS, le développement de méthodes innovantes de reconstruction de l'énergie est impératif. La mise à niveau de la phase II, prévue pour 2026-2028, implique le remplacement du système de lecture du calorimètre LAr.

Cette mise à niveau implique le remplacement de l'électronique de lecture et de traitement des données LAr par des cartes d'électronique à base de FPGA de pointe, offrant davantage de capacités de traitement et permettant la mise en œuvre de réseaux neuronaux pour la reconstruction de l'énergie. Ces algorithmes de reconstruction de l'énergie sont destinés à être intégrés, pour la phase-II, dans ces cartes, appelées LASP et contenant deux FPGAs.

Les architectures Vanilla-RNN et LSTM ont été développées pour reconstruire l'énergie déposée dans les cellules LAr et ont montré des performances améliorées par rapport à l'algorithme de filtrage optimal dans les circonstances exigeantes du HL-LHC. Les RNN ont été conçus pour reconstruire précisément l'énergie des impulsions superposées, ce qui permet de remédier aux vulnérabilités de l'algorithme de filtrage optimal en particulier pour de faibles énergies.

Cette thèse présente la mise en œuvre du réseau neuronal développé en firmware à l'aide du langage HLS, son implémentation dans la bibliothèque HLS4ML et les performances obtenues. De plus, elle décrit la conception d'un firmware de test, utilisé pour valider la mise en œuvre du réseau neuronal dans le matériel. La bibliothèque HLS4ML traduit les réseaux neuronaux développés avec des outils courants d'apprentissage automatique (tels que Keras) en langage HLS pouvant être utilisé pour générer du firmware. Elle rationalise l'étalonnage et l'optimisation des paramètres pour les réseaux neuronaux afin de les adapter aux structures FPGA, accélérant ainsi la phase de prototypage.

Cependant, le package HLS4ML ne prenait pas en charge les architectures RNN et les FPGA INTEL. J'ai implémenté les architectures Vanilla-RNN et LSTM dans HLS4ML de manière à réduire l'utilisation des ressources sur les FPGA tout en maintenant les performances. Cette implémentation ciblait les FPGA INTEL avec Quartus HLS. J'ai utilisé cette implémentation pour scanner les paramètres du réseau neuronal et trouver les réglages optimaux pouvant s'insérer dans les ressources du FPGA. Cependant, HLS, et donc HLS4ML, s'est révélé manquer de flexibilité pour générer un firmware conforme à la spécification LASP.

Aussi, un firmware en VHDL a été développé pour optimiser davantage le firmware HLS et répondre aux spécifications. Bien que le développement en HLS ait été très utile dans les phases de prototypage pour atteindre une optimisation rapide, pour valider le firmware RNN sur les FPGA, j'ai créé et déployé un firmware de test ciblant le kit de développement Intel Stratix 10. Ce firmware de test fournit les valeurs d'entrée et les poids au firmware RNN. Il permet également l'extraction de l'énergie reconstruite par le firmware RNN au moyen d'un système d'opération embarqué sur puce (NIOS II) et d'une connexion JTAG. Le firmware de test a été utilisé pour valider le firmware NN, et les valeurs extraites du FPGA se sont avérées correspondre aux valeurs attendues de la simulation.

En conclusion, les RNN se présentent comme des candidats prometteurs pour la reconstruction de l'énergie dans le calorimètre LAr pendant l'ère du HL-LHC. Ils surpassent

l'algorithme de filtrage optimal et peuvent être intégrés de manière transparente dans les dispositifs FPGA. Le firmware du FPGA Stratix10 devrait être adapté au FPGA Agilex, qui servira de FPGA final pour la carte LASP. L'automatisation du processus de génération de code VHDL pourrait simplifier considérablement le processus de développement du firmware. Des améliorations supplémentaires dans HLS devraient également être envisagées, car le langage est récent et en expansion rapide.

INTRODUCTION

In 2012, a pivotal moment in particle physics occurred when both ATLAS and CMS teams at the Large Hadron Collider (LHC) revealed the discovery of the Higgs boson, the last puzzle piece of the Standard Model (SM) of particle physics, fundamentally enriching our comprehension of how particles like the W and Z bosons, quarks and leptons acquire mass via interactions with the Higgs field. The SM serves as a stepping stone toward a more comprehensive theory capable of elucidating enigmas like dark matter, dark energy, and matter-antimatter asymmetry. The High-Luminosity LHC (HL-LHC) phase is a unique opportunity to closely examine the Higgs boson and discover rare phenomena that extend beyond the SM. This is essential for our pursuit of a deeper understanding of the fundamental particles. Chapter 1 offers a overview at the Standard Model.

The HL-LHC, set to operate until 2040, represents the final phase of the LHC evolution, aiming to boost its luminosity by a substantial 5-7-fold increase compared to its original design, resulting in 140 to 200 proton-proton interactions every 25 ns. This endeavor also involves gathering a remarkable tenfold increase in data compared to all previous LHC runs combined. To accommodate this heightened luminosity, the Phase-II upgrade of the ATLAS detector, scheduled for 2026-2028, is essential. The ATLAS detector serves the dual purpose of scrutinizing the Standard Model of particle physics and probing the realms beyond it. Chapter 2 shows the complexity of the LHC accelerator, the ATLAS detector, their current capabilities, and future upgrade initiatives.

The ATLAS liquid argon (LAr) calorimeter primarily measures the energies of particles interacting electromagnetically produced during proton-proton collisions at the LHC. The readout electronics of the LAr Calorimeter will be replaced during the ATLAS phase II upgrade in preparation for the HL-LHC era. The new readout electronics will encompass high-end FPGAs for real time data processing. This upgrade aims to enable higher trigger and readout rates and enhanced granularity at the trigger level. To reconstruct the energy deposited in the LAr calorimeter, an optimal filtering (OF) algorithm is employed to analyze the electronic pulse signal. However, the anticipated increase in LHC luminosity will lead to a heightened occurrence of simultaneous multiple proton-proton collisions, commonly referred to as "pileup." This pileup activity can significantly degrade the energy resolution obtained from the OF algorithm. The phase II electronics will have higher computational capacity, enabling the implementation of neural network (NN) based algorithms in FPGAs to reconstruct the energies deposited in the LAr calorimeter. NNs have been shown to be able to outperform OF, specially when pileup

occurs. FPGAs are chosen for their capacity to process substantial volumes of data with minimal latency, a crucial requirement for the ATLAS trigger system. The hardware implementation requires the conversion of a NN model into an hardware language for FPGAs, as HLS or VHDL. This thesis investigates the development of the NN firmware in the HLS language. The produced HLS is incorporated into the HLS4ML library, which is an open source library designed to automate the firmware production process. Support for the NN and FPGA types used for energy reconstruction in the LAr calorimeter is added to HLS4ML. Chapter 3 introduces the use of recurrent neural networks (RNNs) for the energy reconstruction in the LAr colorimeter and the HLS4ML library which is an open source tool capable of automatically generating firmware code for NNs. This chapter details my work to implement the support for RNNs and for INTEL FPGAs in the HLS4ML library. It also shows the studies that I performed to optimize the resource usage in the FPGAs and the firmware computation resolution.

Chapter 4 describes a test firmware that I designed to validate the NN implementation in the hardware. The test firmware allows for the injection of input values into the neural network and the retrieval of computed energy values. These values are then compared with simulation results to validate the NN implementation. This chapter also introduces the development kit utilized for hardware testing, as well as an overview of the overall structure, methodologies, and techniques employed within each component of the test firmware architecture.

CHAPTER

1

STANDARD MODEL OF PARTICLE PHYSICS

Developed during the 1960s and 1970s, the Standard Model (SM) [4, 5, 6] is a comprehensive theoretical framework in the field of particle physics that provides a clear understanding of elementary particles and the fundamental forces that govern their interactions. It is considered one of the most successful theories in physics, having been tested and verified with great precision over the last few decades. The SM unifies the three of the four fundamental interactions in nature i.e. electromagnetic, the strong, and the weak forces, but leaves out the theory of gravity. However, despite its success, the SM is still incomplete and has limitations in explaining certain phenomena such as baryon asymmetry, dark matter, dark energy, and the origin of mass. The discovery of the Higgs Boson particle in 2012 [7] was a major achievement in the SM theory.

Section 1.1 provides an overview of the Standard Model (SM) and its composition of particles. In Section 1.2, Quantum Chromodynamics (QCD), Quantum Electrodynamics (QED), and the concept of electroweak unification are introduced. The role of the Higgs boson in electroweak symmetry breaking is explored in Section 1.3. Moving on to Section 1.4, the characteristics of proton-proton collisions at the Large Hadron Collider (LHC) are delved into. Finally, Section 1.5 discusses the limitations of the Standard Model and highlights the need for theoretical models beyond the Standard Model.

1.1 Fundamental Theory

The Standard Model of particle physics describes the fundamental building blocks of matter and their interactions. These particles are categorized as either fermions, which obey Fermi-Dirac statistics, or bosons, which are governed by Bose-Einstein statistics. Fermions make up matter and have half-odd-integer spin (e.g. spin $\frac{1}{2}$). They can be classified into quarks and leptons. Quarks and leptons are further classified based on the types of interactions they are subject to, with quarks being subject to the strong, weak, and electromagnetic interactions and

leptons to the electromagnetic and weak interactions, except for neutrinos, which are subject to only weak interactions. They exist in three generations of increasing mass, each with two quarks and two leptons. On the other hand, elementary bosons act as force carriers and are spin 1 particles, with the exception of the Higgs boson, which has spin 0. The Higgs boson is a scalar particle responsible for giving mass to the other SM particles. The SM provides a theoretical framework that captures the relationship between the fundamental interactions and the elementary particles. All fermions have an oppositely charged antimatter counterpart, and the bosons mediate the interactions between the fermions. The properties of different particles in the SM, such as their mass, charge, spin, and color, are summarized in figure 10.

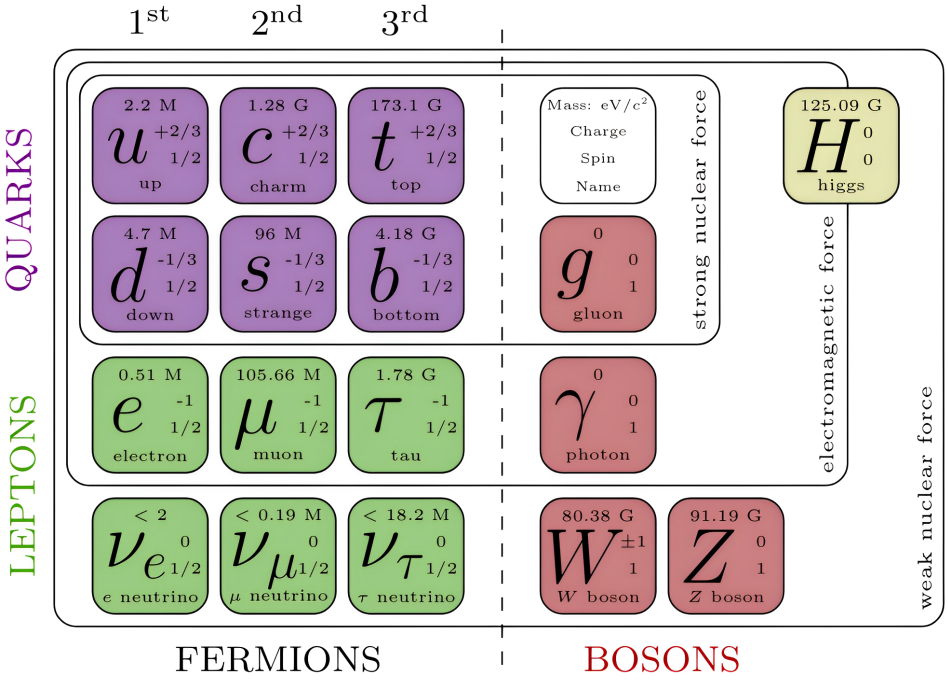


Figure 10 – Summary of SM elementary particles and their properties, including theorized graviton as a force carrier for the gravitational force.

A quark pair consists of an up-type quark and a down-type quark. Up-type quarks include up (u), charm (c), and top (t) quarks, while down-type quarks include down (d), strange (s), and bottom (b) quarks. Quarks carry an electric and a colour charge. In every quark generation, an up-type quark with an electric charge of $e = +\frac{2}{3}$ and a down-type quark with a charge of $e = -\frac{1}{2}$ are present. Colour confinement is the phenomenon where quarks exist only in bound states called hadrons, which can be composed of a quark-anti-quark pair (mesons) or three quarks (baryons). Although hadrons with four or five quarks have also been observed, these are considered to be exotic hadrons. In nature, only particles without colour charge are observed, which helps to explain the confinement of quarks within hadrons and why free quarks have never been observed.

Leptons are composed of the electron (e), muon (μ), and tau (τ), each accompanied by its respective neutrino: the electron-neutrino (ν_e), muon-neutrino (ν_μ), and tau-neutrino (ν_τ).

Leptons have a negative electric charge of $e = -1$, while neutrinos are electrically neutral, as shown in table 1 [8, 9].

	Flavour	**Electric charge (Q)**	**Weak isospin (I_3)**	**Weak hypercharge (y)**
Left-handed helicity fermions	ν_e, ν_μ, ν_τ	0	$+1/2$	-1
	e^-, μ^-, τ^-	-1	$-1/2$	-1
	u_L, c_L, t_L	$+2/3$	$+1/2$	$+1/3$
	d_L, s_L, b_L	$-1/3$	$-1/2$	$+1/3$
Right-handed helicity fermions	e_R^-, μ_R^-, τ_R^-	-1	0	-2
	u_R, c_R, t_R	$+2/3$	0	$+4/3$
	d_R, s_R, b_R	$-1/3$	0	$-2/3$

Table 1 – Summary of helicity (right or left-handed) and different flavors. Right-handed helicity corresponds to a positive projection of the spin vector onto the momentum vector, while left-handed helicity corresponds to a negative projection.

The vector bosons, known as gauge bosons, serve as the carrier particles for the fundamental forces. These force-carrying particles require fermions to possess the corresponding charge to each type of interaction as shown in the table 2 [10, 11]. The strong interaction relies on gluons (g), which are massless spin-1 particles. There are a total of eight colors of gluons known. Since strong interactions require the presence of color, the g is able to mediate quarks. The electromagnetic interaction requires an electrical charge (Q) and involves the exchange of a neutral, massless spin-1 particle called the photon (γ). Thus neutrinos are the only particles that do not electromagnetically interact or interact with photons. The weak interaction is mediated by either the massive charged boson ($W^\pm$) or the massive neutral boson (Z). The peculiarity of the weak force is that it is the only force that does not exhibit mirror symmetry, meaning it is not invariant under parity transformations. It can be divided into two components: left-handed particles, which interact with the weak force while having a spin polarization opposite to their direction of motion, and right-handed particles, which interact with the weak force while having a spin polarization in the same direction as their motion.

Bosons	**Interactions**	**Mass**	**Charge(e)**
g	strong	0	0
γ	electromagnetic	0	0
$W^\pm$	weak	80.38 ± 0.012 GeV	± 1
Z	weak	91.19 ± 0.0021 GeV	0

Table 2 – Summary of the interactions, mass, and charge properties of the gauge bosons presented in the Standard Model.

In addition to fermions and bosons, an additional particle called the Higgs boson exists,

characterized by its scalar nature and spin-zero and electric neutrality. More details about the Higgs are explained in section 1.3.

The behavior of the SM is described by the theoretical framework known as Quantum Field Theory (QFT). It is built upon a set of symmetries that describe the fundamental particles and their interactions. In the SM, these particles are seen as excitations in a quantum field that spans all points in space and time.

The mathematical foundation of the SM is based on the unitary product group represented by the equation 1.1.

$$G_{SM} = \mathrm{SU}(3)_C \otimes \mathrm{SU}(2)_L \otimes \mathrm{U}(1)_Y \tag{1.1}$$

Where $SU(3)_C$ represents the symmetry group for strong interactions, with C referring to color which represents the conserved quantity associated with this group. The gauge group $SU(2)_L \otimes U(1)_Y$ governs the unified electroweak interactions. Within this group, $SU(2)_L$ is associated with Weak Isospin (I_3), which involves only left-handed helicity particles, indicated by the subscript L. The subscript Y stands for Weak Hypercharge.

1.2 Fundamental Interactions

Within the framework of the SM, each of the fundamental forces (electromagnetic, weak and strong interactions) is described by its own QFT. However, the electromagnetic and weak forces have been combined and unified under the Electroweak theory.

1.2.1 Strong Interaction

Quantum Chromodynamics (QCD)[12] is a theoretical framework, founded on Yang-Mills theory [13], that explains the strong interactions between quarks through the mediation of gluons, using the $SU(3)_C$ symmetry group. Within this framework, there are three charges associated with the group, referred to as color charges. Gluons possess both color (r, g, b) and anti-color (r^-, g^-, b^-) charges. The combination of color charges leads to a total of 9 possibilities, with 8 possible gluon configurations resulting from this, and 1 color singlet (rr^-, gg^-, bb^-) state that is disregarded due to the absence of observed infinitely ranged strong interactions. Since there are eight group generators associated with the $SU(3)_C$ algebra, just as many gauge fields need to be introduced. Since gluons carry color charges themselves, they can interact not only with quarks but also with other gluons. This characteristic sets them apart from photons, which are electrically neutral. The direct coupling of gluons to one another gives rise to additional vertices in addition to the fundamental quark-gluon vertex. Specifically, there are two types of primitive gluon-gluon vertices: three gluon vertices and four gluon vertices. These

vertices are represented in figure 11 using Feynman diagrams.

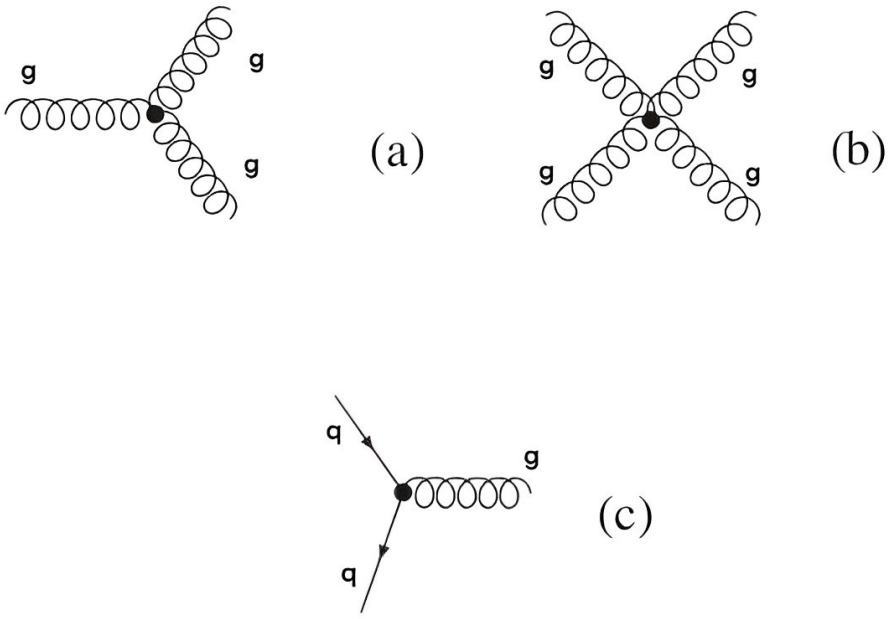


Figure 11 – Representation of the potential interaction vertices within QCD are depicted as follows: (a) represents a three-gluon vertex , (b) represents a four-gluon vertex, and (c) represents a quark-gluon interaction. [12]

An intriguing characteristic of QCD is that the direct coupling of gluons to one another adds complexity to chromodynamics compared to electrodynamics. It allows for intriguing phenomena like the existence of glueballs. At short distances, the coupling strength between quarks weakens, allowing them to behave like independent particles, exhibiting asymptotic freedom. Asymptotic freedom implies that inside particles such as protons, for instance, the quarks move about without significant interactions. Conversely, as distances increase, the coupling strength intensifies, preventing the isolation of quarks. As a result, quarks are always confined within hadrons, meaning they are perpetually bound together. This phenomenon is known as confinement.

1.2.2 Electromagnetic Interaction

Quantum Electrodynamics (QED) is a comprehensive theory that underlies our understanding of electromagnetism. Within the framework of QED, charged particles have the ability to emit and absorb virtual photons as represented in figure 12 using Feynman diagram. Although these virtual photons cannot be directly observed, they play a significant role in influencing the probabilities of observable events. From a mathematical perspective, QED is described as an abelian gauge theory with a symmetry group known as U(1). This theory is defined on Minkowski space, which represents a flat spacetime. The electromagnetic field serves as the

gauge field, facilitating the interaction between charged spin-1/2 fields. The QED Lagrangian, which describes the interaction between the spin-1/2 field and the electromagnetic field using natural units, gives rise to the action. In accordance with Noether's theorem [14], symmetries are associated with conserved quantities. In the case of QED, the conserved quantity is the electromagnetic charge.

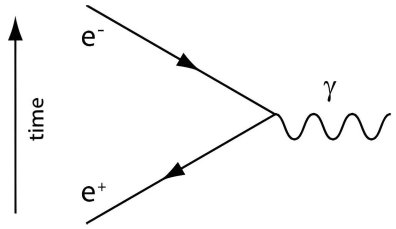


Figure 12 – Feynman diagram representing any charged particles scattering process. [12]

Re-normalization provides a means to equate the observed mass with the effective mass while upholding established principles of physics. By and large, this technique eliminates infinite values from the theory by assimilating them into existing free parameters, all the while maintaining fidelity to known physical principles.

1.2.3 Electroweak theory

The electroweak theory, known as the Glashow-Weinberg-Salam (GWS) theory, was developed based on extensive experimental data on charged-current weak interactions. This theory focuses on the leptonic sector of the $SU(2)_L \otimes U(1)_Y$ framework, where the charged-current weak interaction only affects left-handed leptons. The GWS theory employs gauge theory, which describes the formal interactions involving electroweak gauge fields without altering the system's dynamics. In other words, these interactions arise from symmetries. The gauge fields associated with the $SU(2)_L \otimes U(1)_Y$ symmetry group correspond to four massless mediating bosons: the weak isospin fields W_1, W_2, and W_3, and the weak hypercharge field B. This symmetry is referred to as the electroweak symmetry.

The electroweak gauge symmetry combines the coupling constants of $SU(2)_L$ (weak isospin symmetry) with $U(1)_Y$ (weak hypercharge phase symmetry). This unification leads to the definition of the electric charge, denoted as e, as given by Equation 1.2. As a result of this mixing, the physical gauge bosons γ, Z, and $W^{\pm}$ emerge as combinations of the W and B fields.

$$Y = 2(e - I_3) \tag{1.2}$$

On electroweak gauge symmetry, it is necessary for the particles in the model to be massless. This is because the inclusion of mass terms for bosons and fermions in the Lagrangian

would lead to a violation of local gauge invariance. However, experimental observations have indisputably shown that particles like electroweak gauge bosons and fermions do possess mass. To reconcile this contradiction, the mechanism of spontaneous electroweak symmetry breaking is introduced, allowing for the incorporation of both gauge and fermion mass terms.

1.3 Spontaneous Symmetry Breaking and the Higgs Mechanism

The electroweak and QCD theories only allow for the existence of massless particles. In 1964, the Brout-Englert-Higgs model [15] was formulated as a solution to the mass problem. A crucial addition to the Standard Model is the Higgs boson. In the Electroweak Lagrangian (EW), gauge fields do not possess mass notably due to the $SU(2) \times U(1)$ symmetry. Adding mass to the bosons would break this gauge symmetry, while adding mass to fermions is prohibited by the requirement of chiral symmetry. The Higgs mechanism involves the spontaneous symmetry breaking, as represented in figure 13, separating the local symmetries of the Lagrangian from the symmetries of the quantum vacuum.

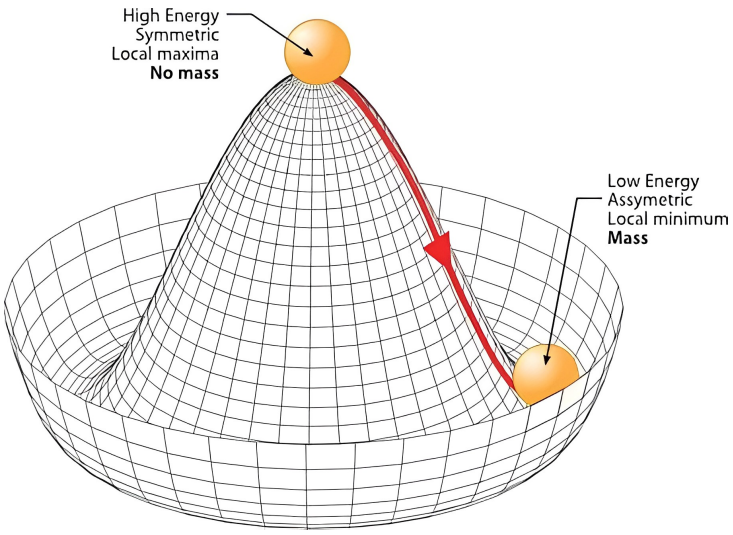


Figure 13 – Illustration of the Higgs potential configuration in relation to the component of the Higgs field ϕ.

The naive introduction of non-zero masses for fermions is impossible due to the way the violation of parity symmetry of the W and Z bosons is modeled. The GWS model solves this problem by unifying electromagnetic and weak interactions into a single electroweak interaction, whose combined symmetry is spontaneously broken by the universe's ground state. The spontaneous symmetry breaking of $SU(2) \times U(1)Y$ to $U(1)$em gives rise to the gauge bosons Z and $W^{\pm}$ acquiring mass through the Higgs mechanism, which arises from a combination of the W and B fields generating a self-interaction of the Higgs field ϕ. This field is configured in such a way that it possesses degenerate ground states with a non-zero vacuum expectation value.

The remaining degree of freedom from the breaking becomes a real massive scalar field called the Higgs boson H, which gains coupling to fermions. This coupling interaction is proportional to the fermion's mass.

1.4 Characteristics of proton–proton collisions

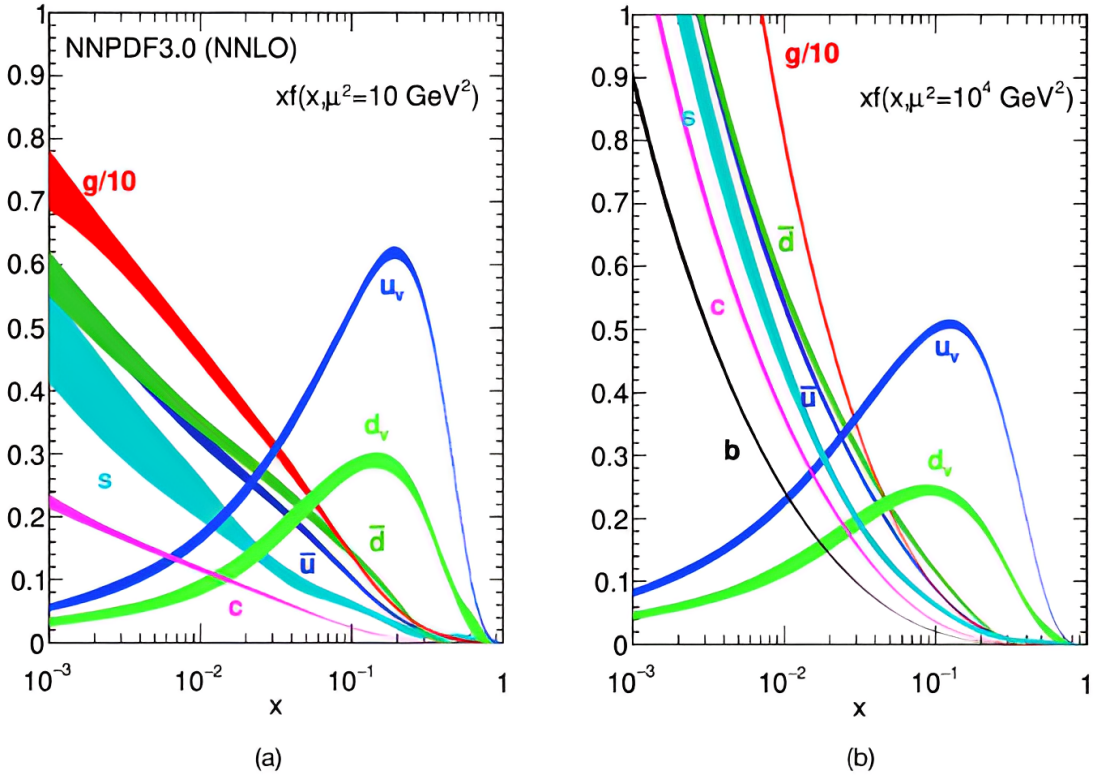


Figure 14 – Graph of the longitudinal momentum fraction (x) multiplied by the PDF as a function of x, for two momentum transfer scales: $Q^2 = 10, \mathrm{GeV}^2$ (a) and $Q^2 = 10^4, \mathrm{GeV}^2$ (b). [16]

To study the physics of elementary particles at the smallest distance scales and highest energies, it is needed to investigate proton-proton collisions to generate hard-scatter interactions. The proton consists of three primary valence quarks and an indefinite number of virtual quarks, known as sea quarks, which arise through quantum fluctuations. During high-energy proton collisions, the interactions take place between the internal components of the protons, collectively known as partons. The individual partons carry the total momentum of the proton, as they are free to move independently within the proton due to asymptotic freedom. The Parton Distribution Functions (PDFs) model how momentum is distributed between the partons and are used to describe the proton's initial state. However, PDFs are too complex to be calculated theoretically and must be found experimentally using deep inelastic scattering experiments. The figure 14 shows that the probability of a valence quark or gluon carrying a large fraction of the proton momentum is significantly higher than for any other partons, but their dominance decreases at

higher momentum scales. To test the theory, particle collisions are essential, and one way to achieve the highest collision energy is through proton-proton (pp) collisions.

1.5 Beyond Standard Model Physics

The SM has been incredibly successful in explaining the particle content of the universe and has provided accurate descriptions of quantum electrodynamics (QED), QCD and the electroweak theory. However, despite its achievements, the SM is not yet a complete theory and lacks the ability to explain certain phenomena as dark energy, dark matter, gravity, Neutrino masses and matter–antimatter asymmetry. These shortcomings hint at the existence of physics beyond the Standard Model (BSM), which suggests the presence of additional particles and interactions that are not accounted for in the current framework. There are observed phenomena that remain unexplained and inadequately addressed by the Standard Model, indicating the need for further advancements in our understanding of fundamental physics. The HL-LHC program, which facilitates proton-proton collisions, aims not only to investigate the properties of the Higgs boson but also to explore the existence of new particles. The ATLAS Phase II project is dedicated to recording these collisions and necessitates a Phase II upgrade. Within this context, this thesis is focused on enhancing the Calorimeter's performance capabilities.

CHAPTER

2

ATLAS EXPERIMENT

In 1954, the European Council for Nuclear Research (CERN) was established with the objective of unifying European physicists and distributing the rising expenses of nuclear physics research. The location chosen for constructing the first accelerator of the complex was the Geneva area due to its central position in Europe, neutrality during World War II, and existing international institutions. CERN is home to the Large Hadron Collider (LHC), the most extensive accelerator in the complex. The LHC is a massive and advanced circular synchrotron accelerator that is presently the most powerful experimental machine for examining fundamental particles.

In Section 2.1, the LHC is discussed, covering the accelerator design, injection chain, and the existing experiments at the LHC. Section 2.2 focuses on the ATLAS detector, which is the largest general-purpose particle detector at the LHC. The upcoming high-luminosity upgrade of the LHC prompts a significant upgrade for the ATLAS detector, which is detailed in Section 2.3.

2.1 The Large Hadron Collider

The LHC is a particle accelerator located at the Franco-Swiss border near Geneva, operated by the European Organization for Nuclear Research (CERN). It consists of a 26.7 km underground ring that collides charged particles such as protons and heavy ions at incredibly high energy and velocity. Bunches of approximately 10^{11} protons collide with a gap of 25 ns between two consecutive LHC bunches, known as bunch-spacing, and researchers use particle detectors installed at specific points of the ring to investigate numerous Standard Model (SM) processes, rare processes, and new physics scenarios that may arise at high energies. The LHC was approved in 1994 and built from 1998 to 2008 to replace the Large Electron-Positron (LEP) collider. It had its first operational run (Run 1) in March 2010, with successful collisions of proton beams at a center-of-mass energy of 7 TeV, and the collision energy was successfully

boosted to 8 TeV in 2012. The LHC and the detectors were upgraded from 2013 to 2015 for higher center-of-mass energy and luminosity, and the Run 2 phase, from mid-2015 to the end of 2018, had a 13 TeV energy. The LHC and the detectors underwent another upgrade (LS2) between 2019 and 2022 [17], and the Run 3 phase happened in 2022, with a center-of-mass energy increase from 13 to 13.6 TeV energy recorded. Following Run 3, the LHC will enter an upgrade period to enable it to run with a significant increase in the instantaneous luminosity. The High-Luminosity LHC (HL-LHC) is expected to start in 2026 and end in 2028 and is anticipated to bring researchers ten times more collected data during HL-LHC data-taking, which is planned to start in 2029, than the LHC.

2.1.1 The accelerator design and injection chain

The LHC's primary purpose is to accelerate protons, as they are easy to obtain, stable, and experience less energy loss through synchrotron radiation than electrons or positrons. To produce the protons used in collisions at the LHC, hydrogen gas is ionized using a powerful electric field. The protons then undergo a chain of pre-accelerators before reaching the LHC, which is the final component of the CERN accelerator chain represented by figure 15.

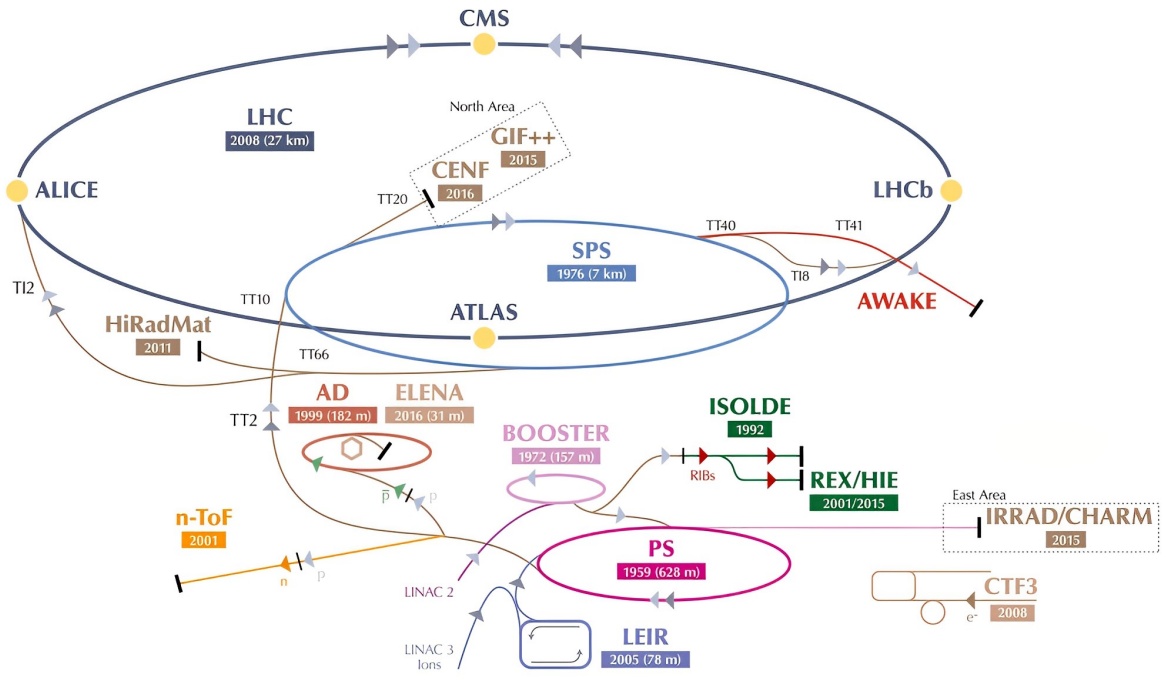


Figure 15 – Schematic view of CERN accelerator complex and locations of experiments at LHC. [18]

The protons, produced by stripping valence electrons of from hydrogen atoms (H^-), are first accelerated by a 30 m long linear accelerator, the Linear accelerator 4 (Linac4), to an energy of 160 MeV and after being squeezed into high-intensity bunches, they enter the Proton Synchrotron Booster (PSB). The PSB is a circular machine composed of four super imposed

150 m synchrotron rings. The PSB increases the protons' energy up to 1.4 GeV and guides them into the first ever CERN synchrotron accelerator of 638 m in circumference, the Proton Synchrotron (PS), where they are accelerated to an energy of 25 GeV. In the final step of the pre-acceleration chain, the protons are transported to the Super Proton Synchrotron (SPS), which is the second largest accelerator at CERN with 7 km in circumference, in order to significantly rise their energy up to 450 GeV before injecting them into the LHC. The proton beams injected into the LHC from the SPS are separated into 2808 bunches that are evenly spaced by 25 ns, each containing 1.15×10^{11} protons and are transferred to the LHC's two beam pipes, where they travel in opposite direction beams. After approximately 20 minutes, the proton beams in the LHC are stabilized and given their final shape while their energy is increased to 6.8 TeV in Run 3. The LHC consists of eight straight sections and arcs, and the particles are accelerated by radiofrequency (RF) cavities located in the straight sections. The protons are accelerated to the required energy by an electric field of 5 MV/m that oscillates at 400 MHz and is generated from a series of 16 superconducting the RF cavities that is operated at a temperature of 4.5 K together with a potential of 2 MV. The proton beams in the LHC are kept in a curved trajectory by a variety of superconducting dipoles magnets that is produced using 1232 superconducting dipole magnets, about 15 m long each. These magnets are cooled using superfluid helium to a temperature of 1.9 K and nominally generate a field of 8.3 T. Also, 858 quadrupoles magnets are used to focus the beams and keep the protons away from the pipe walls. The LHC also employs additional magnetic multipoles to counteract divergence-inducing effects that lead to proton beam defects.

An accelerator is characterized by the energy of the accelerated particles and its luminosity. The luminosity ($\mathscr{L}$) of the accelerator is completely determined by the beam parameters and is defined as the ratio of the beam intensity (I_{beam}) to the collision area (A).

$$\mathscr{L} = \frac{I_{beam}}{A} \tag{2.1}$$

When considering a circular accelerator, assuming Gaussian beam profiles for two beams travelling in opposite directions, the luminosity can be mathematically expressed as

$$\mathscr{L} = \frac{N_1 N_2 f_r n_b}{4\pi\sigma_x\sigma_y} S_\phi. \tag{2.2}$$

Where, N_1 and N_2 represent the number of particles contained in each bunch per beam. The parameter n_b indicates the number of bunches present in each beam as they circulate around the ring. The σ_x and σ_y represent the horizontal and vertical convolved beam widths, respectively. Additionally, S_ϕ is the geometrical loss factor that takes into account the non-zero crossing angle at the interaction point, and f_r is the revolution frequency. Multiple proton-proton ($p-p$) collisions take place when particle bunches intersect in the LHC. The degree of such interactions

per bunch crossing (BC) can be measured by a quantity known as pileup, which is represented by $<\mu>$, and is given by:

$$\mu = \frac{\mathscr{L}\sigma_i}{f_r n_b}, \tag{2.3}$$

where σ_i represents the total inelastic cross-section for a $p-p$ interaction. During the Run 3 data taking period in 2022, the LHC delivered a total integrated luminosity of 38.5 fb^{-1}, as illustrated in figure 16. Due to alterations in the pileup configuration, the average pileup for the Run 3 dataset was 42.5. The distribution of $<\mu>$ is presented in figure 17.

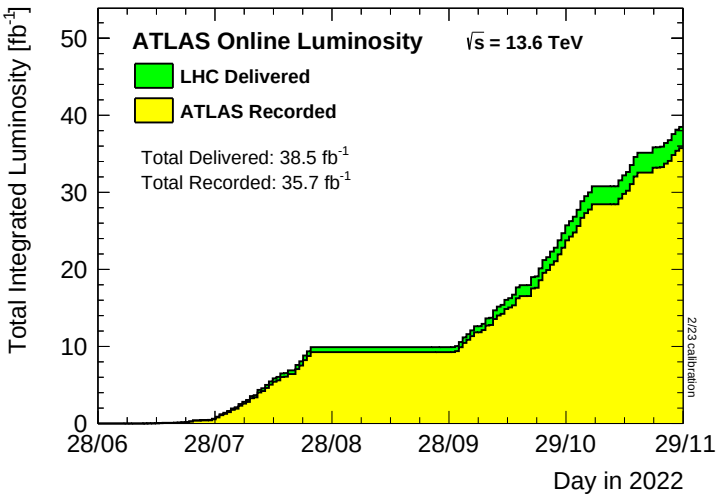


Figure 16 – Comparison between the cumulative luminosity delivered to (green) and the one recorded by ATLAS (yellow) during stable beams for $p-p$ collisions at 13.6 TeV centre-of-mass energy in 2022. [19]

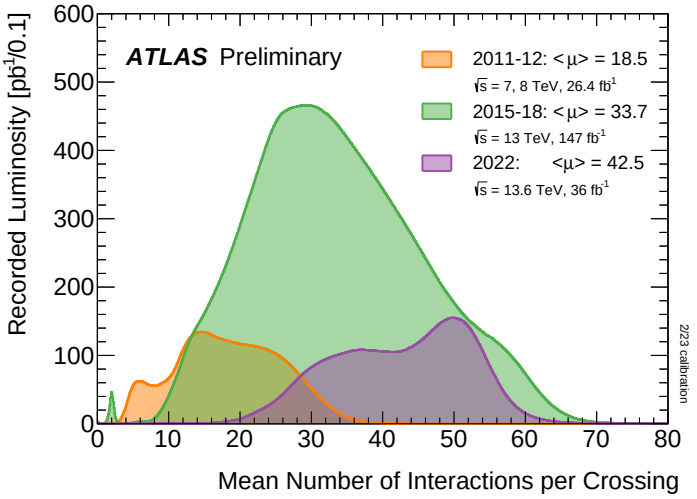


Figure 17 – Distribution of the average pileup $\langle\mu\rangle$. The average pileup during the full Run 3 amounts to $\langle\mu\rangle = 42.5$.

2.1.2 Experiments at the LHC

There are four primary experiments, commonly known as detectors, situated in caverns around the LHC ring. The ATLAS, CMS, LHCb, and ALICE have the potential to reveal new particles and physical phenomena that challenge our current understanding of particle physics and the nature of the universe. Each experiment has its own purpose in physics, which is described below:

- ATLAS (A Toroidal LHC ApparatuS) is a particle detector designed to study a wide range of physical phenomena, including the Higgs boson, dark matter, and supersymmetry. It is capable of detecting electrons, muons, photons, and particle jets, as well as more exotic particles such as top quarks and W and Z bosons.
- CMS (Compact Muon Solenoid) is a particle detector similar to ATLAS. CMS also searches for evidence of particles and interactions that do not fit into the standard description of particle physics, such as dark matter and supersymmetry.
- LHCb (Large Hadron Collider beauty) is a detector specialized in the study of b quarks, also known as beauty quarks. LHCb is used to measure the properties of these quarks and study the interactions in which they are involved, particularly CP (charge-parity) violations in matter.
- ALICE (A Large Ion Collider Experiment) is an experiment designed to study the physics of quark-gluon plasma, which is the form of matter believed to have existed in the early moments after the Big Bang. ALICE is used to study collisions of heavy ions at the LHC and the particles produced in these collisions, including free quarks and gluons.

2.2 The ATLAS detector

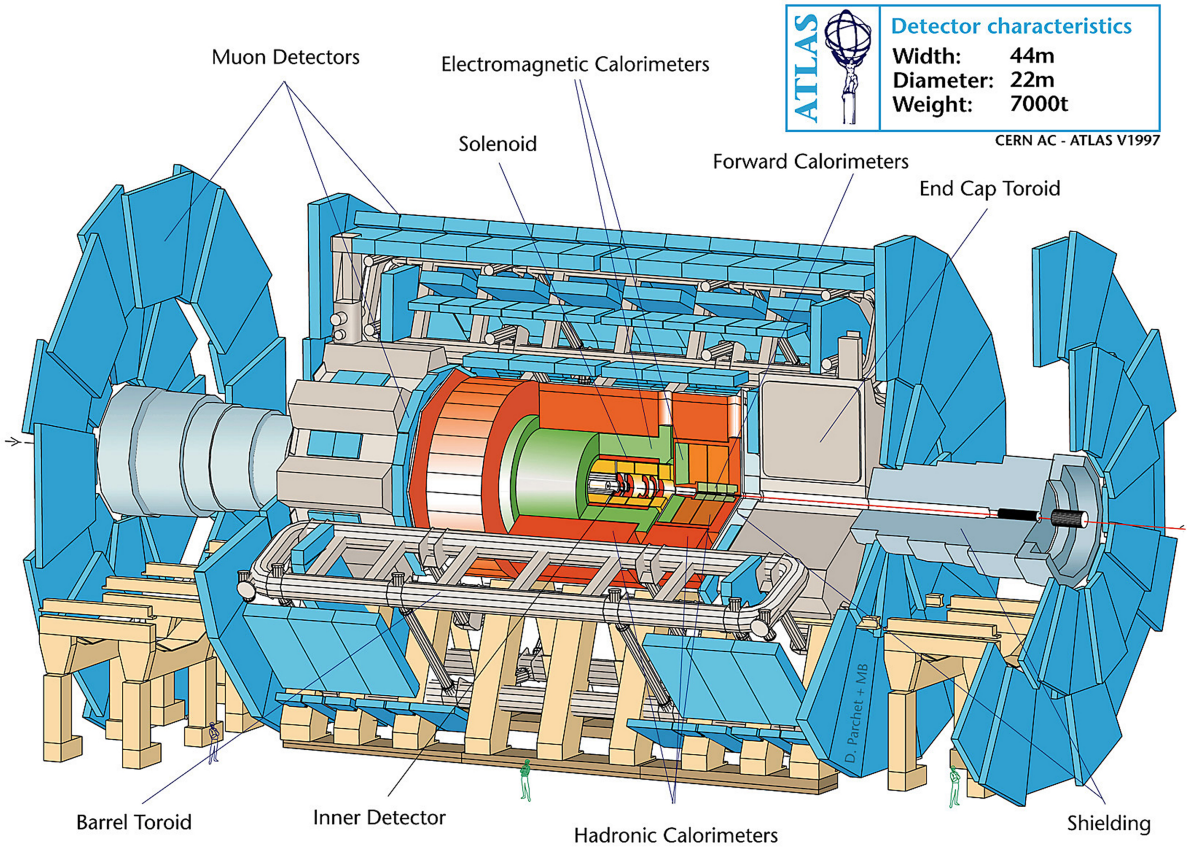


Figure 18 – Schematic view of the ATLAS detector. [20]

The ATLAS detector, situated at an interaction point in the Large Hadron Collider (LHC), is a versatile detector designed to study a broad range of particle physics phenomena, such as dark matter, extra dimensions at the TeV scale, and the Higgs boson. The detector, represented in Figure 18, has a symmetric cylinder shape with a length of approximately 44 meters and a height of 25 meters. It provides nearly full solid angle coverage of 4π and weighs approximately 7000 tons. The detector is positioned in the UX15 underground cavern, which almost entirely houses the detector, but due to the radiation levels present in UX15, the back-end electronics are housed in a different cavern, the USA15, which is not subject to radiation. The ATLAS detector is divided into subdetector systems arranged in layers surrounding the interaction point. These systems consist of an Inner Detector (ID), electromagnetic and hadronic calorimeters, and an outer Muon Spectrometer (MS) placed on three large superconducting air-core toroidal magnets. High granularity, fast electronics readout, efficient object reconstruction, and resolution are the key features of this detector. Moreover, the ATLAS detector made a notable discovery of the Higgs boson in 2012.

2.2.1 The ATLAS coordinate system

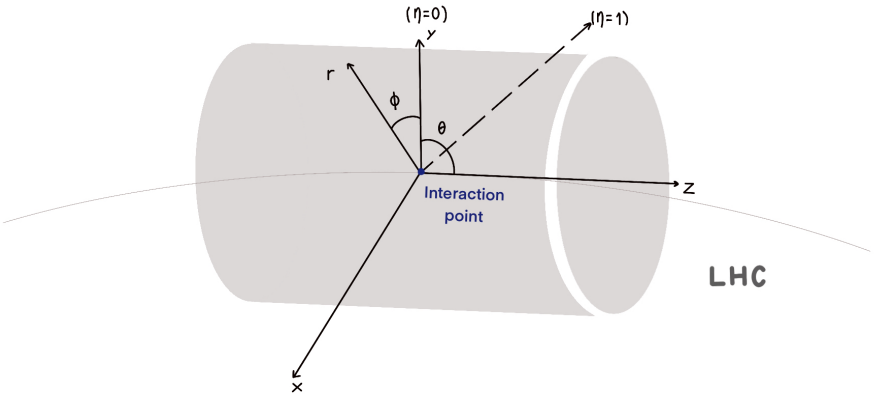


Figure 19 – The ATLAS detector's system of coordinates.

The ATLAS detector uses a coordinate system, represented by figure 19, to determine directions and positions within the detector. The origin of the system is placed at the nominal interaction point, with the z-axis along the beam direction, the x-axis pointing towards the center of the LHC ring, and the y-axis pointing upwards. The transverse plane, which is perpendicular to the beam line, is described using cylindrical coordinates $(r-\phi)$, where $r=\sqrt{x^2+y^2}$ measuring the distance from the beam line and ϕ is denoting the azimuthal angle measured from the x-axis around the beam line with a range of $[-\pi,\ \pi]$. The polar angle θ is defined in $[0,\ \pi]$ with respect to the z-axis. Since the distance in θ depends on the longitudinal longitudinal momentum $p_z=|p|\sin(\theta)$ and a particle with energy E, the rapidity y is defined as

$$y=\frac{1}{2}\ln\left(\frac{E+p_z}{E-p_z}\right). \tag{2.4}$$

For particles traveling at ultra-relativistic speeds, where the mass is insignificant compared to its energy, the pseudo-rapidity η is a useful approximation for rapidity y. This can be calculated using the formula:

$$y\approx\eta=-\ln\left[\tan\left(\frac{\theta}{2}\right)\right]. \tag{2.5}$$

The utilization of pseudo-rapidity proves advantageous due to its invariance under Lorentz boost along the beam axis, which ensures that the difference $\Delta\eta$ remains constant. As a result, the distance between two points in the $\eta-\phi$ plane can be determined as

$$\Delta R=\sqrt{\Delta\eta^2+\Delta\phi^2}. \tag{2.6}$$

2.2.2 The Inner Detector

ATLAS's tracking system, also known as the Inner Detector (INDET) and represented in Figure 20, is situated closest to the beam axis. It consists of three sub-detectors: the Pixel Detector, the Semiconductor Tracker (SCT), and the Transition Radiation Tracker (TRT), as depicted in Figure 21.

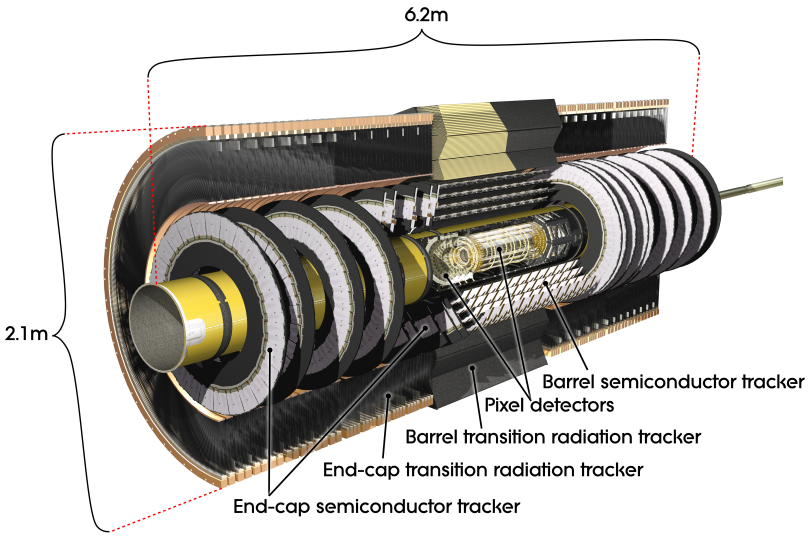


Figure 20 – Cut view of the ATLAS Inner Detector. [21]

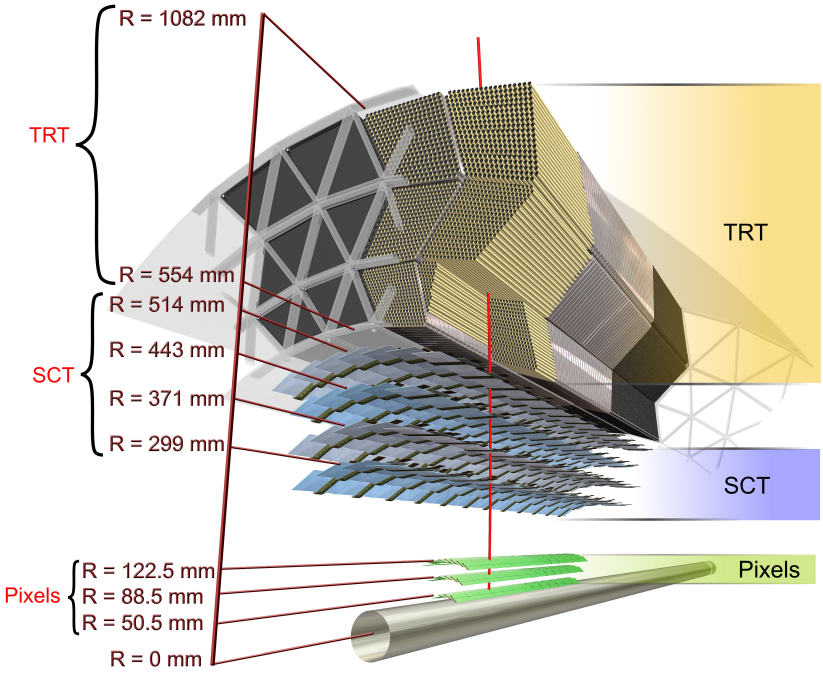


Figure 21 – Transverse section and structural elements of the ATLAS Inner Detector. [22]

The sub-detectors are housed in a cylindrical casing that measures 6.2 m in length and 1.2 m in radius along the beam pipe, covering a pseudorapidity range of $|\eta| \leq 2.5$. An axial magnetic field of 2 T is generated by a thin superconducting solenoid magnet surrounding the INDET, deflecting the trajectories of charged particles within the system. By detecting particle interactions with the material, the INDET reconstructs particle trajectories and determines their charges and momenta. The momentum resolution of the INDET can be mathematically expressed as a function of the particle's transverse momentum p_T, using the equation 2.7. The term <a> represents the intrinsic resolution of the momentum measurement in a magnetic field, while <b> accounts for multiple-scattering effects.

$$\frac{\Delta p_T}{p_T} = a p_T \oplus b \tag{2.7}$$

$$\frac{\Delta p_T}{p_T} = 0.05\% p_T[GeV] \oplus 1\% \tag{2.8}$$

2.2.2.1 The Pixel Detector

The Pixel Detector is an essential component of the INDET. It serves multiple functions, including particle tracker with a high resolution, pattern recognition that fulfils the requirements on vertex and reconstructing interaction point and secondary vertices. This sub-detector is the most interior component of the INDET, and it provides precise measurements in the region closest to the interaction site. The detector layers is composed of four barrel and six end-caps. The end-caps are rings with sensor modules positioned perpendicular to the beam axis for hermetic coverage. The barrel consists of four cylindrical layers arranged around the beam pipe, where the innermost layer is the Insertable B-Layer (IBL), as represented in figure 22. The IBL was installed in 2012 to enhance the reconstruction of secondary vertices of decaying B-hadrons.

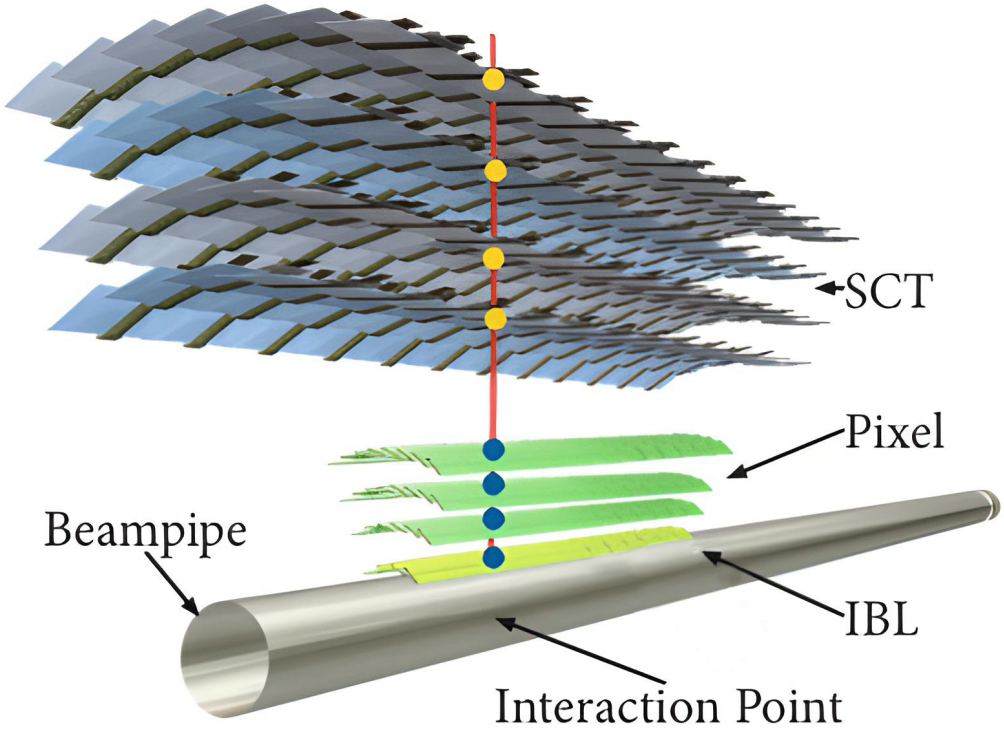


Figure 22 – The ATLAS Silicon system including the Insertable B-Layer.

The Pixel Detector is comprised of n-type silicon pixel sensors with an external dimension of 19 mm 63 mm and a thickness of 250 μm. Individual sensors have pixel dimensions of modules are 50×400 μm^2 for external layers and 50×250 μm^2 for the innermost layer. The detector has approximately 80.4 million read-out channels, with the four-barrel layers located at nominal radii of 33.25 mm, 50.5 mm, 88.5 mm and 122.5 mm from the beam axis. The resolution of the pixel detector is presented in figure 23 and 24.

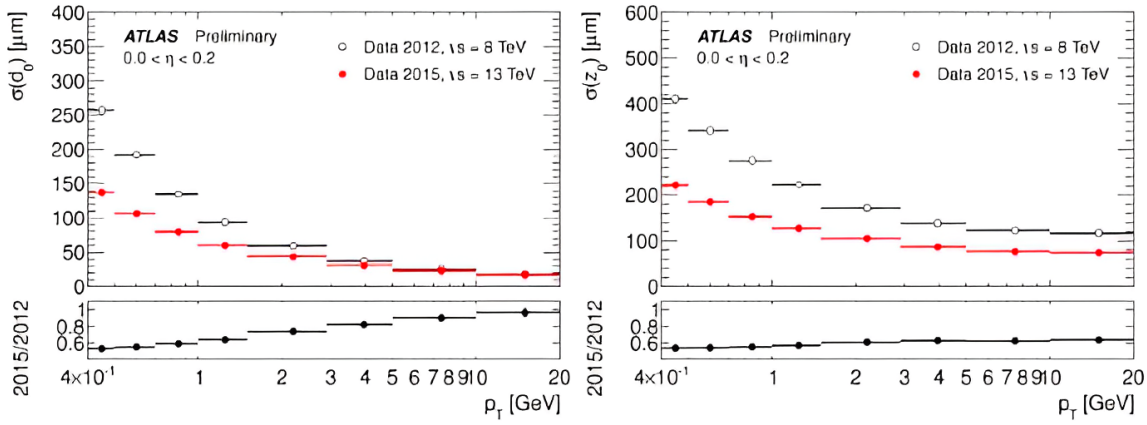
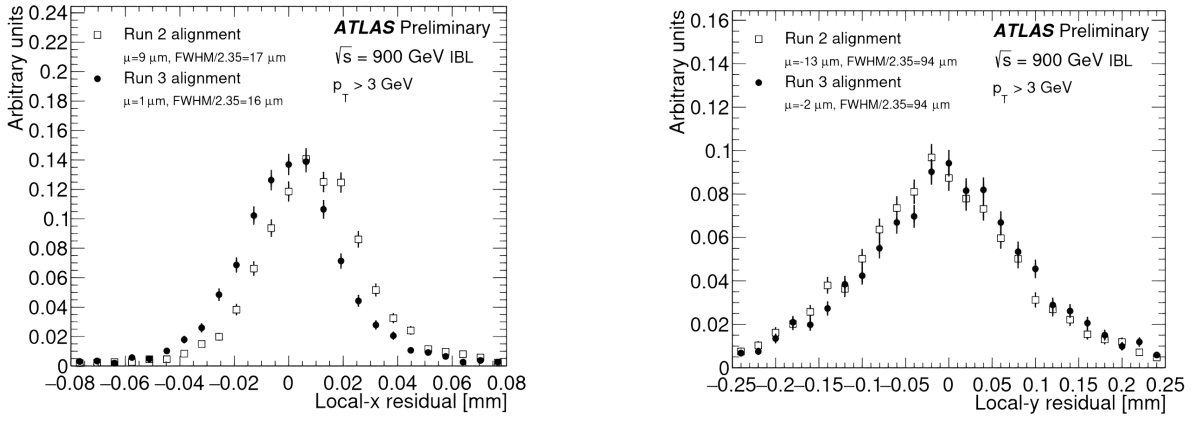
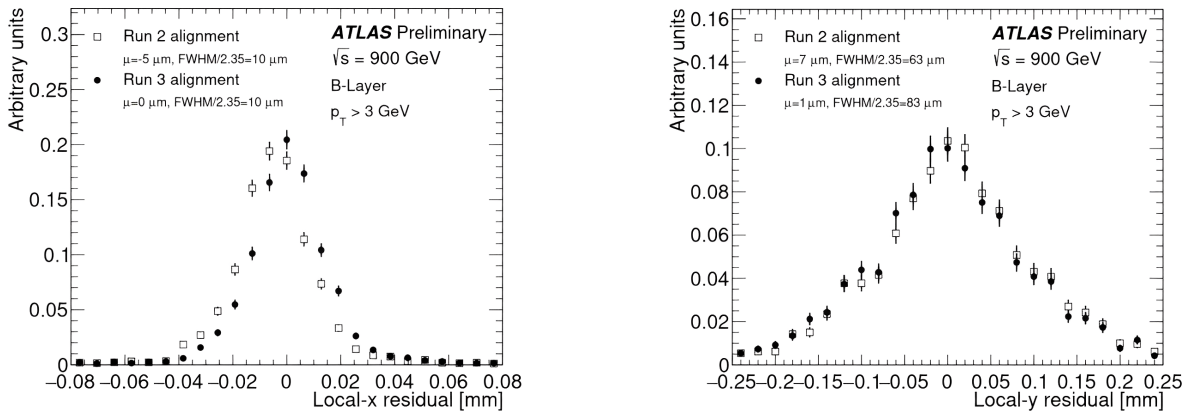


Figure 23 – The resolution of d0 (left) and z0 (right) depends on the value of p_T and it differs between ATLAS Run-1 (without IBL) and Run-2 (with IBL). [23]



(a) (b)



(c) (d)

Figure 24 – Difference between ATLAS Run-2 and Run-3 alignment. Track-hit residuals in the IBL in the local X (a) and Y (b) coordinates and in the B-Layer in the local X (c) and Y (d) coordinates. [24]

2.2.2.2 The Semiconductor Tracker Detector

The Semiconductor Tracker is an important component of the precision tracking part of the Inner Detector surrounding the Pixel Detector. The SCT uses micro-strip sensors and is divided into a barrel and end-cap part. The barrel is comprised of four concentric layers and the end-caps consist of nine disks, providing coverage up to $|\eta| < 2.5$. The SCT utilizes individual modules made up of double-sided silicon micro-strip sensors, each strip having a nominal dimension of 80 μm in $r-\phi$ and 126 mm in z. The SCT-Detector is installed at a distance of 299 mm to 514 mm from the beam axis, providing a z resolution achieved by mounting strips on the two sides of a module under a small stereo angle of 40 mrad. With a total number of readout channels in the SCT is approximately 6.3 million and having an intrinsic resolution of $17 \times 580\ \mu m^2$ per space point in $r-\phi$ and z respectively, the SCT provides an average of four measurements per track, making it an important tool for accurate particle detection. The average number of SCT hits can be seen in figure 25 and the hit efficiency for each sector of the semi conductor tracker detector is shown in figure 26.

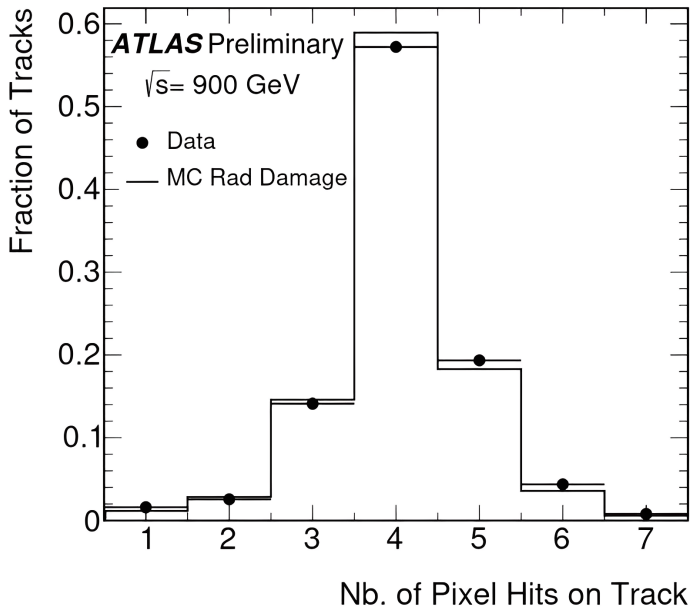


Figure 25 – Distributions of the average number of SCT hits associated to selected particle tracks as a function of the pseudo-rapidity, η, of the tracks in data (filled points with error bars) and simulation (continuous line).

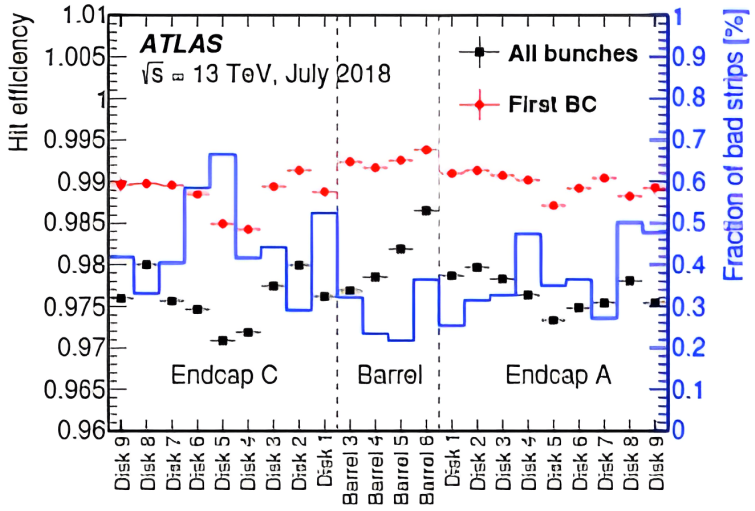


Figure 26 – The hit efficiency of the SCT detector was measured during a standard p-p collision run in July 2018. The black squares and red dots represent the values measured using all bunches and only the first BC, respectively. The first bunches were used to avoid any pileup effect from past BC. The hit efficiency was averaged over each SCT barrel layer and endcap disk. The fraction of bad strips was also superimposed. [25]

2.2.2.3 The Transition Radiation Tracker Detector

The Transition Radiation Tracker is a crucial part of the ATLAS INDET, situated outside the Pixel Detector and SCT volumes. The barrel of the TRT contains 50,000 long polyimide drift tubes, each divided into two parts at the center and filled with a gas mixture composed by Xenon (70%), Carbon Dioxide (27%) and Oxygen (3%), while the end-caps contain an additional 320,000 radial straws. This large number of straws allows for an average of 36 measurements per track, significantly improving momentum resolution. The gaps between the straws are filled with a transition radiator that emits transition radiation when charged particles pass through it, allowing particles identification. By filling the straws with gas consisting mainly of xenon, the particles passing through the straws ionize the gas inside the tube, resulting in a current flowing along the wires and leading to detection. Due to multiple leaks in the flexible active gas exhaust pipes that emerged during Run 1 and Run 2, operating the entire detector with the baseline xenon-based gas mixture became excessively expensive. Consequently, for Run 3, an argon-based gas mixture is being utilized throughout the entire barrel, and within a few endcap wheels on one side of the detector [26].

The TRT plays an important role in extending tracks built in the inner part of the tracker, improving momentum resolution and providing $r-\phi$ information with an accuracy of 130 μm. With approximately 351,000 readout channels in total, the TRT compensates for the lower precision per point of the silicon detectors with a larger track length and high number of hits (figure 27), making it an essential component for momentum measurement in the ATLAS INDET. The average number of TRT hits can be seen in figure 28.

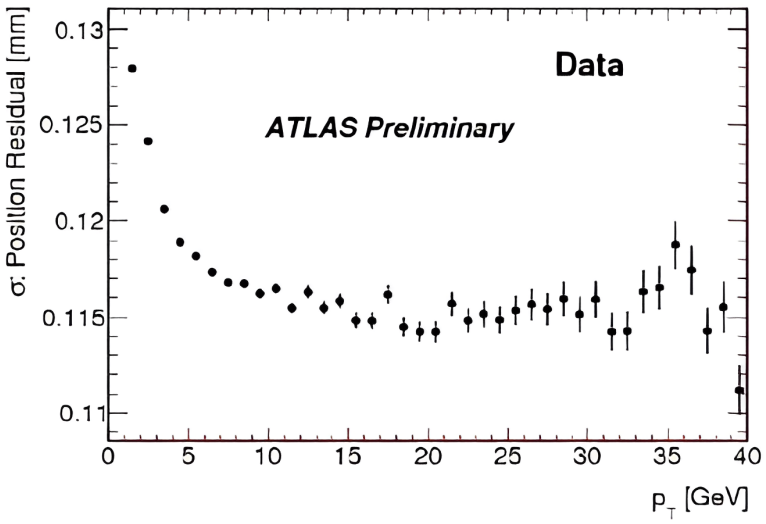


Figure 27 – The width of position residuals in the TRT barrel can be expressed as a function of track P_T. As track P_T decreases, the scattering of the track increases and results in a rise in residuals. [27]

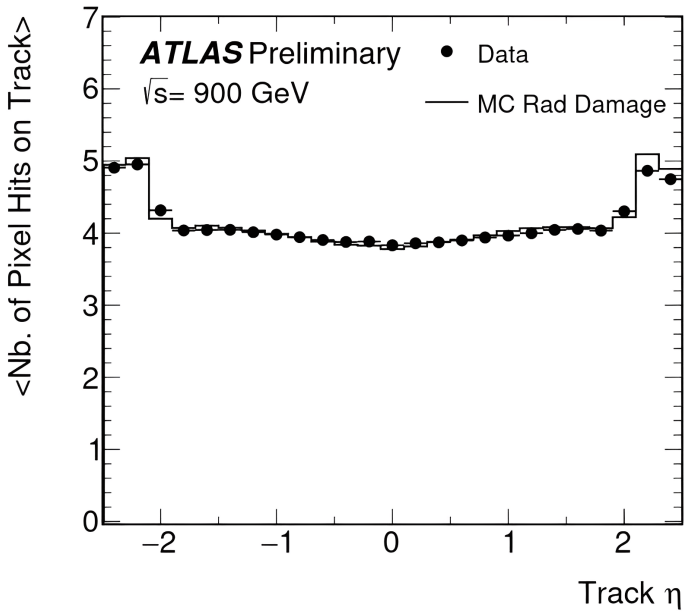


Figure 28 – Distributions of the average number of TRT hits associated to selected particle tracks as a function of the pseudo-rapidity, η, of the tracks in data (filled points with error bars) and simulation (continuous line).

2.2.3 The Calorimeters

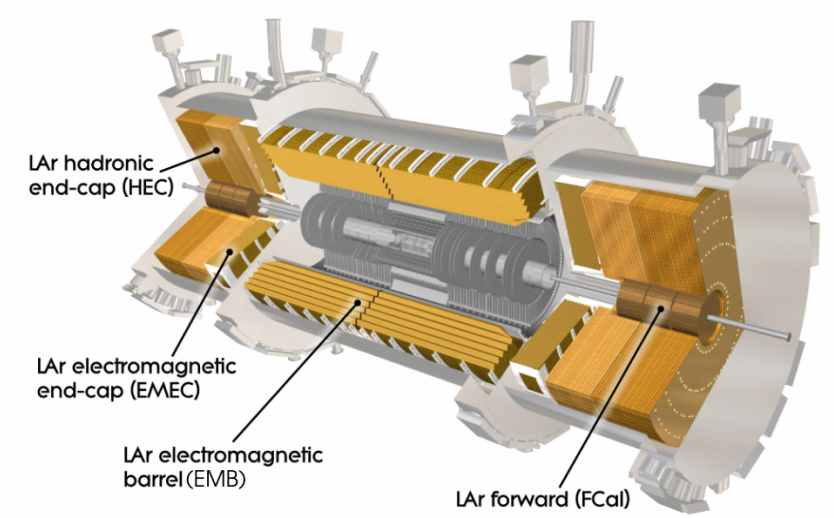


Figure 29 – Schematic view of the ATLAS calorimeters. [28]

The ATLAS detector features a complex system of calorimeters that surround the inner detector and the solenoid magnet. Located outside of the INDET with $|\eta| < 4.9$ and full ϕ coverage around the beam axis, these calorimeters are made of heavy materials like lead and steel, which can absorb particles generated from collisions and measure their energy. As particles cross the calorimeters, they gradually lose energy, producing secondary particles known as particle showers. By measuring the total energy of these showers, the incident particle's energy can be reconstructed. Most particles, including charged and neutral particles, are stopped inside the calorimeters, except for muons and neutrinos. The calorimetry system of ATLAS consists of three different sampling calorimeters (electromagnetic, hadronic and forward calorimeter represented by the figure 29), each with alternating layers of produce the particle shower with the measurement the deposited energy and active material for both absorption and measurement.

Although the configuration of the ATLAS calorimeter system allows for precise reconstruction of the shower evolution in all dimensions, it comes at a cost to energy resolution. The energy resolution in a calorimeter can be described using Equation 2.9:

$$\frac{\sigma_E}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c \tag{2.9}$$

which accounts for the sampling term, noise term, and constant term. The sampling term includes statistical fluctuations of the shower and the signal, along with inefficiencies of energy deposition in the active material. The noise term takes into account the effects of electronics noise and pileup, while the constant term accounts for systematic effects such as energy leakage and non-uniformity of energy response/collection. An excellent calibration procedure is required to minimize the constant term, which limits the performance of the calorimeter system at high

energies.

2.2.3.1 Electromagnetic Calorimeter

The ATLAS electromagnetic calorimeter (ECAL) is a sampling calorimeter that operates in an energy range from GeV to the TeV scale and uses liquid Argon (LAr) as its active medium. LAr fills the gaps between the absorber layers made of lead and is maintained at around 88 K. The accordion geometry of the ECAL makes it possible to cover the entire ϕ range without an instrumentation gap. The ECAL consists of one barrel and two end-caps, each made of two wheels.

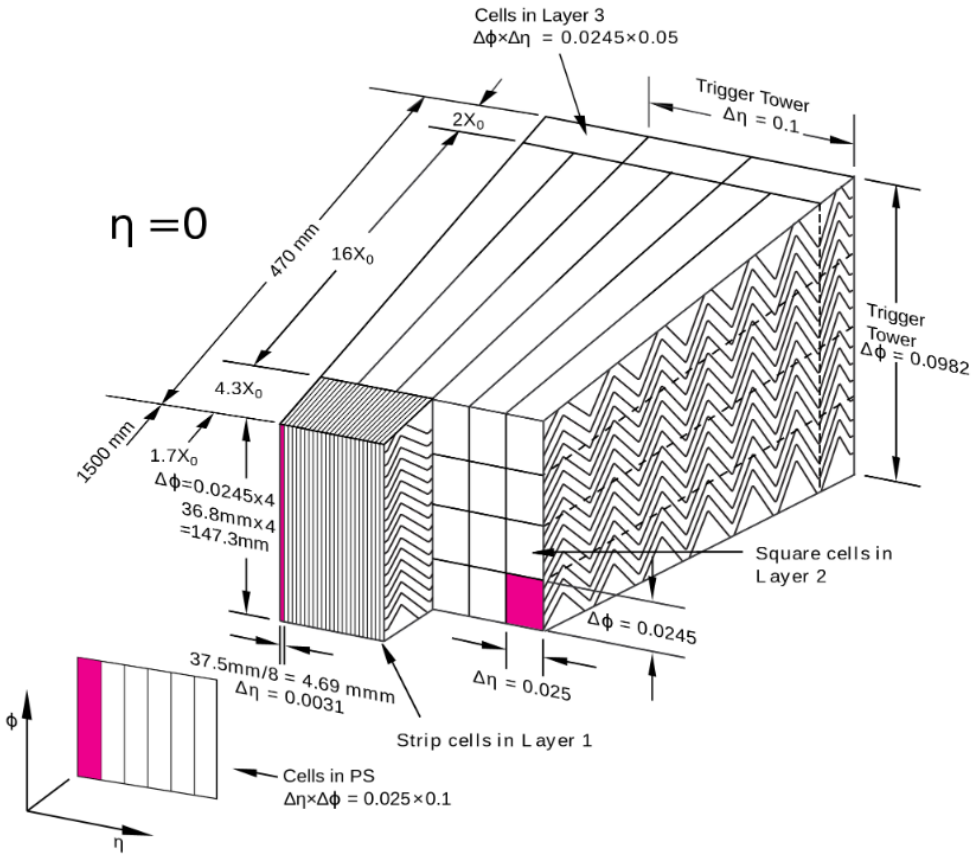


Figure 30 – Schematic view of the LAr calorimeter barrel showing the arrangement of single cells in different layers. [29]

$$\frac{\sigma_E}{E} = \frac{10\%}{\sqrt{E}} \oplus \frac{170\ [MeV]}{E} \oplus 0.7\% \tag{2.10}$$

The barrel ECAL has three layers, which decrease in granularity and cover a pseudo-rapidity range up to $|\eta| < 1.475$. The end-caps cover a region within $1.375 < |\eta| < 3.2$ and are segmented into three longitudinal layers in their outer wheels and two layers with a coarser granularity in both directions in their inner wheels. The ECAL has been designed to achieve an energy resolution of equation 2.10. A schematic view of a barrel module is illustrated in figure 30. In this detector, the electrons undergo bremsstrahlung, they generate secondary photons,

and these photons, in turn, produce electron-positron pairs. The resulting secondaries continue to produce additional particles via the same mechanism, giving rise to a cascade of particles with decreasing energies, which is commonly known as an ectromagnetic (EM) shower. Each charged secondary particle will ionize the liquid argon along its track, which generates a electric current on layers that can be read out, as represented in figure 31. The signal read on the readout layer has a triangular pulse shape and the signals are propagated to the front-end crates through copper cables.

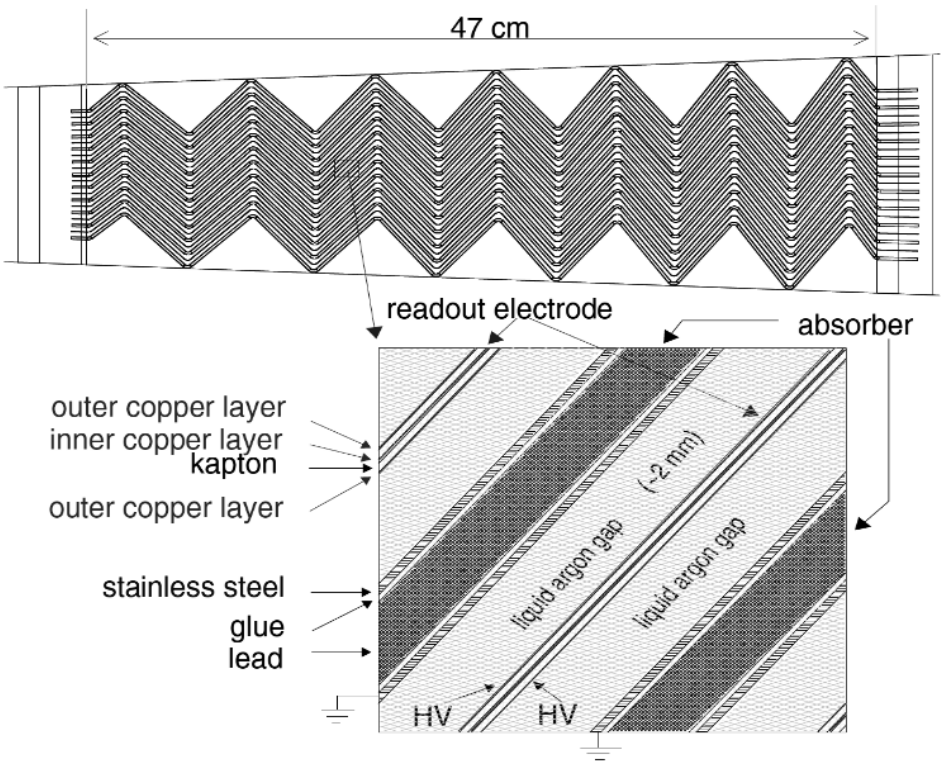


Figure 31 – Schematic of a cell of electromagnetic calorimeter.

The ATLAS readout and level 1 trigger systems receive data from the LAr calorimeter. The ATLAS readout path comprises the Front End Board (FEB) and the Readout Driver (ROD), while the trigger paths contain the LAr Trigger Digitizer Board (LTDB) and the LAr Digital Processing Blade (LDPB). The FEB amplifies, shapes, and digitizes the ionization signals for each readout cell, and the ROD computes the energy and time from the digitized signals. The LTDB boards shape the amplified signals and sent to the LDPB board trough optical links.

2.2.3.2 Tile Hadronic Calorimeter

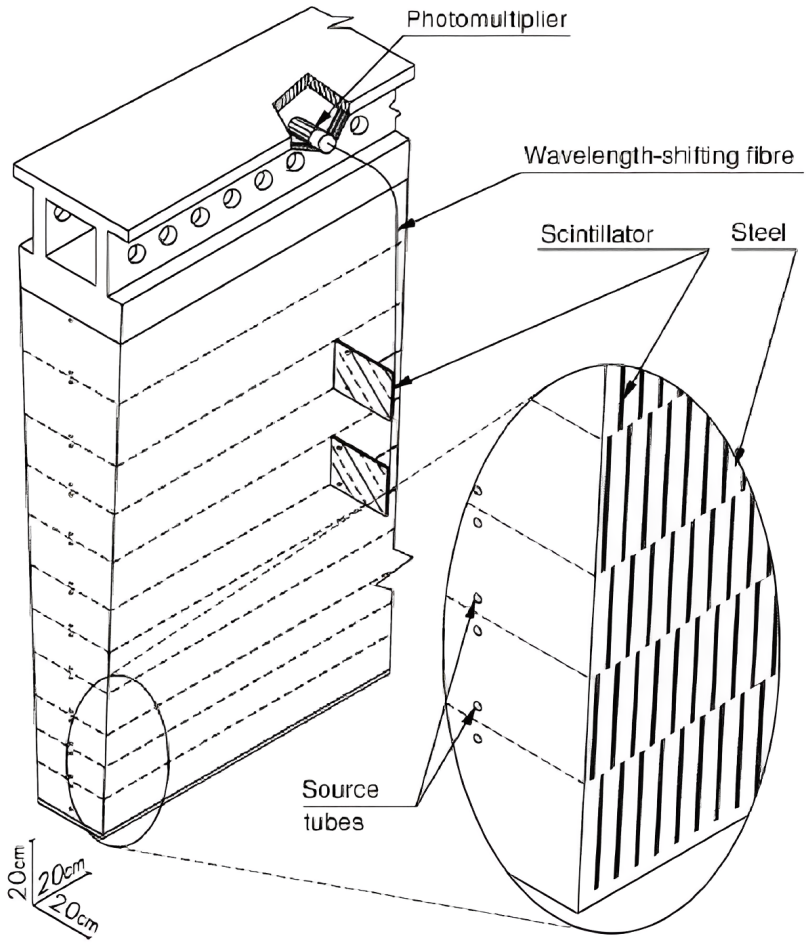


Figure 32 – Tile calorimeter module with its various optical read-out components, including the tiles, fibers, and photomultipliers. [29]

$$\frac{\sigma_E}{E} = \frac{50\%}{\sqrt{E}} \oplus \frac{1\%}{E} \oplus 3\% \tag{2.11}$$

The Hadronic Calorimeter (HCAL) is located outside of the ECAL and is responsible for measuring the energy deposited by strongly interacting hadrons that escape the ECAL. The HCAL consists of a barrel and two end-caps, each using different technologies. The Tile Calorimeter (TileCal), represented in figure 32, makes up the barrel HCAL, covering a range of $|\eta| < 1.7$ with three segments, the central one with a range from $|\eta| < 1.0$ and two outer segments over $0.8 < |\eta| < 1.7$. Steel is used as the absorber material, and scintillating tiles are the active material. Charged particles passing through the scintillating tiles produce ultraviolet light, which is collected by wavelength-shifting fibers before reaching photomultiplier tubes (PMTs) that output an amplified electrical signal. The End-Cap Calorimeters (HEC) are located behind the end-cap ECAL and cover the pseudorapidity range of $1.5 < |\eta| < 3.2$. The energy resolution of the HCAL represented by the equation 2.11 is less precise than the ECAL due to the fraction of initial energy diverted to low energy nuclear processes, which are not reconstructed.

Additionally, hadronic showers have electromagnetic components that interact differently with the HCAL than the hadronic components.

2.2.3.3 Forward Calorimeter

The Forward Calorimeter (FCAL) plays a vital role in measuring the energy of both electromagnetic and hadronic particles in the forward region, where the particle flux is particularly high. Covering a pseudorapidity range of $3.1 < |\eta| < 4.9$, it is essential in the reconstruction of the Missing Transverse Momentum, originating from particles that escape the detector, such as neutrinos and BSM particles. The FCAL is comprised of three layers, with each layer utilizing liquid-argon as the active material. The first layer uses copper as the absorber material, and it is specifically designed to measure electromagnetic showers. In contrast, the two other layers deploy tungsten absorbers, targeting the reconstruction of hadronic showers. The use of different absorber materials allows the FCAL to measure both types of particle showers accurately.

2.2.4 Muon Spectrometer

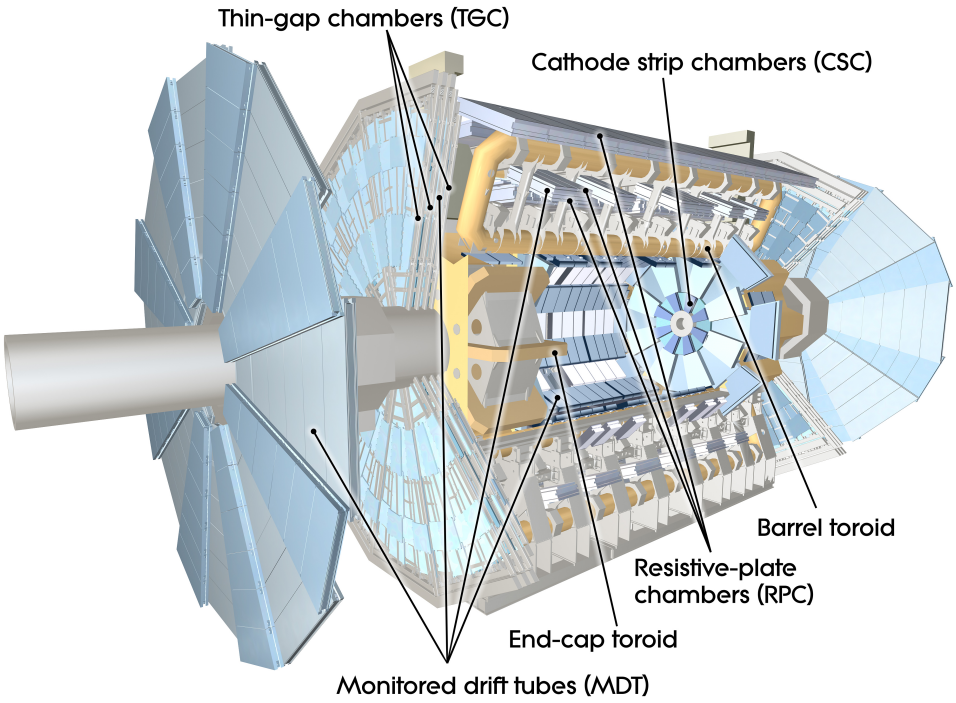


Figure 33 – The ATLAS Muons subsystem. [30]

The Muon Spectrometer (MS) is a crucial component of the ATLAS experiment, located at the outermost part of the detector. Muons are special particles due to their ability to travel through the calorimeters unstopped and the small amount of energy deposited ($\approx 3GeV$). The muons only can be identified and measured using a system of trigger and high-precision tracking chambers located outside the calorimeters. This system includes a powerful toroidal magnetic field, which is responsible for deflecting the muons as they exit the calorimetry system. The

MS is made up of several types of concentric cylindrical shell chambers, covering a range up to $|\eta| = 2.7$, and is divided into a long barrel and two inserted endcap magnets. The barrel toroid provides magnetic bending in the region $|\eta| < 1.4$, while two smaller endcap magnets are used for the region $1.6 < |\eta| < 2.7$. The MS is also equipped with trigger chambers, which cover a range up to $|\eta| = 2.4$. In addition, the forward muon-tracking region, known as the Small Wheel (SW), has been upgraded with two New Small Wheel (NSW) for Run 3, which serve both precision tracking and trigger functionality. The MS can be categorised into the precision tracking chambers (Monitored Drift Tube and Cathode Strip Chambers) and the fast trigger chambers (Resistive Plate Chambers and Thin Gap Chambers). The figure 22 shows the whole muon system with its components.

2.2.4.1 Monitored Drift Tube and Cathode Strip Chambers

The precision-tracking chambers in the large hadron collider are designed to accurately determine the path and momentum of muons. These chambers consist of two types of detectors, the Monitored Drift Tubes (MDT) and the Cathode Strip Chambers (CSC). The MDT chambers are placed in three layers between and on the coils of the superconduction barrel toroid magnet and cover the region $|\eta| < 2.7$. The MDT chambers are responsible for measuring the η position and have an average resolution of $80\mu m$ per tube and $35\mu m$ per chamber. On the other hand, the CSC chambers are located in the innermost layer covering the range $2.0 < |\eta| < 2.7$ and provide four independent measurements in both the η and ϕ directions. The CSC has a spatial resolution of $40\mu m$ in the η-plane and 5 mm in the ϕ-plane. With the precision-tracking chambers, a muon transverse momentum resolution of $\frac{\Delta p_T}{p_T} = 10\%$ at $p_T = 1TeV$ can be achieved, aided by the open structure of the air-core toroid magnet which reduces multiple-scattering effects.

2.2.4.2 Resistive Plate Chambers and Thin Gap Chambers

The LHC uses two trigger systems to detect muons. This system consists of special muon chambers that can quickly send signals after the passage of a particle, allowing for the identification of a beam crossing. These chambers measure both the bending and non-bending coordinates of the track, providing additional information about the muon tracks. The Resistive Plate Chambers (RPC) are installed in the barrel region and have a resolution of about 10 mm in both the bending and non-bending planes. On the other hand, Thin Gap Chambers (TGC) are installed in the endcap region, and they provide more precise information with a precision of 2 to 7 mm in the η coordinate and 3 to 7 mm in the ϕ coordinate. Together, the two trigger systems cover the range $|\eta| < 2.4$, with the RPC operating at $|\eta| < 1.05$ and TGC covering the remainder of the range. While both systems primarily provide BC identification and coarse muon tracking information, their combined data helps researchers study the characteristics of muons in greater detail.

2.2.5 Trigger and Data Acquisition

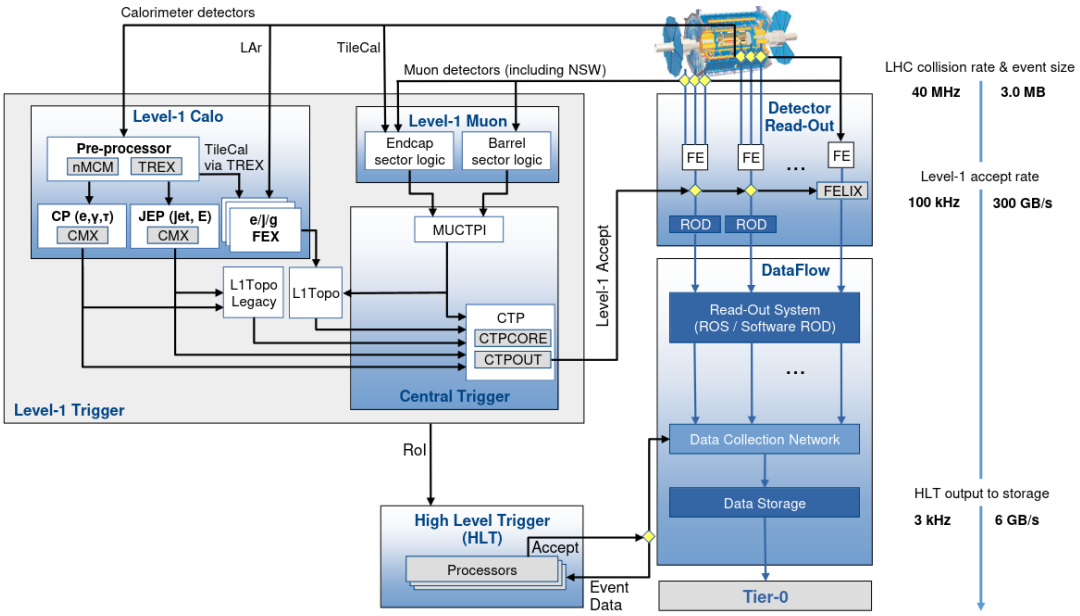


Figure 34 – The ATLAS TDAQ system in Run 3 with emphasis on the components relevant for triggering as well as the detector read-out and data flow. [31]

The Large Hadron Collider (LHC) produces around 40 million collisions per second, resulting in a massive challenge for data acquisition and storage. With each event size being approximately 1.5 MB, it is not feasible to store all data for later analysis. To address this problem, the ATLAS experiment uses a Trigger and Data AcQuisition system (TDAQ) to filter data in almost real-time and determine which events should be kept. The trigger system involves two levels: the Level 1 hardware trigger, which uses custom electronic hardware to decide whether to keep an event based on information from the calorimeter and muon spectrometer systems, and the High-Level Trigger (HLT), a software-based trigger that performs reconstruction algorithms on complete event readout data or isolated regions of the detector. The HLT reduces the data rate from 100 kHz to an expected average output rate of 3 kHz, resulting in a total data rate reduction by a factor of approximately 1.2 million. This allows the remaining events to be written for permanent storage, despite the increasing luminosity recorded by ATLAS each year. TDAQ system is represented by the figure 34 during run 3.

2.3 High Luminosity-Large Hadron Collider

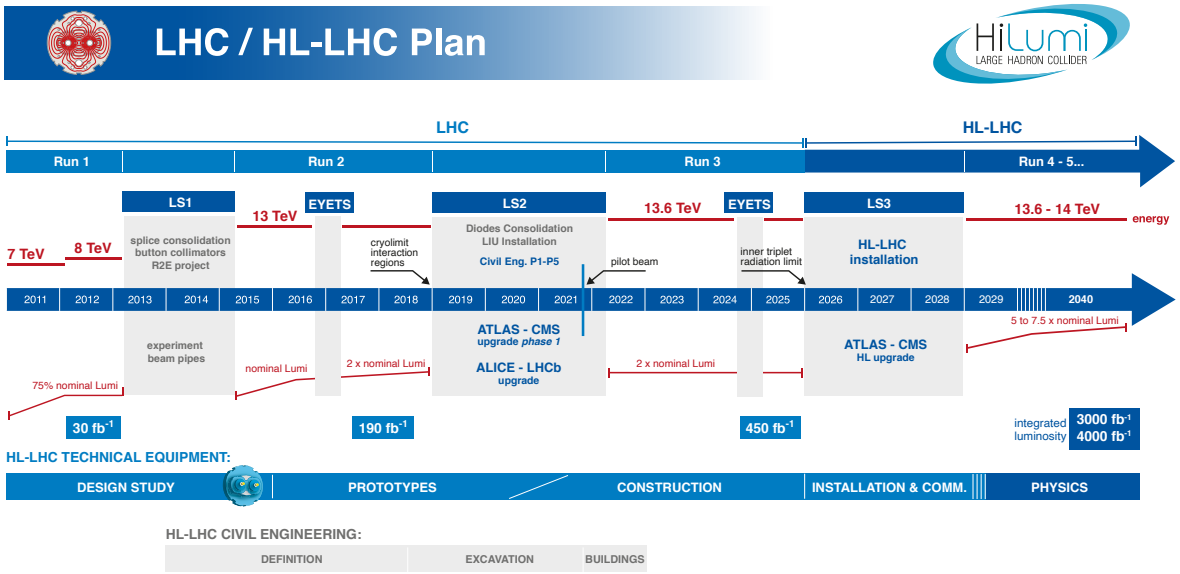


Figure 35 – The LHC to HL-LHC plan.

The Large Hadron Collider (LHC) has been a vital tool in particle physics research since its full commissioning for proton-proton collisions in 2010. Over the years, it has been upgraded to reach its nominal luminosity of $1 \times 10^{34} cm^{-2} s^{-1}$ in 2016. However, to fully exploit the machine's physics potential, CERN decided to upgrade it to the High Luminosity Large Hadron Collider (HL-LHC) from 2029 [32]. The primary goal of the HL-LHC is to increase the instantaneous luminosity, which is expected to reach up to $7 \times 10^{34} cm^{-2} s^{-1}$, resulting in an accumulated integrated luminosity of around $4 ab^{-1}$ by the end of HL-LHC data-taking. This upgrade will also lead to an increase of up to 200 proton-proton collisions per BC.

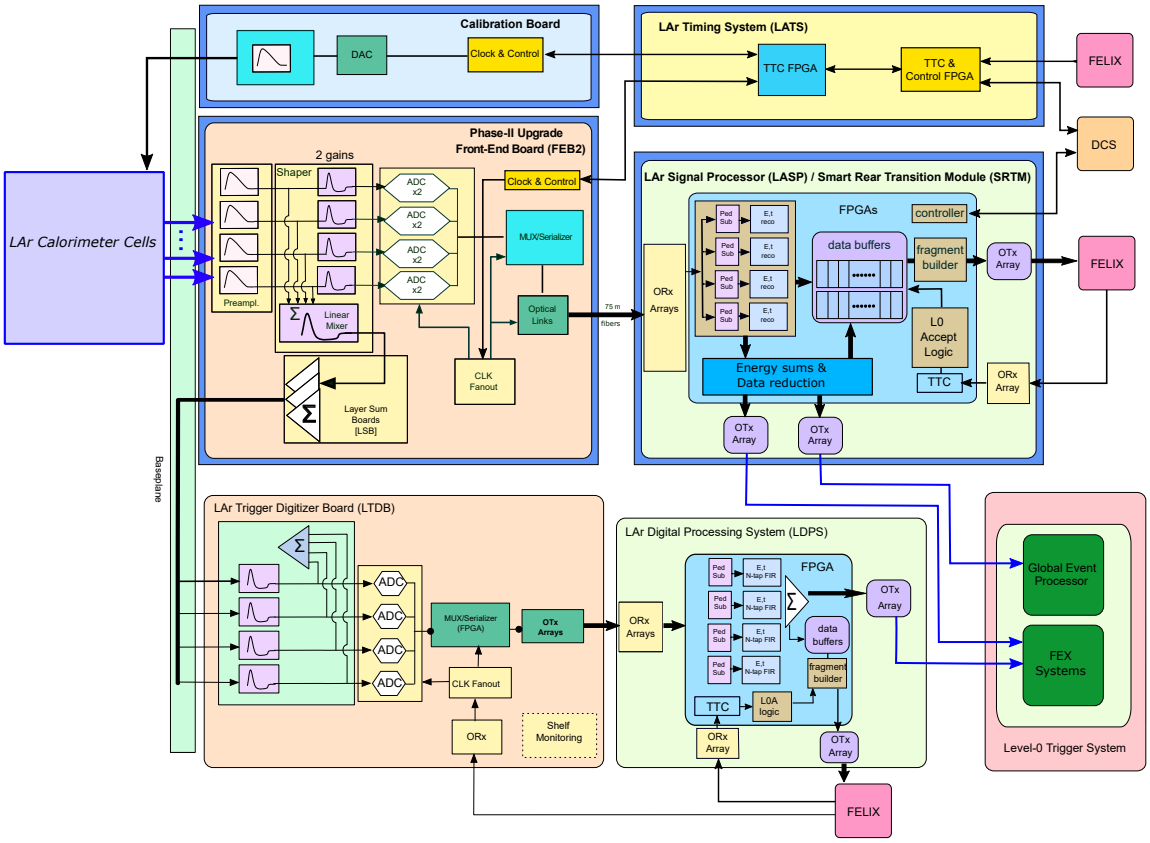


Figure 36 – Schematic block diagram of the LAr calorimeter readout architecture for the Phase-II upgrade. [33]

The HL-LHC upgrade includes the upgrade of the LAr calorimeters. While the detector itself will remain unchanged, the full electronics readout will be replaced and enhanced. The upgrade will occur in two phases, with phase 1 currently being commissioned and phase 2 scheduled for installation in 2026 (figure 35). The first phase focuses on upgrading the trigger readout path, while the second phase will improve the main readout. The second phase upgrade involves exchanging the front end boards with the Front End Board Two (FEB2) and the backend ROD boards with the new LAr Signal Processing (LASP) board, figure 36 shows the upgraded electronics. The FEB2 will digitize signals at 40 MHz, with an increased precision compared to the previous equipement, and the LASP will compute and reconstruct the energies from the digitized signals. Major upgrades are also planned for the ATLAS ID with the ATLAS Inner Tracker (ITk), which will improve the experimental precision of SM measurements and allow for the investigation of rare phenomena such as the Higgs boson self-coupling.

The large amount of data collected during the HL-LHC phase will greatly improve the precision of SM measurements and allow for the investigation of rare phenomena.

CHAPTER

3

IMPLEMENTATION OF RECURRENT NEURAL NETWORK IN HLS4ML TARGETING THE ENERGY RECONSTRUCTION IN THE LAR CALORIMETER

This chapter presents recurrent neural networks (RNNs) for energy reconstruction and their implementation in HLS4ML. In Section 3.1, we discuss the LHC upgrade and the problems faced with the update. The operation of the optimal filtering algorithm is discussed in Section 3.2, highlighting its importance in data processing and the need for its replacement. Next, Section 3.3 explores the concept of neural networks and their broad applications. The subsequent Section 3.4 focuses specifically on the use of neural networks for energy reconstruction, delving into the methodologies and techniques employed. Section 3.5 provides a detailed overview of Field-Programmable Gate Arrays (FPGAs) and elucidates their fundamental role in the context of neural networks implementation. In Section 3.6, the HLS4ML (High-Level Synthesis for Machine Learning) framework is explained, providing detailed information on its purpose and practical utilization for machine learning algorithm implementation. Section 3.7 analyzes the specific implementation of recurrent neural network (RNN) algorithms in the HLS4ML framework, discussing the intricacies and considerations involved in this process. Finally, Section 3.8 examines the optimization strategies employed and the results obtained using RNNs in energy reconstruction. This section offers a comprehensive analysis of the performance and effectiveness of RNNs, providing valuable insights into their practical applicability and possible enhancements.

3.1 Introduction

ATLAS is designed to measure the properties of particles produced during high-energy proton-proton collisions that take place every 25 nanoseconds, corresponding at a frequency of 40 MHz. Energy deposits in the LAr Calorimeter lead to triangular current pulses arising from the ionization of liquid argon by charged particles under high voltage potential between two absorber plates. After reaching the FEB, a bipolar shaping function is applied to the pulse, which is then sampled at 40 MHz as shown in figure 37. The current energy deposited computation in LAr uses optimal filtering algorithms [34], which assume a nominal pulse shape of the electronic signal. The additionally injected signals have a Gaussian-distributed time gap centered around 30 BC, with a standard deviation of 10 BC between two pulses. This results in both overlapping and non-overlapping high-energy pulses. However, the large out-of-time pileup, where up to 25 signal pulses from subsequent LHC BCs can overlap, as shown in figure 38, results in significant energy degradation, particularly for low time-gap between two consecutive pulses.

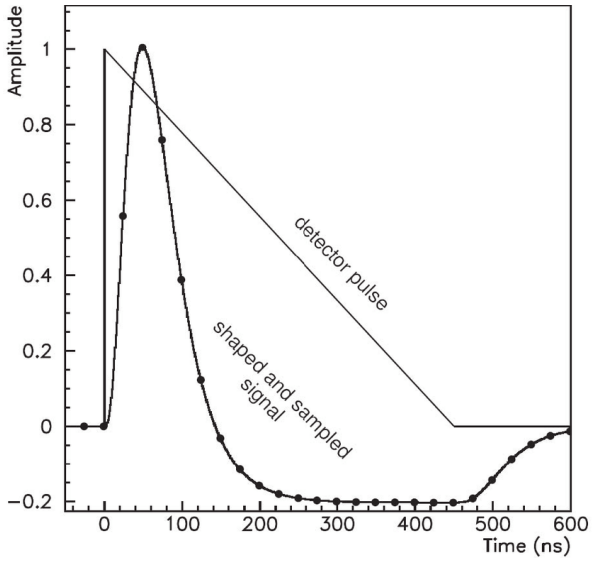


Figure 37 – Shapes of the LAr calorimeter current pulse in the detector and of the signal output after bipolar shaping. The dots represent the samples seperated by 25 ns. [1]

Therefore, a new energy reconstruction algorithm, based on artificial neural networks is being considered. This energy reconstruction algorithm uses deep learning techniques to identify and correct for pileup noise. These improvements in the energy reconstruction algorithms will enable ATLAS to meet the challenges of high-precision measurement at the HL-LHC, opening up new opportunities for discoveries in particle physics.

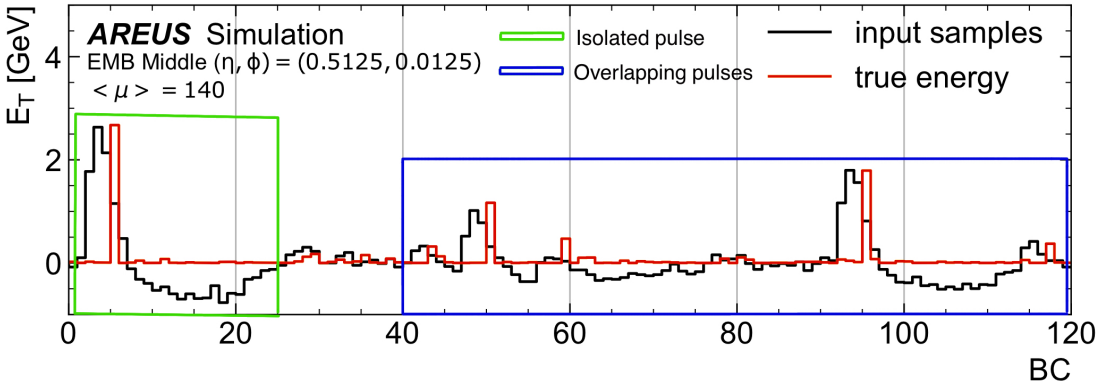


Figure 38 – Representation of a sample sequence (black) simulated by AREUS. The green box indicates the pulses of a cell under HL-LHC pileup conditions, whereas the blue box highlights the presence of overlapping pulses. The true transverse energy deposits is shifted by five BC to the right and is shown in red. [2]

3.1.1 Data Simulation

To replicate the FEB2 pulses, a simulation software called ATLAS Readout Electronics Upgrade Simulation (AREUS) [35] is utilized. AREUS was initially created to imitate the complete LAr trigger readout chain for Phase-I upgrades. It is an efficient and adaptable detector simulation framework, particularly designed to assess the performance of various digital signal filters. Within AREUS, signals produced by a Monte-Carlo event generator are processed in a specific time frame on a BC-by-BC basis. The events are then processed, as they occur in the continuous data flow, and the filter algorithms are applied accordingly. The AREUS software has the ability to consider the granularity of the LAr calorimeter cells. It is also capable of producing pulse signals for individual cells. The resulting pulse shapes generated by AREUS closely match those produced by the electronic readout circuitry, even under specified LHC pileup conditions. Consequently, AREUS can be utilized for a complete simulation of the FEB2 expected pulse signals without the need for physical hardware implementation.

In HL-LHC operation, the expected energy spectrum contains low-energy deposits of up to about 1 GeV generated from particles in inelastic proton-proton collisions. To simulate hard-scattering events [35], a random uniform transverse energy spectrum is injected, with a maximum energy deposit of 5 GeV. Additionally, the software enables the simulation of LHC bunch patterns, which refer to the regular interruptions in the proton-proton collision series. For this study, a sample is obtained with an average of 140 pileup events ($\langle\mu = 140\rangle$) with additional hard scatter signals produced at a mean time interval of 30 BC and a standard deviation of 10 BC, leading to the creation of both overlapping and non-overlapping high-energy pulses.

3.2 The Optimal Filter Technique

The signal utilized for the energy reconstruction algorithm in the LAr calorimeter is obtained through the ionization current adeptly a triangular waveform, where the peak corresponds to the energy deposited by the shower. The ionization signal undergoes pre-amplification and is subsequently shaped by a CR-RC2 positioned at the end of the readout chain. Sampling of the bipolar signal takes place at a frequency of 25 *ns*, which aligns with the LHC bunch crossing period. These samples are then digitized and employed in the signal reconstruction procedure.

The current readout electronics of the LAr calorimeters applies an OF to determine the energy in each cell.The OF technique operates by applying a filter to the Analog-to-Digital Converter (ADC) samples time series s(i), generating an energy time series E(t) through equation 3.1.

$$E(t) = \sum_{i=t}^{t+n} a_i \cdot s_i, \tag{3.1}$$

Where n is the pre-set number of samples, and a(i) are the optimized filter coefficients calculated using an algorithm that takes into account the noise characteristics of the readout system.The OF technique uses the known noise auto-correlation function and the depth filter, which is a filtering method that takes into account the depth or time information of the data, to determine the optimal filter coefficients. These coefficients are then applied to the ADC samples for each cell, producing the energy time series.

To ensure accurate energy reconstruction, the choice of N need to be considered. It affects the precision of the energy estimates. The number of samples is a critical component that balances the accuracy of the energy estimates with the computational resources required for the reconstruction. Currently, ATLAS is operating with 4 samples at trigger level and 3 at main readout around the bipolar pulse shape as represented on the Figure 37. By linear combination of digitized pulse samples, electronic noise and signal pileup are suppressed. To further identify true energy deposits and assign them to a certain BC, a peak finder is applied to the output sequence of the OF by selecting the maximum value in each group of three consecutive BCs, as shown in figure 39[36].

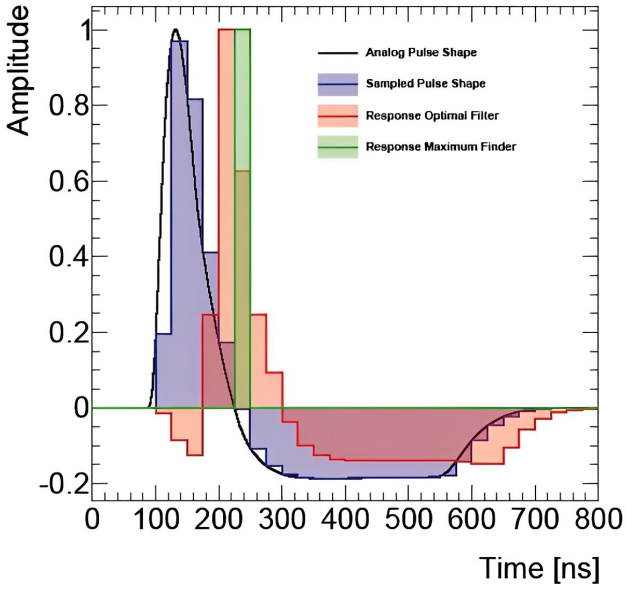


Figure 39 – The shape bipolar shaping of the LAr calorimeter's pulse in the detector is depicted in black, while the signal following digitizing is displayed in purple. The red line shows the reconstruction done by the Optimal Filtering (OF) on the sampled pulse shape shifted by 5, and the green line displays the OF with the maximun value shifted by 6.

The OF resolution degrade at low time gap because of distorted pulse shapes due to overlapping pulses as shown in figure 40. This degradation is important at HL-LHC due to high pileup (up to 200 simultaneous p-p collisions per bunch crossing) leading to overlapping pulses; the degradation at single cell level is shown in figure 41 and for the diphoton mass resolution in figure 42.

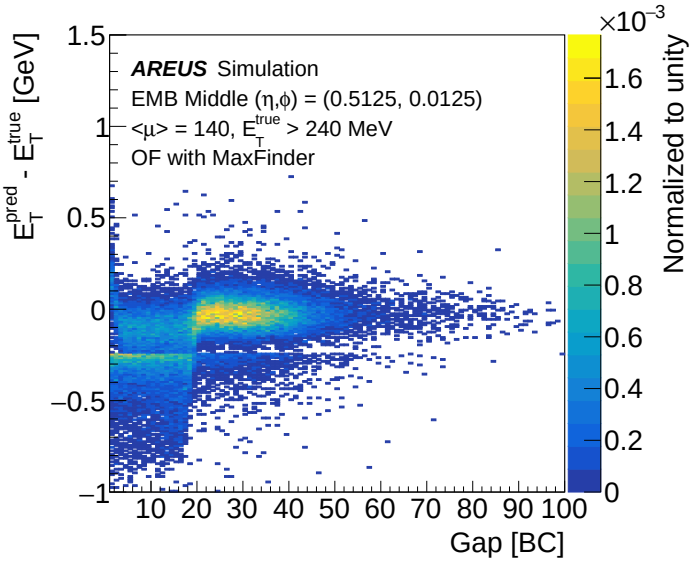
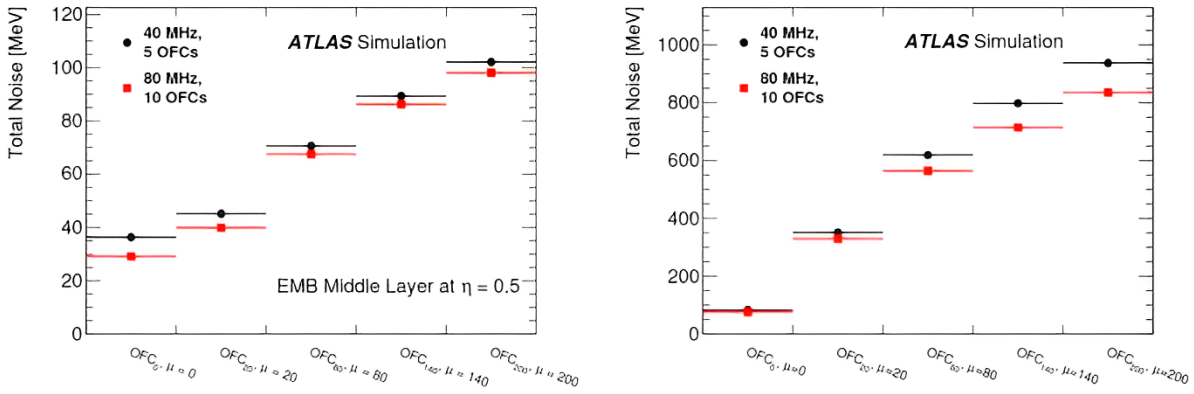


Figure 40 – The resolution of the OF with maximum finder as a function of the gap between two energy deposits for higher energies than 240 MeV. [3]



(a) (b)

Figure 41 – The amount of noise, given optimal filtering with either 5 samples at a 40 MHz sampling rate or 10 samples at an 80 MHz sampling rate, is determined by the level of pileup for a cell in the EM middle layer at $\eta = 0.5$ (a) and a HEC cell in the first layer at $\eta = 2.35$ (b). [33]

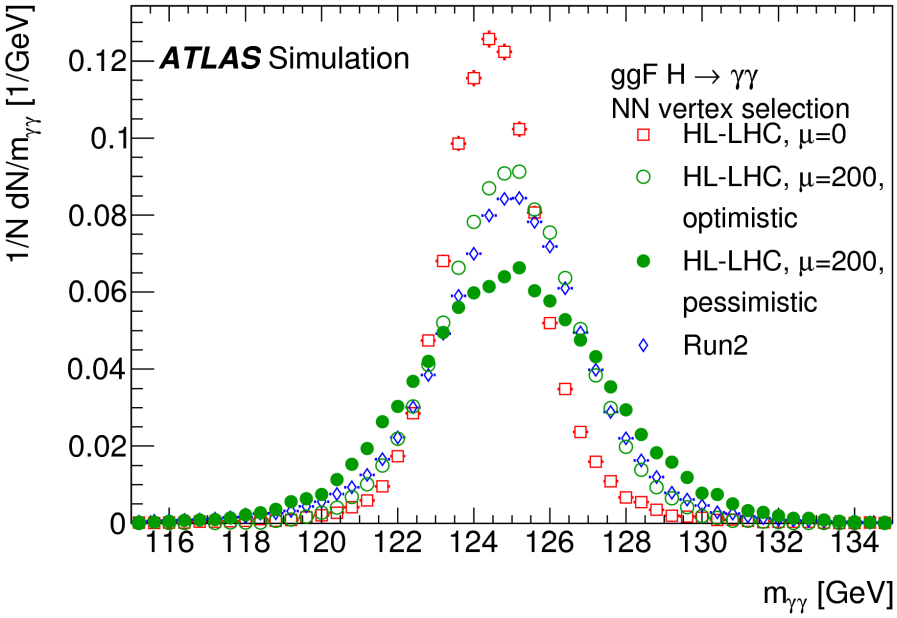


Figure 42 – Diphoton invariant mass obtained using data in Run 2, $< \mu >= 0$ simulation and $< \mu >= 200$ simulation at HL-LHC using the optimistic (reduction of pileup noise and more unconverted photons) and pessimistic (no improvement) photon resolution scenarios. High pileup decrease significantly the $\gamma\gamma$ resolution. [33]

3.3 Neural Networks

3.3.1 Artificial Neural Networks History

Artificial Neural Networks (ANNs) are a type of machine learning algorithm that has become increasingly popular in recent years. ANNs are modeled following the structure and function of the human brain, consisting of interconnected nodes, or neurons, that process and transmit information. The history of ANNs can be traced back to the work of Warren McCulloch

and Walter Pitts [37] in the 1940s, who proposed a model of artificial neurons that could be used to simulate the functions of the human brain. However, it was not until the 1980s and 1990s that ANNs became more widely used in artificial intelligence (AI) research, thanks to advances in computing power and the development of new algorithms.

One of the major breakthroughs in ANN research was the development of backpropagation, a learning algorithm that enables ANNs to learn from data and adjust their weights and biases to improve their performance. The backpropagation algorithm was first proposed by Paul Werbos in 1975 [38], but it was not until the 1980s that it became widely used in ANN research. During the 1980s and 1990s, ANNs were used in a wide range of applications, including speech recognition, image processing, and natural language processing. One of the most famous examples of ANNs in action was the development of a system for recognizing handwritten digits, known as the MNIST dataset. This system, which was developed by Yann LeCun and his colleagues in the 1990s [39], achieved a level of accuracy that surpassed human performance on this task.

In recent years, ANNs have become even more powerful, thanks to the development of deep learning, a type of machine learning algorithm that uses multiple layers of neurons to extract complex features from data. Deep learning has been used in a wide range of applications, including computer vision, natural language processing, and robotics.

Despite their impressive performance, ANNs still face several challenges and limitations. One of the main challenges is the problem of overfitting, which occurs when an ANN is trained too well on a particular dataset and is unable to generalize to new data. To address this problem, researchers have developed techniques such as dropout and early stopping [40], which can help to prevent overfitting. Another challenge is the interpretability of ANNs, which can make it difficult to understand how they arrive at their decisions. This is particularly important in applications such as healthcare, where the decisions made by ANNs can have significant consequences for patients. To address these challenges, researchers are developing new techniques and algorithms for training and interpreting ANNs, as well as exploring new architectures, such as spiking neural networks [41], that more closely mimic the structure and function of the human brain.

Overall, the history of ANNs is a story of constant innovation and progress, driven by advances in computing power, algorithms, and our understanding of the human brain. As ANNs continue to evolve and improve, they hold great promise for solving some of the most challenging problems in AI and improving the quality of life for people around the world.

3.3.2 Fundamentals of Neural Networks

Neural networks (NNs) consist of multiple processing layers of interconnected nodes, also known as neurons, that are designed to learn from data and perform a specific task. The most essential aspects of artificial neural networks include robustness, fault tolerance, handling

unclear information, and parallelism.

One of the key considerations when working with NNs is designing an appropriate architecture and selecting an appropriate training approach. The architecture of a neural network refers to the number and type of layers, the number of neurons in each layer, and how they are interconnected. The learning method used to train the neural network involves adjusting the network's weights, which determine the strength and direction of the connections between neurons, to minimize the difference between the network's predictions and the targeted values.

Neurons are the basic processing units of a neural network. They receive input signals from other neurons or external sources, perform a computation on the inputs, and produce an output signal that is passed on to other neurons. A typical neuron processing involves the linear combination of inputs and weights, followed by the application of an activation function to the linear combination. The activation function introduces non-linearity into the NN and enables it to model complex relationships between inputs and outputs. Common activation functions include sigmoid, hyperbolic tangent (tanh), Softsign, and rectified linear unit (ReLU) (Figure 53 and 56). The type of NN and learning method used is often dictated by the nature of the problem being solved. For example, feedforward networks are commonly used for classification tasks, while recurrent networks are suitable for sequential data processing tasks.

NNs have revolutionized the field of AI and have become essential tools for solving complex problems in a wide range of applications. They offer a powerful means for modeling complex relationships in data and can provide insights and predictions that are difficult to obtain by other means. Understanding the fundamentals of neural network architecture, learning methods, and activation functions is crucial for designing and training efficient and effective neural networks.

3.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) have been an important focus of research and development since the 1990s. They are designed to learn sequential or time-varying patterns. A recurrent net is a neural network with feedback (closed loop) connections [42]. As shown in the Figure 43, there are four distinct manners in which RNNs can be employed to map inputs onto outputs:

- Many-to-Many: In this type of mapping, the network receives a sequence of inputs and produces a sequence of outputs with the same ($x^{T_x} = \hat{y}^{T_y}$) or different length ($x^{T_x} \neq \hat{y}^{T_y}$). This is useful when we want to model relationships between input sequences and corresponding output sequences. Examples of this include text translation, where a sequence of words in one language is mapped to a sequence of words in another language, and video captioning, where a sequence of video frames is mapped to a sequence of captions.

- One-to-Many: The network receives a single input and produces a sequence of output ($x^{T_x} = 1, \hat{y}^{T_y} > 1$). This is useful when one needs to generate a sequence from a single input, such as in music generation or image description.

- Many-to-One: The network receives a sequence of input and produces a single output ($x^{T_x} > 1, \hat{y}^{T_y} = 1$)). This is useful when one needs to model the relationship between a sequence of input and a single output, such as in sentiment classification of text or prediction of stock prices based on a sequence of historical data.

- One-to-One: The network receives a single input and produces a single output ($x^{T_x} = \hat{y}^{T_y} = 1$)). This is useful when one needs to model a simple relationship between an input and an output, such as in image classification.

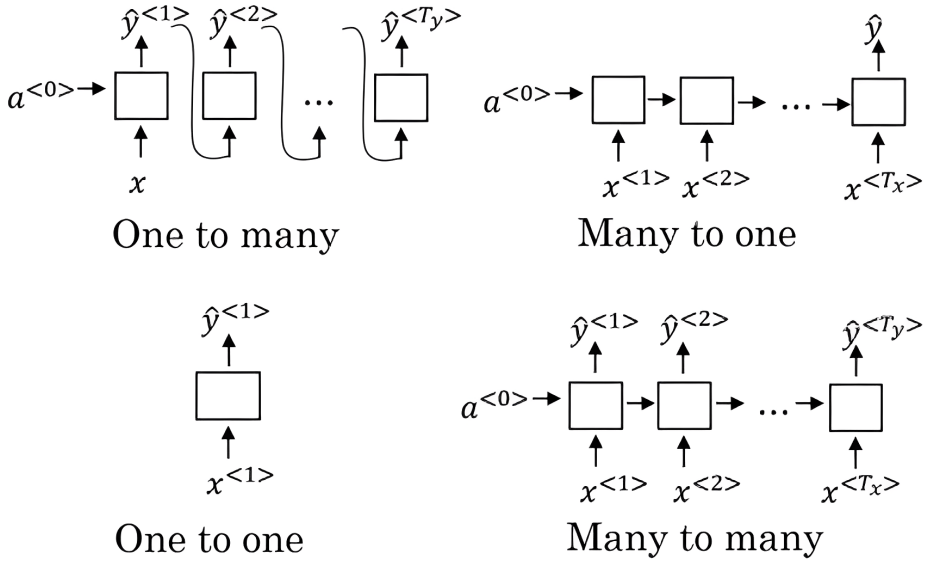


Figure 43 – The diagram illustrates different possible architectures for RNNs, with the input layer cells denoted as 'x', the hidden layer cells as blocks, the cell state as 'a', and the connections between blocks as hidden state. The output layer cells are represented by '$\hat{y}$'. The one to one architecture maps a single input to a single output, while the one to many architecture maps a single input to multiple outputs. The many to many architecture maps multiple inputs to multiple outputs, and the many to one architecture maps multiple inputs to a single output.

Another important point for RNN arises during the training process is to calculate the partial derivatives of the error (loss function [43]) with respect to the weights. This is done by going backward through the network. Once the partial derivative is obtained, it is multiplied by the learning rate to calculate the step size. Then, the step size is added to the original weights to calculate new weights. The gradient measures how much the output of a function changes when the inputs are changed slightly. It can be thought of as the slope of a function. The steeper the slope (higher the gradient), the faster a model can learn. Conversely, if the slope is almost zero, the model stops learning. The gradient simply measures the change in all weights with

respect to the change in error. When training an RNN algorithm, the gradient can sometimes become too small or too large, which can cause issues such as poor performance, low accuracy, or a long training period. The exploding gradient is characterized by the exponential growth of the gradient value as it is propagated through the layers of the network.This uncontrolled growth of the gradient can lead to the gradient becoming too large and the slope growing exponentially. This issue can be solved using methods such as Identity Initialization [44], Truncated Back-propagation, or Gradient Clipping [45]. However, the vanishing gradient issue occurs when the values of the gradient are too small, causing the model to stop learning or take too long to train. To solve this issue, methods such as Weight Initialization [46] or choosing the right Activation Function [47] can be used.

There are several variations of RNNs, such as Long Short-Term Memory (LSTM) and Vanilla Recurrent Neural Network (Vanilla-RNN), that use different types of recurrent neurons to deal with problems such as vanishing or exploding gradients during training. However, all variations share the main characteristic of RNN, which is the ability to maintain an internal state that allows the network to keep information about previous input data while processing the current input.

3.3.3.1 Vanilla Recurrent Neural Network

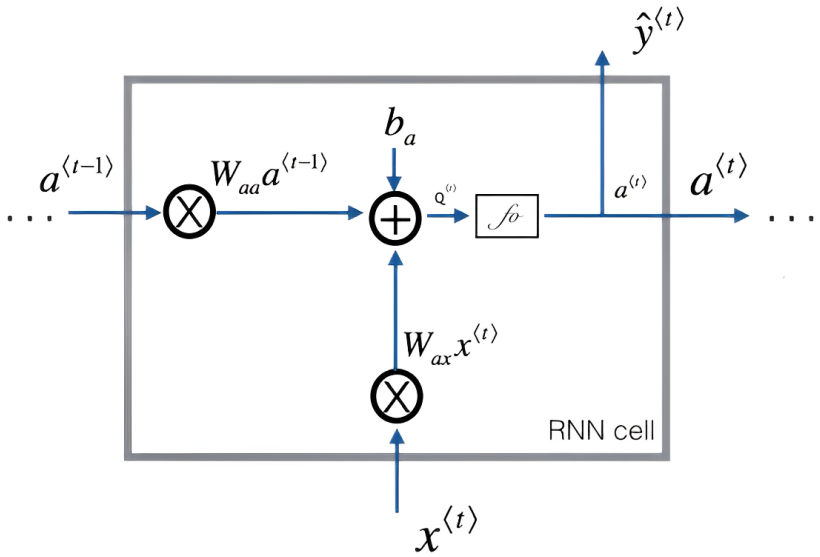


Figure 44 – Schematic of a Vanilla-RNN cell, a fundamental processing unit in recurrent neural networks. The cell takes two inputs: $x^{<t>}$, representing the current input, and $a^{<t-1>}$, representing the previous hidden state which contains information from past inputs. The cell computes a new hidden state, $a^{<t>}$, which is passed to the next RNN cell and used to predict the output, $\hat{y}^{<t>}$.

The Vanilla-RNN is a neural network that uses feedback to handle sequences of data and their structure. It is represented in Figure 44 and described by the equation 3.2 and 3.3.

$$q^{<t>} = W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a \tag{3.2}$$

$$\hat{y}^{<t>} = a^{<t>} = f_o(q^{<t>}) \tag{3.3}$$

Where,

- $f_o(q^{<t>})$ is a mathematical function that is applied before the output of a layer to introduce nonlinearity into the network and allow it to model more complex data. The most common activation functions in RNNs include the hyperbolic tangent function, sigmoid function, and ReLU function.
- b is a bias term added to the weighted sum of inputs in a neuron. It allows the activation function to shift to the left or right, enabling the network to better fit the data.
- W_{aa} and W_{ax} are matrices that define the weights of the connections between the hidden layer units and the input units, as well as the connections between the hidden units at different time steps.

The structure of the Vanilla-RNN consists of an input layer, a hidden layer, and an output layer. The input layer is responsible for receiving the input data of the sequence, which is encoded as input vectors. Each element in the sequence is fed into the network at a time interval, with a corresponding input vector. Mathematically, we have a $[X_{t-\infty}, \ldots, X_{t-1}, X_t, X_{t+1}, \ldots, X_{t+\infty}]$ series where X_t is the input vector at time t. For our use case, the dimension of X_t is one. The hidden state is calculated from the current input vector, bias, the previous hidden state, and a non-linear activation function. The input value X_t is multiplied by the weight W_{ax} defined by the network constructor. This weight adjusts the importance of different computational outcomes obtained by the neuron. The previous hidden layer, as shown in equation 3.2, uses the matrix W_{aa} to transform the hidden state $a^{<t-1>}$ before the addition and activation function. It has dimensions m, which represents the number of units in the output layer. This enables the network to capture information about the sequence and the current context in which the input vector is embedded. The variable $q^{<t>}$ is obtained after the addition and before the activation function. The output of the hidden layer, represented by $a^{<t>}$ in figure 44, is obtained after applying the non-linear transformation. It is responsible for storing the state of the network, which is updated at each time interval as new data is received. The $\hat{y}^{<t>}$, shown in equation 3.3, illustrates the output of the network.

The training process of the Vanilla-RNN is similar to that of other neural networks. The loss function is defined based on the performance of the network on a training dataset. Then, the error backpropagation algorithm is used to update the weights of the network and minimize the loss function.

The main limitation of the Vanilla-RNN is the gradient problem that explodes or vanishes, which makes it difficult to train deep networks with long time intervals. To solve this problem,

variations of the RNN have been developed, such as the LSTM, which have flow control mechanisms that allow the network to capture relevant information over long periods of time.

3.3.3.2 Long Short-Term Memory

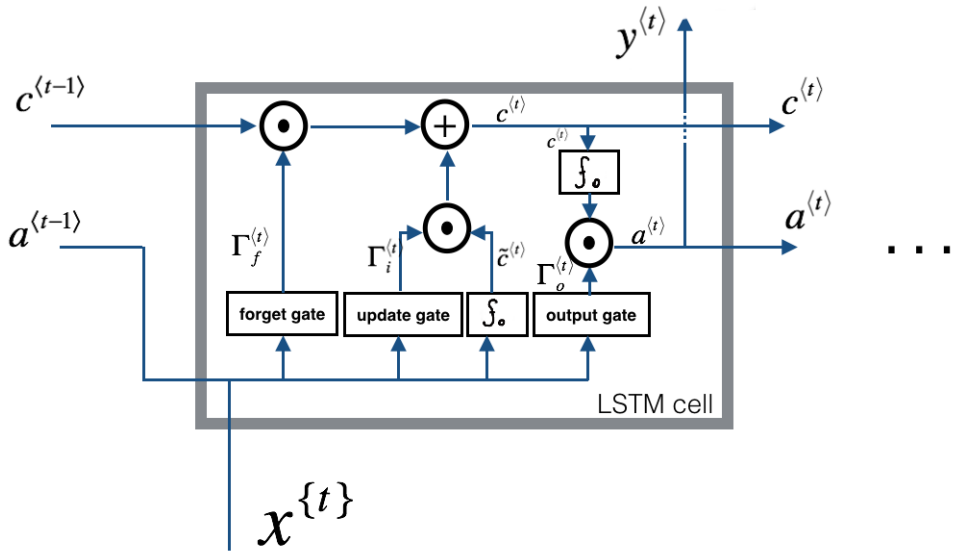


Figure 45 – Schematic of a LSTM cell. At each time-step, the cell tracks and updates a memory variable (the cell state) $c^{<t>}$, which is different from the hidden state, $a^{<t>}$.

The LSTM is a type of RNN that has gained popularity due to its ability to address the gradient problem that can arise in Vanilla-RNN. The LSTM was proposed by Hochreiter and Schmidhuber in 1997 [48]. This architecture is composed of an activation layer, which is applied to the cell's output before it is passed to the next cell in the sequence, and memory cells that have three gates controlling the flow of information:

- The update gate controls how much of the new information should be added to the cell's memory ($c^{<t>}$). This gate combines two main sets. The first set is calculated from the combination of the cell's previous state with the new input data, as shown in equation 3.4. The weights, W_u, determine how the update gate operates. Concatenate $[a^{<t-1>}, x^{<t>}]$ and multiplying them by W_u. The result passes through a σ function, which produces a value between 0 and 1. The equation results a vector, $\Gamma_i^{<t>}$, with values ranging from 0 to 1. A value of 1 in $\Gamma_i^{<t>}$ means the that the element should be completely updated, while a value close to 0 means that it should be ignored.

$$\Gamma_i^{<t>} = \sigma(W_i[a^{<t-1>}, x^{<t>}] + b_i) \tag{3.4}$$

  The second part consists of creating a new vector that can be added to the previous cell state by incorporating the state of the previous cell, as shown in equation 3.5.

$$\tilde{c}^{<t>} = f_o(W_c[a^{<t-1>}, x^t] + b_c) \tag{3.5}$$

Where f_o is the mathematical function, also known as the unit's activation. W_c is a weight matrix corresponding to the candidate cell, $[a^{<t-1>}, x^t]$ is the concatenation of the previous states and the current input, and b_c is the bias associated with the candidate cell. Then, the responses from both parties are combined to obtain the new cell state, which is calculated using the equation 3.6.

$$c^{<t>} = \Gamma_f^{<t>} \cdot c^{<t-1>} + \Gamma_i^{<t>} \cdot \tilde{c}^{<t>} \tag{3.6}$$

This equation is responsible for updating the cell state ($c^{<t>}$), incorporating two essential parts. First, it combines the previous cell state with the element-wise multiplication of the forget gate ($\Gamma_f^{<t>}$) and the previous cell state, allowing control over the amount of relevant information to be preserved. Second, the $\Gamma_i^{<t>}$ is then multiplied element-wise with $\tilde{c}^{<t>}$, ensuring the incorporation of new relevant information into the current cell state.

- The forget gate controls how much of the old information should be retained in the cell's memory ($c^{<t>}$). It is calculated in the same way as the update gate shown in equation 3.7, but the $\Gamma_f^{<t>}$ will be multiplied element-wise with $c^{<t-1>}$ information instead of the new input. A value close to 1 means that the element should be fully retained, while a value close to 0 means that it should be completely forgotten.

$$\Gamma_f^{<t>} = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{3.7}$$

- The output gate controls the computation of the output $a^{<t>}$. It is calculated from the previous state of the cell ($a^{<t-1>}$) and the current input ($X^{<t>}$), as shown in equation 3.8. The result passes through a σ function, also called as recurrent activation, which produces a value between 0 and 1 for each element of the memory. A value close to 1 means that the element should be fully used in the output, while a value close to 0 means that it should be ignored.

$$\Gamma_o^{<t>} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{3.8}$$

After calculating the output gate ($\Gamma_o^{<t>}$), it is multiplied by the result of the output forget function ($f_o(c^{<t>})$. The forget function is responsible for regulating which elements of the memory will be kept and updated to the next RNN cell and used to predict the output, $\hat{y}^{<t>}$.

$$\hat{y}^{<t>} = a^{<t>} = \Gamma_o^{<t>} \cdot f_o(c^{<t>}) \tag{3.9}$$

3.4 Neural Network for Energy Reconstruction

Models using neural networks can improve the accuracy of reconstructing the deposited energy in HL-LHC conditions. These computational models use artificial neural networks to learn the relationship between the readings from the LAr calorimeter sensors and the energy deposited by particles. RNNs are designed to work with sequential data and can take into account the temporal evolution of data. In the case of LAr data, there is a sequence of samples over time that can be input to the RNN. The previous samples in the past can be used to investigate pileup.

Both Vanilla-RNN and LSTM can be used to sequentially process sensor readings. The Vanilla-RNN cell architecture is the most compact, comprising a single internal neural network trained to convey relevant information forward in time and infer energy at a specific BC. Due to limitations imposed by the LAr calorimeter system, the size of the internal network is minimized as much as possible.

In LAr several NN models were developed and all were implemented using the Keras [49] library with the TensorFlow [50] backend. Keras is a high-level library that offers a simple and intuitive interface for creating artificial neural networks. It was developed to be user-friendly and built on top of other machine learning libraries such as TensorFlow. On the other hand, TensorFlow is an open-source software library for numerical computation using data flow graphs. Developed by the Google Brain Team in 2015, TensorFlow is particularly suitable for training and running machine learning models, and is widely used in research and the development of AI applications.

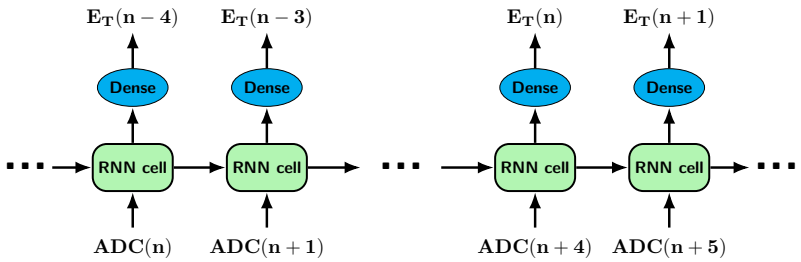


Figure 46 – Single-cell application of LSTM based recurrent networks. The LSTM cell and its dense decoder are computed at every BC. They analyse the present signal amplitude and output of the past cell, accumulating long range information through a recurrent application. By design, the network predicts the deposited transverse energy with a delay of six BCs. [3]

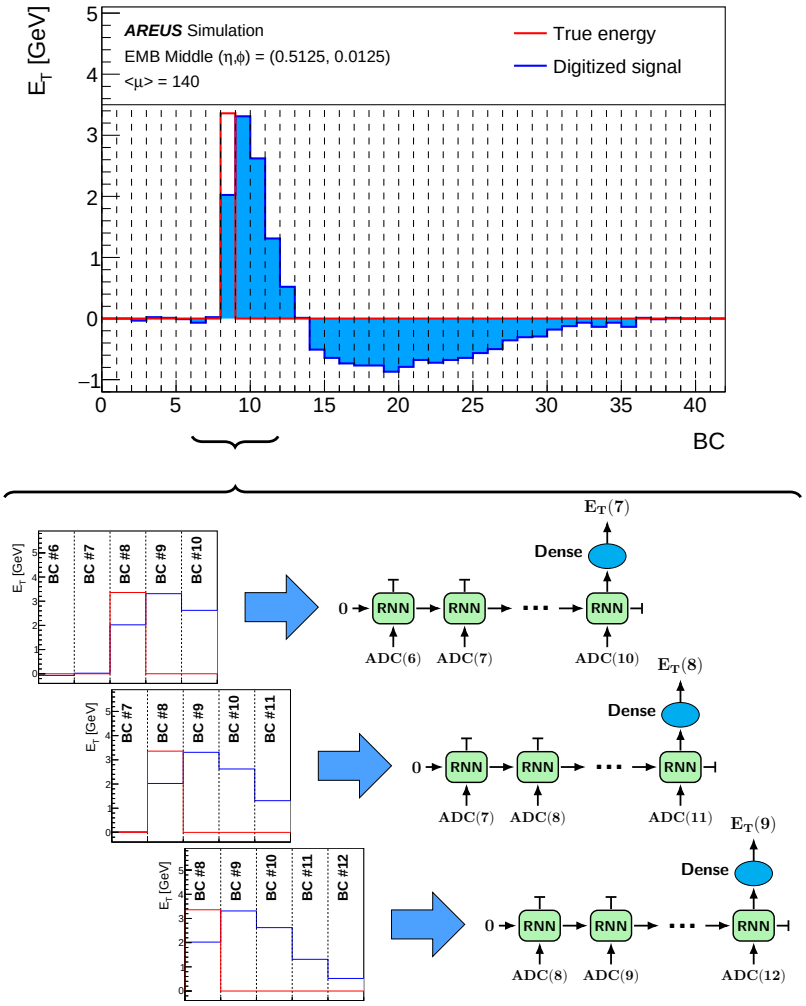


Figure 47 – Sliding window application of LSTM based recurrent networks. At each instant, the signal amplitude of the four past and present BCs are input into an LSTM layer. The last cell output is concatenated with a dense operation consisting of a single neuron and providing the transverse energy prediction. [3]

Two different techniques were used to achieve the best balance between resources and performance for measuring energy in real-time. The first techniques is based on a many-to-many RNN evaluation, and is depicted in figure 46. An RNN cell analyzes the amplitude of the LAr signal and the output of the previous cell to predict the energy at each BC. This operation is repeated until the end of the data. To ensure that the RNN accumulates enough information, a delay of five BCs is imposed during the training process. This also prevents the RNN from learning about collisions that have not yet occurred during training. The second techniques utilizes a sliding-window algorithm, as shown in figure 47. A network is instantiated at each BC and is trained as a many-to-one RNN, using five ADC samples as input, to predict the energy. The target energy corresponds to potential pulses that begin on the second BC, allowing the network to read one BC before the deposit and four on the pulse. This approach strikes the best balance between correcting for past events, inferring energy on the pulse, and accommodating

short sequences that meet FPGA constraints [3]. Due to their limited internal capabilities, Vanilla-RNN networks are unable to effectively handle information over long periods of time, which is why only a sliding window application is considered. However, it is possible to apply LSTM to both designs.

Different models were tested to obtain a good balance beetween performance and resource usage. Among the models tested, three structures were considered :

- Two LSTM models were considered as shown in Table 3. One model utilizes the single-cell design, while the other employs the sliding-window algorithm. Both models have consist of three layers: an input layer, an LSTM hidden layer, and a dense output layer. The LSTM layer has 10 units that determine the dimensionality of the hidden layer [3]. Fewer internal dimensions significantly degrade the energy resolution [51]. The unit activation function is the hyperbolic tangent, while the recurrent layer activation function is the sigmoid. The dense output layer has a single neuron and the ReLU activation function. Positioned subsequent to the LSTM layer, it concatenate the output in a single energy measurement. The model has no parameter regularization or restriction and does not use dropout or recurrent dropout techniques to prevent overfitting [43].
- Vanilla-RNN cell is a single neural network trained for temporal information forwarding and energy inference at a given BC. It uses an 8-dimensional internal representation and employs ReLU activation to avoid FPGA look-up tables. Due to limited internal capabilities, Vanilla-RNN networks are suitable for sliding window applications but struggle with long-term information management.

	Single-cell	***Sliding-window***	
	LSTM	*LSTM*	*Vanilla-RNN*
Time inference Receptive Field	∞	5	5
Samples after deposit	5	4	4
RNN layer Dimension	10	10	8
Activation	tanh	tanh	ReLU
Recurrent Activation	Sigmoid	Sigmoid	N/A
Dense layer Dimension	1	1	1
Activation	ReLU	ReLU	ReLU

Table 3 – Configurable key parameters of the single-cell and sliding-window algorithms. [3]

3.4.1 Neural Networks results

The Vanilla-RNN and LSTM networks outperformed the OF, both in terms of bias in the mean and resolution as shown in figure 48. The range containing 98% of the entries was also displayed, demonstrating non-Gaussian behavior in the far tails of the resolution, especially

at low energies. The OF tended to not reconstruct correctly low deposited energies, while the NNs effectively better reconstruct these energies. The LSTM (single-cell) has a high number of multiplier-accumulator (MAC) units, making it not possible to implement on an FPGA. This is because FPGAs can only accommodate a limited number of MAC units per network, usually a few hundred. The Vanilla-RNN demonstrated similar performance with the both LSTM, despite having less parameters, as shown in Table 4. Overall, the LSTM networks performed little better than the Vanilla-RNN. However, the LSTM (single-cell) implementations required five times more parameters than the Vanilla-RNN.

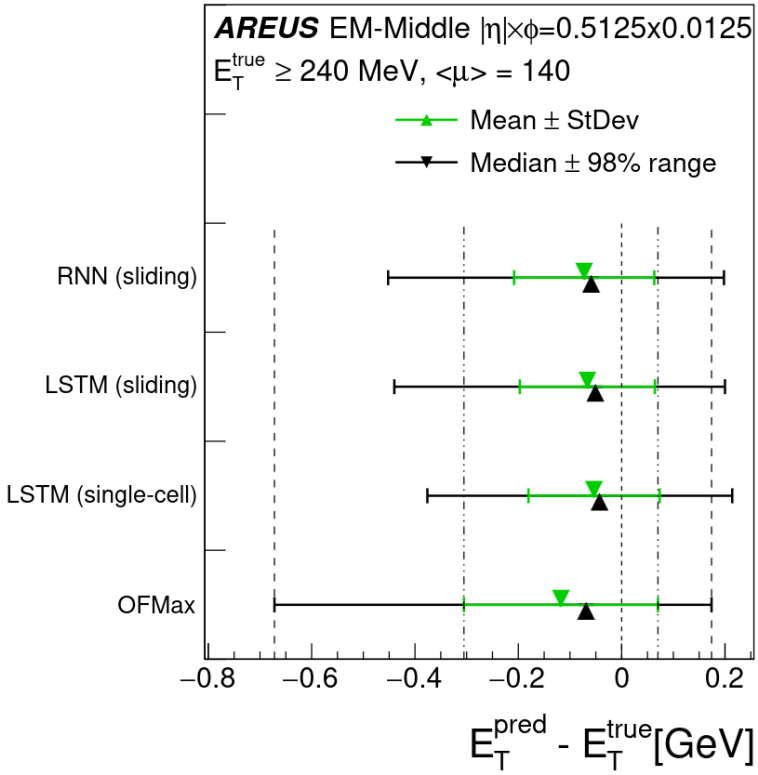


Figure 48 – Transverse energy resolution for optimal filtering and the different RNN algorithms. The performance is measured by comparing the real transverse energy deposited in an EMB middle LAr cell ($\eta = 0.5125$ and $\phi = 0.0125$) to the prediction made by the RNN after simulating the sampled pulse with AREUS and assuming $\mu = 140$. Only energies that are 3σ above the noise level are taken into account. The mean, the median, the standard deviation, and the smallest range that includes 98% of the events are shown. [3]

Algorithm	**Number of parameters**	**MAC units**
LSTM (single-cell)	491	480
LSTM (sliding-window)	491	2360
Vanilla-RNN	89	368
Optimal Filtering	5	5

Table 4 – Algorithm comparison in terms of number of parameters and MAC units.

Figures 49 shows the the energy resolution as function of the time gap for the Vanilla-RNN and the LSTM (sliding) algorithms. Only energy deposits above 3σ of the noise threshold were considered. For values below 20 BC, where overlapping pulses occur, the NN algorithms can handle pulse shape distortion caused by overlapping events and can reconstruct energy more accurately than the OF shown in figure 40. In particular, LSTM-based algorithms in single-cell applications are very stable over time gaps since they can access as many BCs in the past as needed during the training phase.

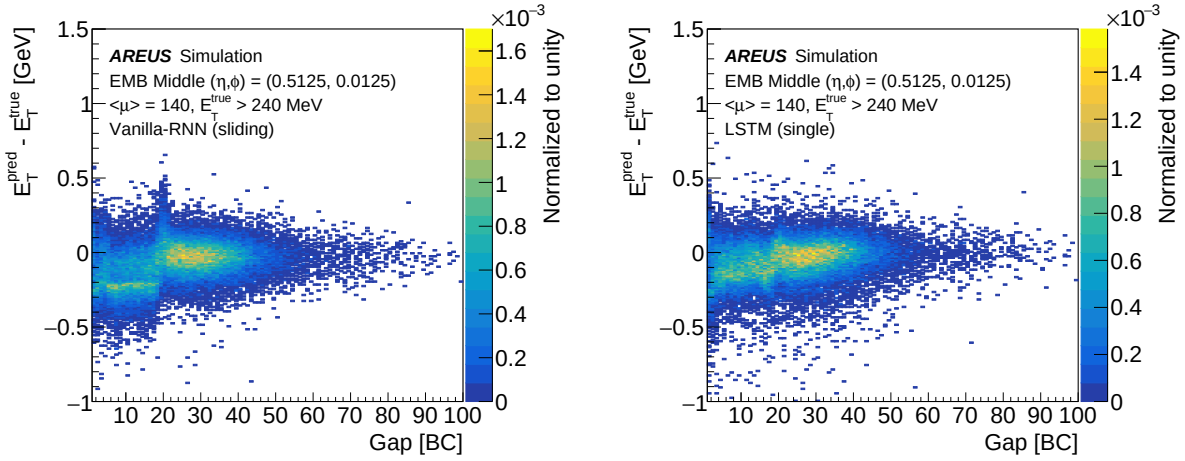


Figure 49 – Resolution as function of the time gap between high energy deposit for LSTM (sliding) and Vanilla-RNN algorithms. [3]

3.5 FPGAS

Field Programmable Gate Array (FPGA) is a programmable electronic device that allows the user to configure its functionality through hardware description language. It consist of several programmable logic blocks that can be inter-wired into a customized configuration to boost application speed. One such block, used for signal processing, is the digital signal processor (DSP). It is possible to customize the digital logic of the FPGA for a wide variety of applications, from signal processing to neural networks and cryptography. The benefits of FPGA include high flexibility, processing speed, low power consumption, relatively low cost compared to custom integrated circuit production, and the ability to make quick and efficient adjustments in real-time systems. FPGAs are also known for their high performance, and fast development time compared to traditional application-specific integrated circuits (ASICs).

3.5.1 Hardware Platform Built with Intel FPGA

The LASP board comprises two Intel FPGAs and is built on an Advanced Telecommunications Computing Architecture (ATCA) platform. FPGAs were a suitable choice for this application as they support long lifecycles, thus reducing the overall cost. Additionally, they

offer fast I/O rates, which are crucial for achieving the required data transmission speed of 200-400Tb/s in the Phase II upgrade. Currently, a demonstrator board for the LASP is available, with two Stratix 10 FPGAs providing enough inputs/outputs for the processing of data from three FEB2s per LASP FPGA. Each FPGA is required to process 384 channels. However, the Agilex FPGA will be used for the final board. Since Agilex was not available for purchase at the time of writing and the corresponding simulation software was not avaliable on time, the implementation was done on Stratix-10 FPGA. Additionally, it is worth noting that the Stratix-10 FPGA used for simulations has a maximum capacity of 2.8 million logic elements, 6,552 DSP blocks, and 77,728 Kbits of on-chip memory. In contrast, the Agilex FPGA, which was selected for Phase II, has a larger capacity of up to 5.5 million logic elements, 13,056 DSP blocks, and 294,912 Kbits of on-chip memory. Despite this difference, the simulations on the Stratix-10 provide valuable insights into the performance and resource utilization of Intel FPGAs.

3.5.2 Intel High-Level Synthesis

High-Level Synthesis (HLS) is a language used in electronic circuit design that allows for the transformation of a high-level functional description into a hardware implementation. This means that designers can express circuit functionality at a higher level of abstraction, and the corresponding hardware logic is generated automatically. Compared to low-level synthesis, which requires designers to code logic at a much lower and specific level, HLS is a more advanced and efficient design method. The primary advantage of using HLS is that it decreases design time and production costs, enabling designers to create more intricate circuits in a shorter amount of time.

The HLS process starts with a description in a high-level language such as C, C++ or SystemC. From this description, the HLS compiler automatically generates the corresponding hardware logic, including elements such as registers, memory, and other required logical components. The final result is an Register Transfer Level (RTL) code file that can be used to implement the circuit on an FPGA or ASIC device. The advantage of using HLS is that designers do not need to be experts in Hardware Description Language (HDL), as most of the work is done by the HLS compiler. This allows designers to focus more on circuit functionality and less on the low-level details of its implementation. Additionally, it allows for better code portability, as the same high-level description can be used to generate implementations on different devices.

3.6 High Level Synthesis for Machine Learning

High-Level Synthesis for Machine Learning (HLS4ML) is a powerful tool for accelerating machine learning applications by transforming high-level models into hardware implementations. The software is built as an open-source Python library that can efficiently translate machine learning models, developed using popular frameworks like TensorFlow, PyTorch, and

Keras, into HLS code. This approach allows for the automatic generation of optimized hardware circuits that can be deployed on FPGAs and ASICs for high-performance inference.

HLS4ML provides a user-friendly interface that allows users to easily configure and optimize the hardware design for their specific application needs. Users can set various parameters, such as the number of processing elements, memory hierarchy, and precision of arithmetic operations, to achieve the desired balance between performance and resource utilization. This flexibility enables developers to easily explore different design options and trade-offs, without the need to modify the code. Moreover, HLS4ML can also provide valuable insights into the behavior of the underlying neural networks. It can generate detailed reports on resource utilization, performance, and energy consumption for a given hardware configuration, allowing developers to better understand the impact of different design choices on the accuracy and efficiency of their models.

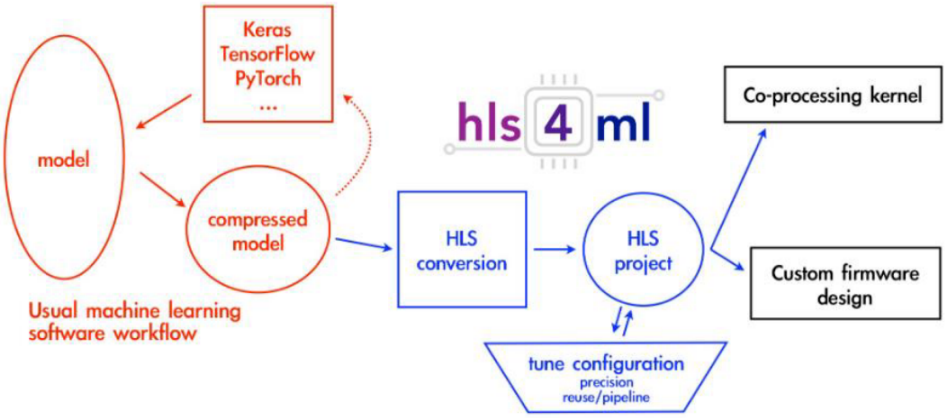


Figure 50 – Schematic view of the HLS4ML workflow division.

The HLS4ML workflow can be divided into 3 main blocks, as represented in Figure 50. In the red section, input data is pre-processed for use in the model. Pre-processing is necessary to transform raw data into a suitable form for use by the model. This may include normalizing data, feature selection, dimensionality reduction, handling missing values, transforming categorical data into numeric data, among others. These techniques help improve the quality of data and, consequently, the performance of the model. The model is then trained using a training dataset to adjust its weights. Training is performed using a machine learning library such as Keras, TensorFlow, or PyTorch.

In the blue section, the model is converted into HLS code to be implemented in an FPGA or ASIC. The python code first converts the NN model into C++ code that can be synthesized into a digital circuit. Each layer and non linear function is implemented as a separate configurable module tailored to perform that specific operation. During the conversion process, these modules are combined to perform the inference for a full ML model. The entire model is synthesized into an IP core that can be integrated into a full application. Due the fact each layer is tailored to that specific model, the resource usage can be optimized because the layer can be configured to

skip multiplications by zero weights, which is known as zero suppression. Finally, in the black section, the HLS is subsequently synthesized, compiled and simulated in Intel Quartus HLS or Xilinx Vivado HLS. Also options are provided to export the RTL available.

3.7 RNN Algorithms Implementation

One of the great advantages of HLS4ML is its ability to support various types of neural network architectures. For example, HLS4ML supports convolutional networks, which are widely used in computer vision applications, and dense networks, which are widely used in classification and regression tasks. However, HLS4ML did not support the Intel FPGAs and the RNNs. In this section, the implementation of RNNs for Intel FPGAs in HLS4ML is presented with its performance results. The implementation is publicly available on the HLS4ML git [52].

3.7.1 Implementation in HLS4ML

To implement RNN support in HLS4ML, new parameters were introduced that allow customization of the program's functionalities for different RNN models. These new parameters include:

- Return sequences: When this parameter is enabled, the layer generates an output for every element in the sequence, functioning as either a "many to many" or "one to many" configuration, depending on the model specification. Conversely, when the parameter is disabled, the RNN layer solely produces the output associated with the final input in the sequence (many-to-one approach). The one-to-one approach doesn't require the use of return sequences.

- Batch Input Shape: Refers to the format of the input data that is supplied to the network during inference. Since RNNs are tailored for processing sequential data, the input is typically represented as a sequence of vectors, where each vector corresponds to a single timestep within the sequence. The batch input shape specifies the number of timesteps in the sequence as well as the dimensionality of each vector. It is a crucial parameter that must be established prior to either training or inference as it governs the input data structural organization that the RNN will be processing.

- Internal Units: The number of bits that are used for mathematical operations and transmitted between the internal components of the system. Values are set per layer or per model through a configuration file. The internal units can impact the performance of a neural network, as larger internal units may require more memory and computational resources to process, while smaller internal units may limit the network performance for complex patterns in the data. Therefore, finding the optimal balance between internal units and

computational resources is an important consideration in designing and training neural networks and will be discussed later in this chapter.

- Activation Functions: selection of non-linear functions that introduce non-linearity in the output of a neuron according to the chosen model.

- Look-up table (LUT) for activation functions: Precomputed table that stores the values of the activation function for a range of input values. Instead of computing the activation function for each input value during forward propagation, the precomputed value is retrieved from the table. Through a parameter file, the user can define the resolution of the LUT. The use of LUTs can significantly reduce computational resources, especially when dealing with complex mathematical functions.

Both LSTM and Vanilla-RNN were implemented in a generic C++ code that allows for customization of input parameters as previously described. Each layer is adapted based on the chosen RNN model, allowing for the creation of N layers of the model with their respective parameters, providing users with a flexible and customizable solution for RNNs for Intel FPGAs. To validate the flexibility of the implementation, as well as its performance and firmware resolution, various RNN models were executed in both HLS4ML and Keras. Keras will serve as reference.

One of the models that was tested is an LSTM utilizing tanh and sigmoid as its non-linear functions, while another model is a Vanilla-RNN employing ReLU as its activation function. Both RNNs models were created using Keras. These models were trained and tested using a dataset of 2 million values simulated by AREUS as explained in section 3.1.1. The dataset is divided into 1 million values for training the models and another 1 million for testing. The LSTM and Vanilla-RNN, respectively illustrated in Figure 51 and Figure 52, show the difference between the value obtained using the tool Keras and the simulation in Quartus HLS4ML. The root mean square (RMS) obtained for LSTM and Vanilla-RNN, respectively, are about 0.0033 [GeV] and 0.0031 [GeV], with a maximum peak at zero. This proves that the implementation of RNNs was successful. The difference presented between the Keras values and HLS4ML occurs because the Keras values use floating-point precision, while HLS4ML is a hardware implementation tool that uses fixed-point precision. There is also the effect of the LUT.

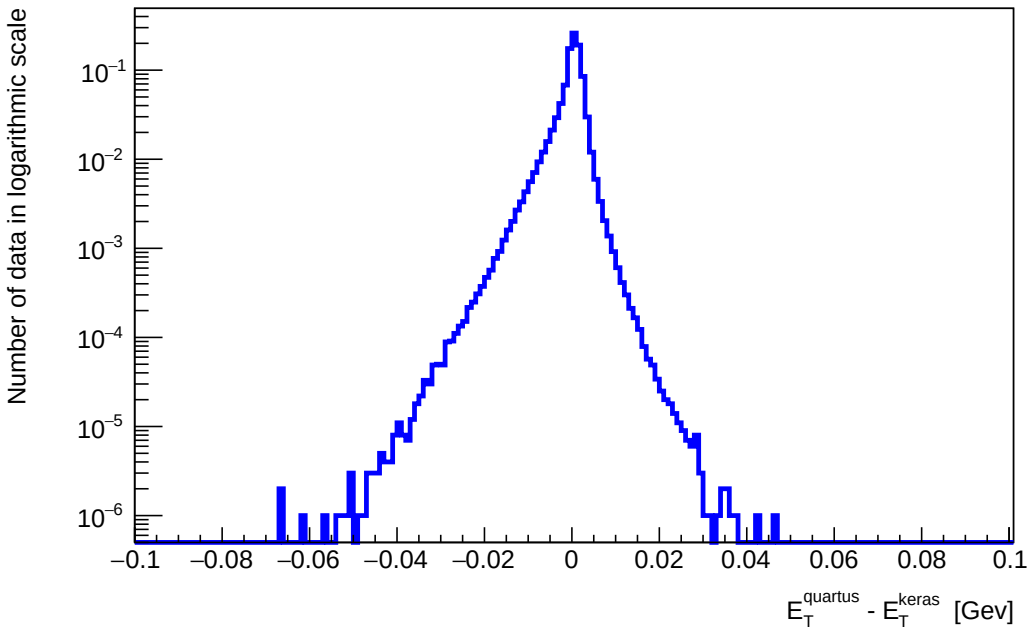


Figure 51 – Difference between the tranverse energy computed with Quartus and the one computed with Keras. The Quartus values are obtained using LSTM implementation in HLS4ML.

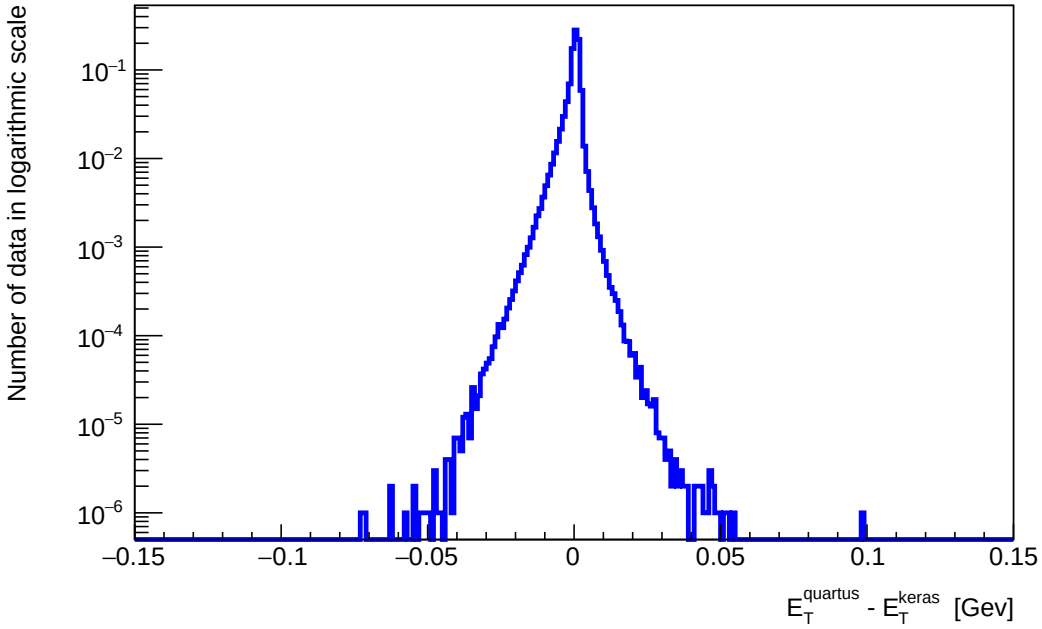


Figure 52 – Difference between the tranverse energy computed with Quartus and the one computed with Keras. The Quartus values are obtained using Vanilla-RNN implementation in HLS4ML.

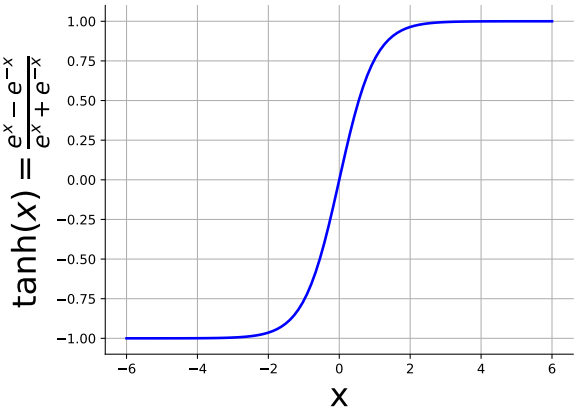
3.8 RNN Optimization and Results

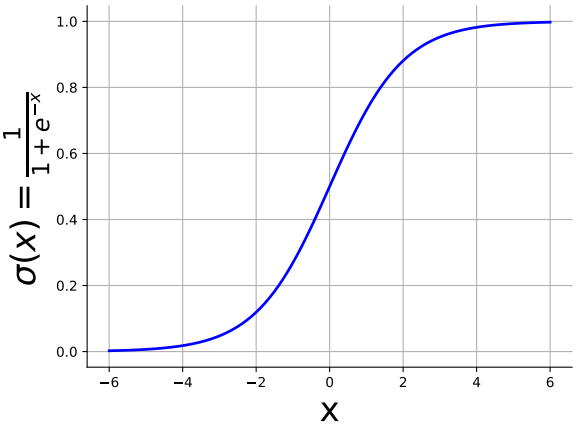
Several optimizations are performed to increase system performance and make better use of the available resources. The following subsections will detail the optimizations and their

respective outcomes. All optimizations have been incorporated into the HLS4ML library.

3.8.1 Look-Up-Table Size

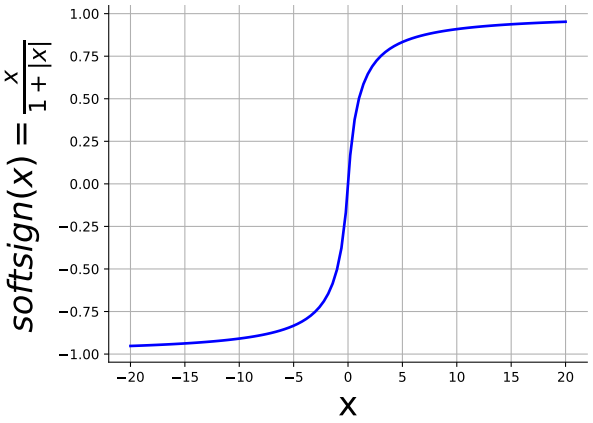
Figure 53 illustrates the sigmoid, hyperbolic tangent (tanh), and softmax activation functions and a symmetry around the X-axis can be observed. By exploiting the symmetric properties of these functions, such as $f(-x) = 1 - f(x)$ for softsign and tanh, and $f(-x) = -f(x)$ for sigmoid, it is possible to reduce the size of the LUT while maintaining the same level of precision. This optimization technique for softsign and tanh involves storing only the function values at x > 0, while the values at x<0 deduced from it. Similarly, the same method can be applied to softsign, where only the function values in x > 0.5 are stored, and the values at x<0.5 are deduced accordingly. This approach improves the resolution by a factor of two while keeping the same LUT size.





$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad (3.10)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (3.11)$$



$$softsign(x) = \frac{x}{1 + |x|} \qquad (3.12)$$

Figure 53 – Representation of various nonlinear functions that are used as activation function for neural network.

The figure 54 shows that the resolution (represented by the error bars) reaches a minimum at a LUT size of 1024 while we observe a small decrease in performance at a LUT size of 512 specialty for the LSTM network. For every LUT size used the firmware resolution is less than the intrinsic NN resolution. As demonstrated in figure 55, which shows the consumption of Adaptive Look-Up Tables (ALUTs), Flip-Flops (FF), Random Access Memory (RAM), and Digital Signal Processors (DSP) of the FPGA, there is a proportional increase in resource usage, especially in the size of the LUT, starting from 1024. The RAM memory of the FPGA is responsible for storing the LUT, which is why only the RAM usage increases.

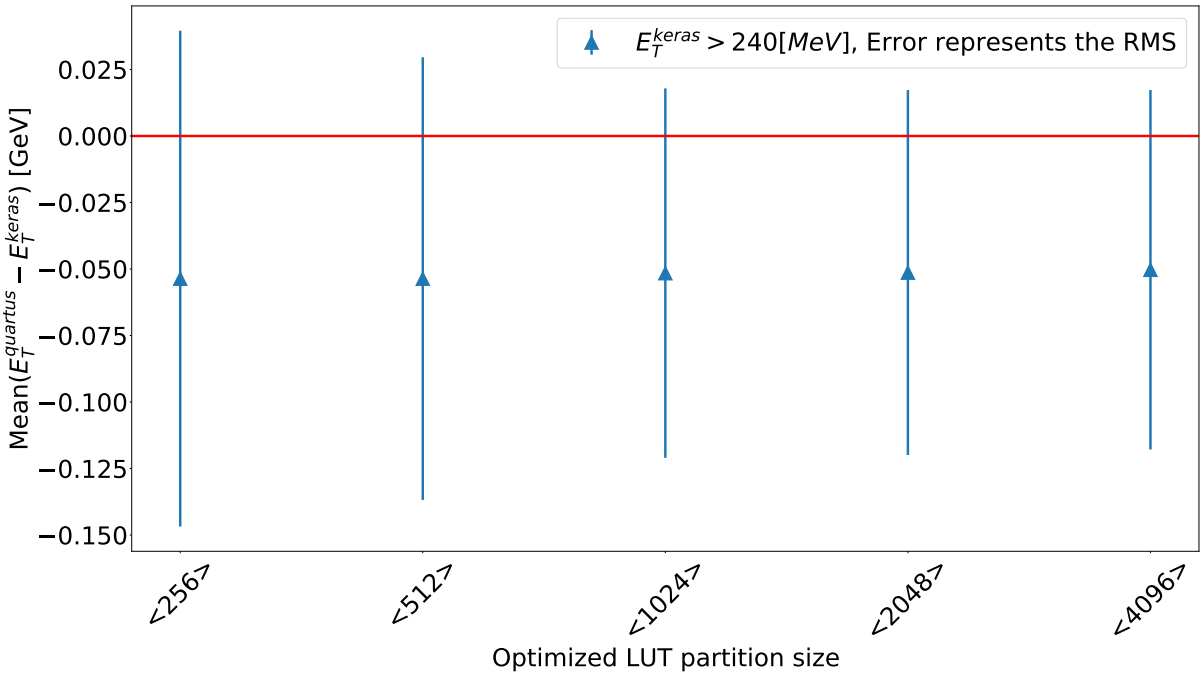


Figure 54 – Firmware energy resolution in a Stratix 10 FPGA as function of the LUT size for an LSTM network at a target frequency of 400 MHz. The markers show the mean value while the error bars show the RMS.

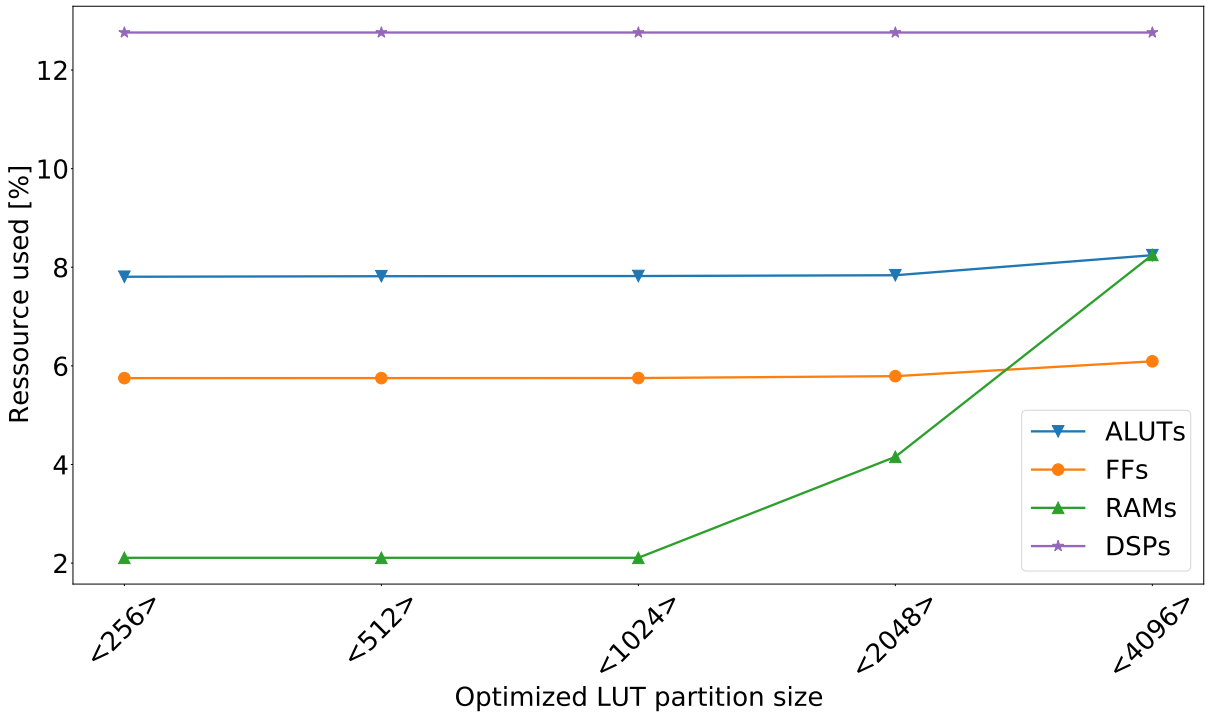
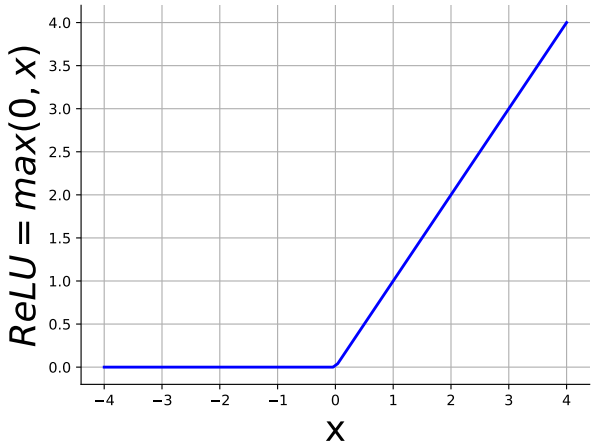


Figure 55 – Resource usage in a Stratix 10 FPGA as function of the LUT size for a LSTM network at a target frequency of 400 MHz.

The Vanilla-RNN model does not use symmetric nonlinear functions. Rectified Linear Unit (ReLU), represented on Figure 56 and defined by Equation 3.14, is more adapted to Vanilla-RNN due the fact it is resilient to the gradient vanishing ploblem. ReLU is not stored in the LUT because the computational resources needed to calculate the formula 3.13 are lower compared to the resources required for storing the values in the LUT.

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{else} \end{cases} \tag{3.13}$$



$$ReLU(x) = max(0,x) \tag{3.14}$$

Figure 56 – Representation of the activation function ReLU.

3.8.2 Resource Usage as Function of the Frequency

The FPGA's capacity to handle 384 channels poses a challenge due to the substantial number of multiplications required for LSTM (752,640) and Vanilla-RNN (116,736) processes. However, the Stratix 10 FPGA, with its 11,520 multiplication blocks, falls short of meeting these demands directly. To address this, multiplexing becomes essential, enabling simultaneous computation of multiple networks within the FPGA. Implementing multiplexing involves optimizing each neural network instance to cater to multiple channels, necessitating an adjustment in the DSP block computation frequency, which can be increased as a multiple of the input frequency (40 MHz). However, ramping up the frequency introduces complexities, notably requiring additional logic to synchronize the network. This surge in frequency disproportionately affects the number of ALUTs, leading to a substantial increase, whereas the DSP blocks remain relatively stable due to their capability to operate at high frequencies as shown in Figure 57 for LSTM for and in Figure 58 for Vanilla-RNN . Consequently, the challenge emerges where higher frequencies become a limiting factor for Vanilla-RNN and LSTM, specifically when surpassing 625 MHz, as it strains the FF or ALUTs.

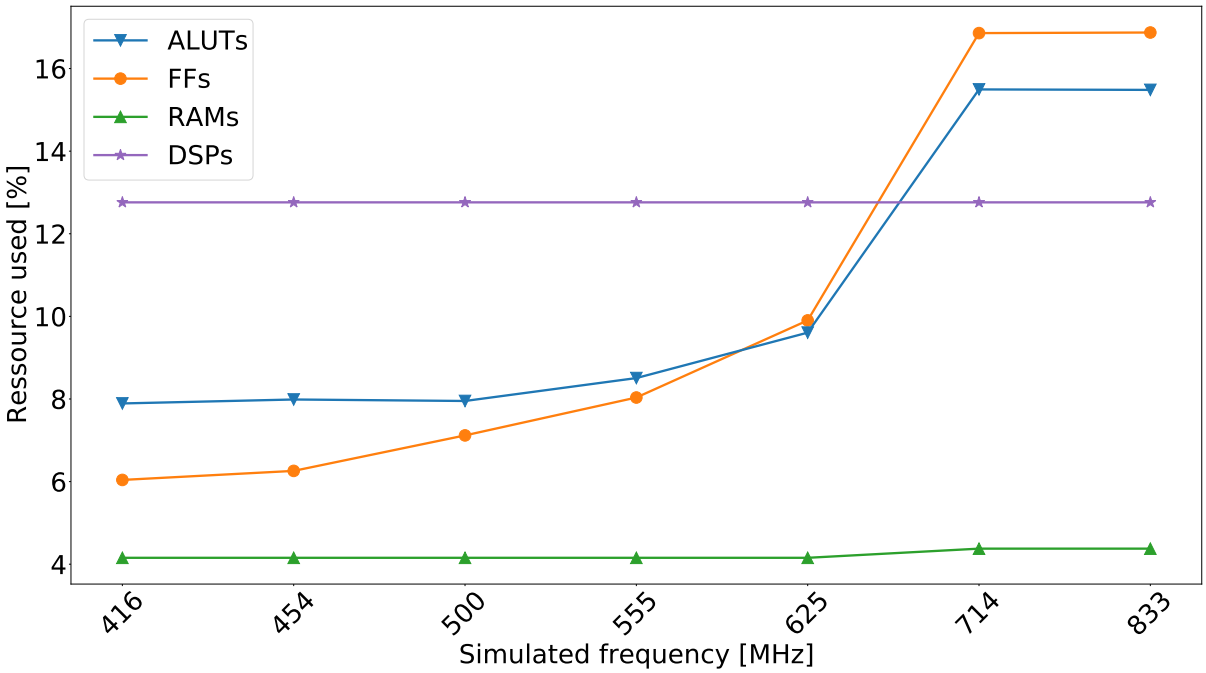


Figure 57 – Resource usage in a Stratix 10 FPGA as function of the frequency for a LSTM network.

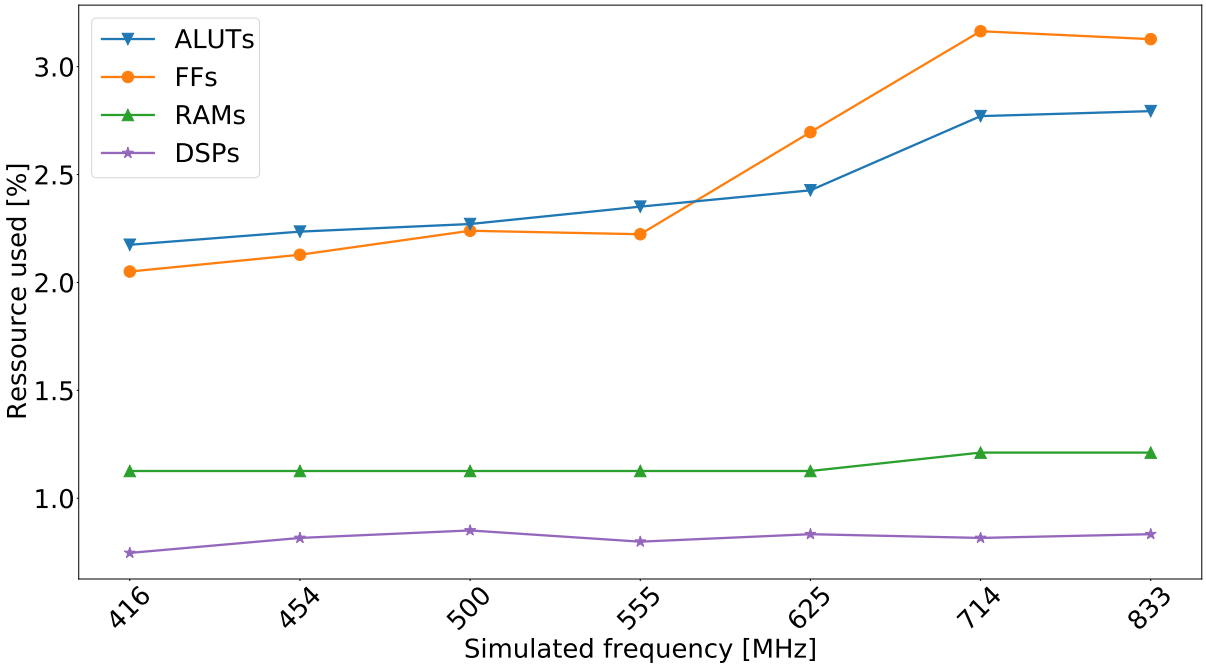


Figure 58 – Resource usage in a Stratix 10 FPGA as function of the frequency for a Vanilla-RNN network.

3.8.3 Internal Fixed Point Size

The Digital Signal Processor (DSP), is an electronic component designed to perform high-speed mathematical operations on digital signals. The Intel Stratix 10 devices offer various operating modes and features that are supported by the variable precision DSP block. Each Stratix 10 DSP contains two 18x19-bit multipliers and can perform a maximum multiplication of

27x27 bits using both multipliers. Additionally, the DSP can be configured to perform fixed-point or floating-point operations as shown in Table 5. From the design of the FEB2, the ADC output has 16 bits, and 18 bits are enough to represent the weights with a good resolution. Therefore, the better balance between resource usage and resolution for the Stratix 10 board is to configure the DSP to use two 18x19-bit multiplications.

Variable-Precision DSP Block	**Operation Mode**	**Supported Operation Instance**
1 variable precision precision DSP block	Fixed-point independent 18 x 19 multiplication	2
	Fixed-point independent 27 x 27 multiplication	1
	Fixed-point independent 18 x 19 multiplier adder mode	1

Table 5 – Intel Stratix 10 devices offer various combinations of operational modes and instances that are supported by the Variable Precision DSP Block.

In the next section, optimizing the integer and decimal components for arithmetic operations using fixed point precision will be undertaken.

3.8.3.1 Integer Fixed Point

In order to achieve the best performance in a neural network, it is necessary to find a balance between the use of available resources and the achieved performance. Simulations were performed to determine the appropriate number of integer bit for each type of network.

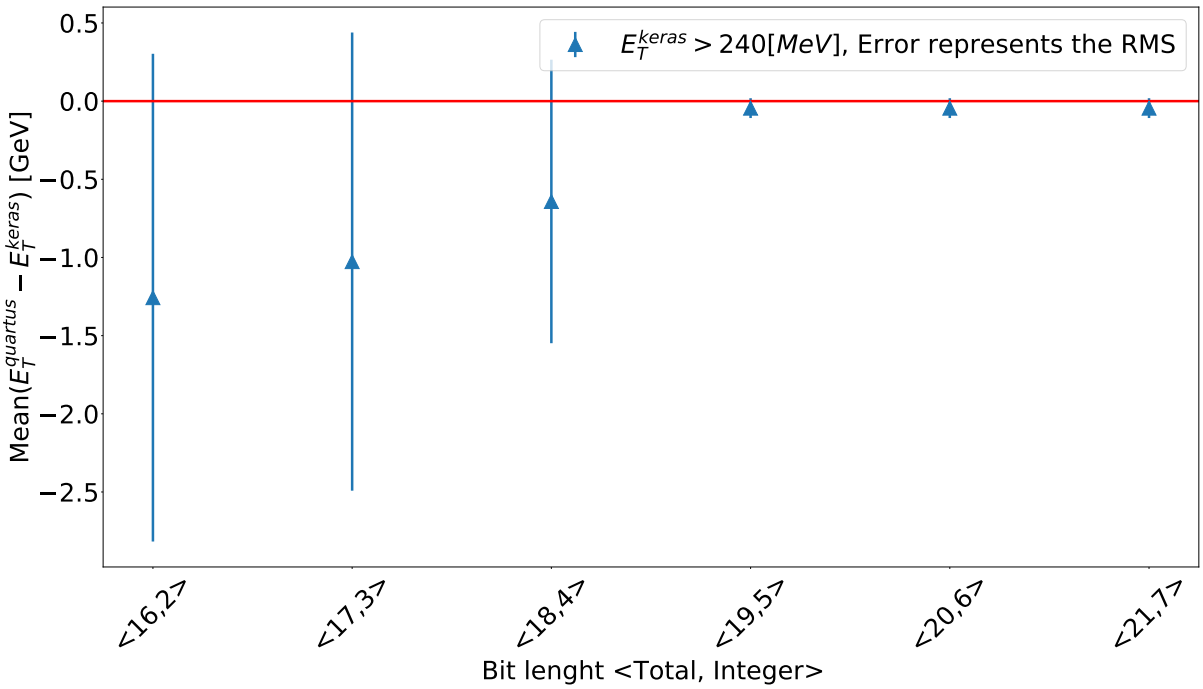


Figure 59 – Firmware energy resolution in a Stratix 10 FPGA as function of the Bit Width size for an LSTM network at a target frequency of 400 MHz. The markers show the mean value while the error bars show the RMS.

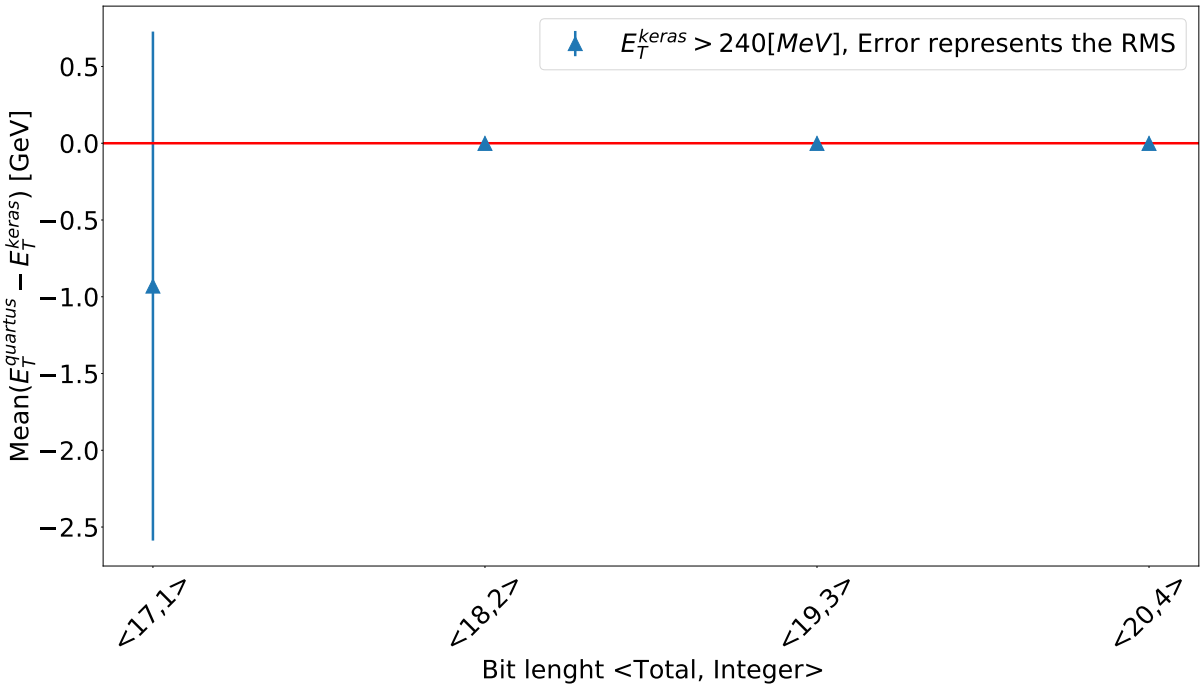


Figure 60 – Firmware energy resolution in a Stratix 10 FPGA as function of the Bit Width size for a Vanilla-RNN network at a target frequency of 400 MHz. The markers show the mean value while the error bars show the RMS.

In the case of the LSTM network, five integer bits are needed to achive good resolution, as can be seen in the Figure 59. For the Vanilla-RNN network, however, two integer bits were sufficient to achieve an acceptable result (Figure 60). However, when increasing the number of bits used to represent the integer part of the value, for both LSTM and Vanilla-RNN, a significant increase in the use of DSPs was observed in figures 61 and 62, especially, for values above 19 total bits, where there is a DSPs jump. The jump at 19 total bits correspond to the max bit width that a DSP can handle in the 2 seperated multiplication mode. This elucidates the necessity for the DSPs' sudden increase in order to carry out the multiplications.

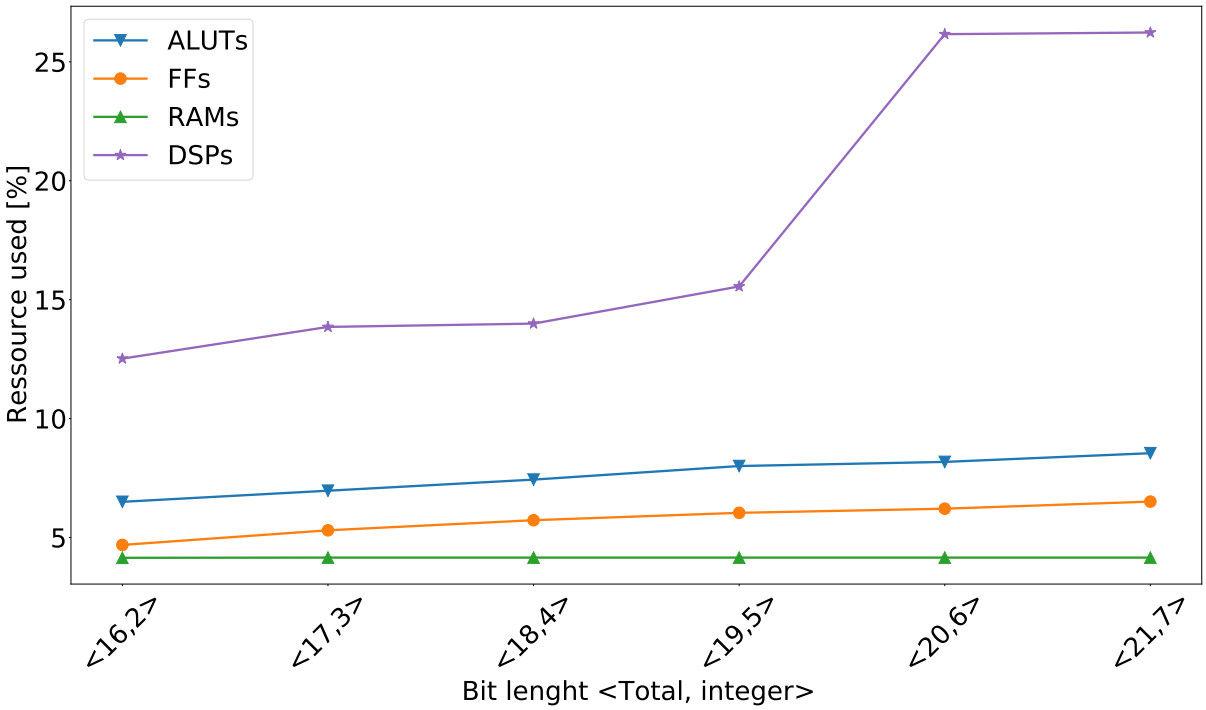


Figure 61 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for a LSTM network at a target frequency of 400 MHz

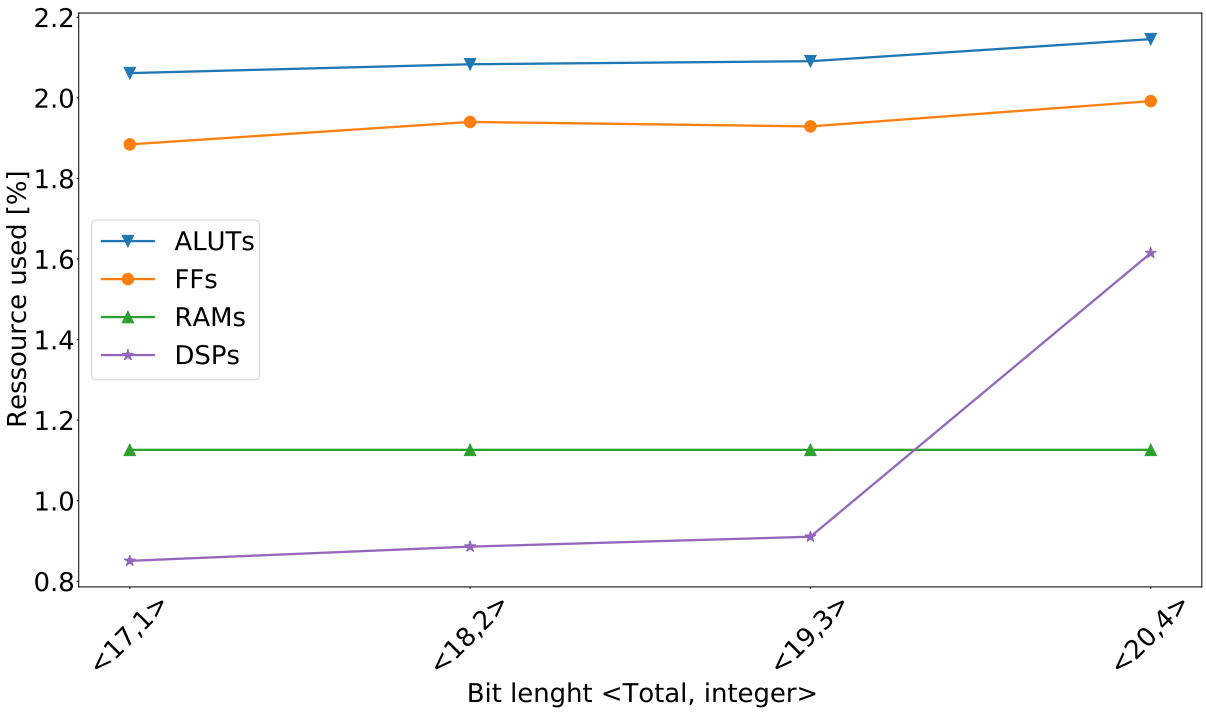


Figure 62 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for a Vanilla-RNN network at a target frequency of 400 MHz

3.8.3.2 Decimal Fixed Point

Figure 63 shows that the resolution starts to level off at around 8 decimal bits. Below this value the resolution degrades significantly. On the other hand, Vanilla-RNN require eleven

decimal bits to achieve acceptable results, as illustrated in Figure 64. Increasing the number of bits used to represent the decimal part for both LSTM and Vanilla-RNN resulted in a significant increase in DSP usage, as depicted in Figures 65 and 66, since more DSPs are needed for multiplications. The jump at 19 total bits corresponds to the maximum bit width that a DSP can handle in the two separated multiplication mode. The increase in the number of flip-flops is explained by the fact that it is necessary to store more information corresponding to the increase in bit width size. For LSTM, 5 integer bits and 8 decimal bits were chosen, while for Vanilla-RNN, 3 integer bits and 11 decimal bits were chosen to be the best compromise between resource usage and resolution. It insures a resolution of less than 1 per mil for the firmware implementation.

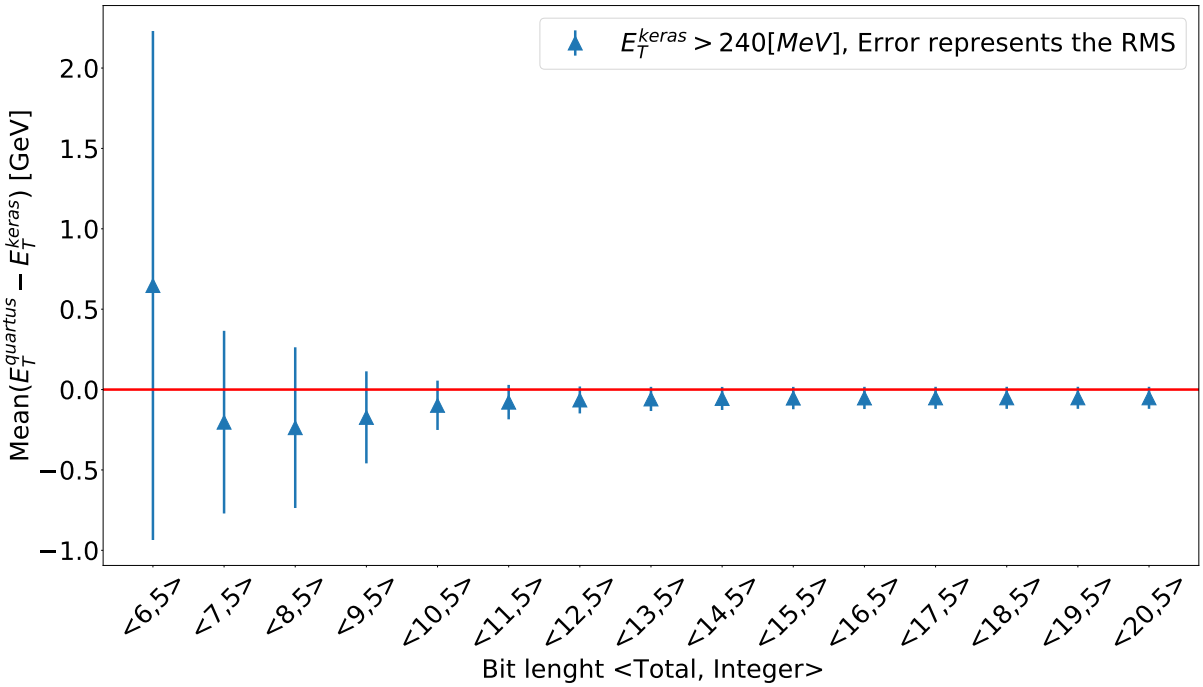


Figure 63 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for LSTM network at a target frequency of 400 MHz. The markers show the mean value while the error bars show the RMS.

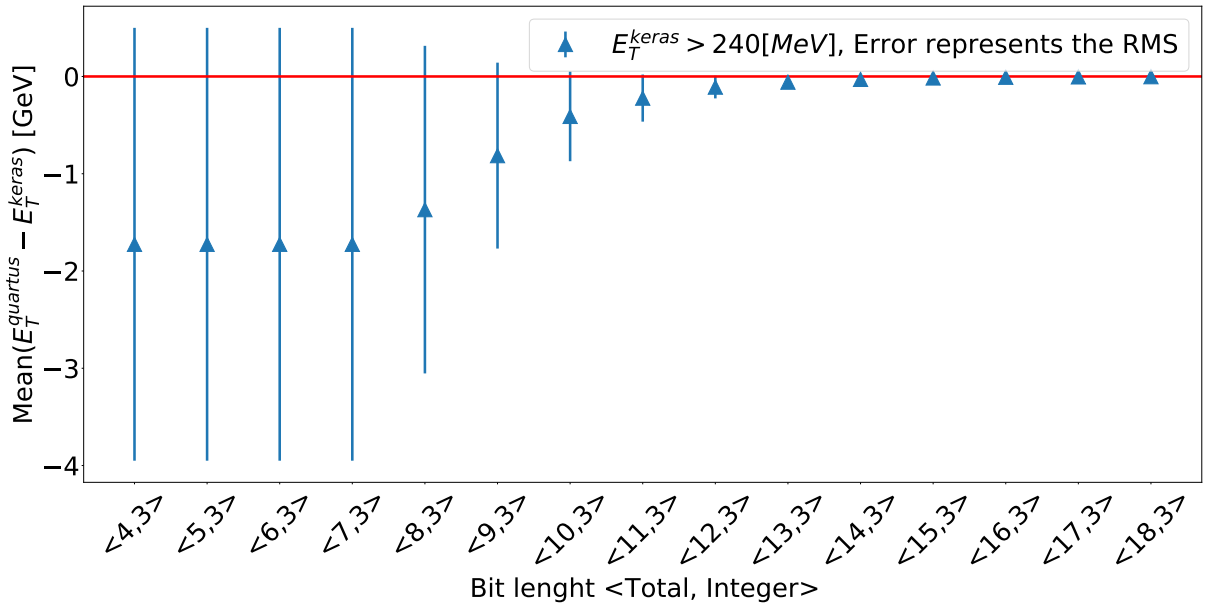


Figure 64 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for a Vanilla-RNN network at a target frequency of 400 MHz. The markers show the mean value while the error bars show the RMS.

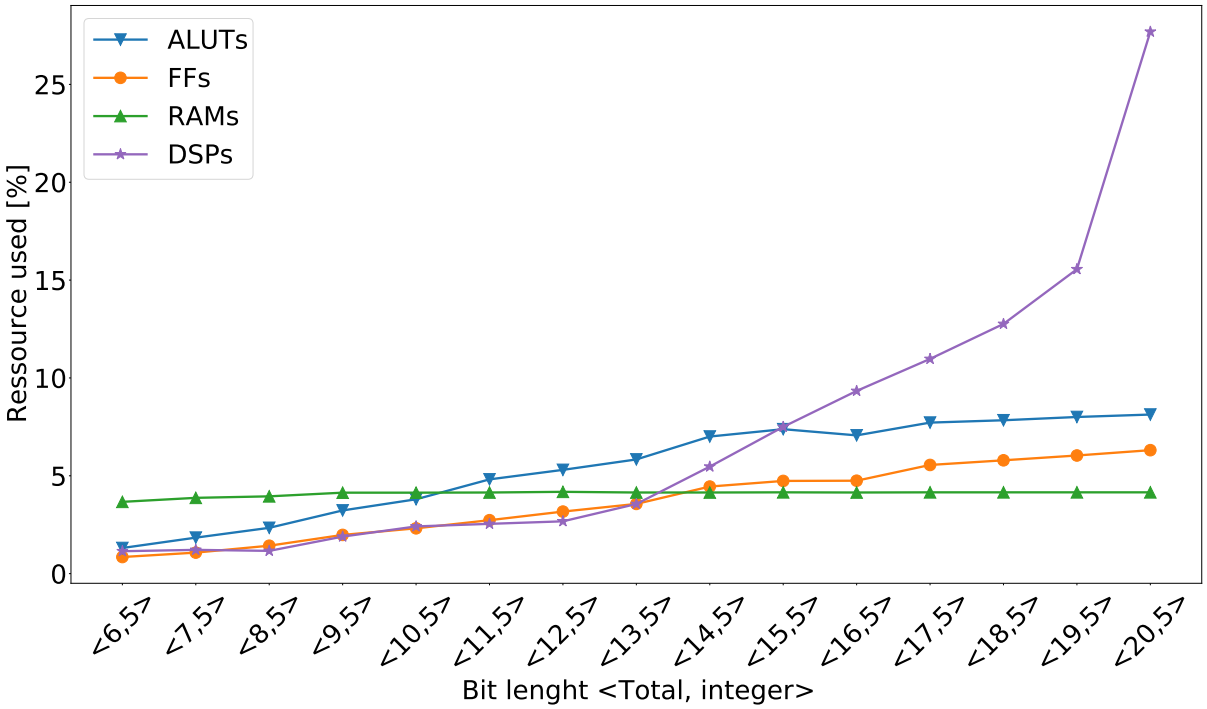


Figure 65 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for an LSTM network at a target frequency of 400 MHz.

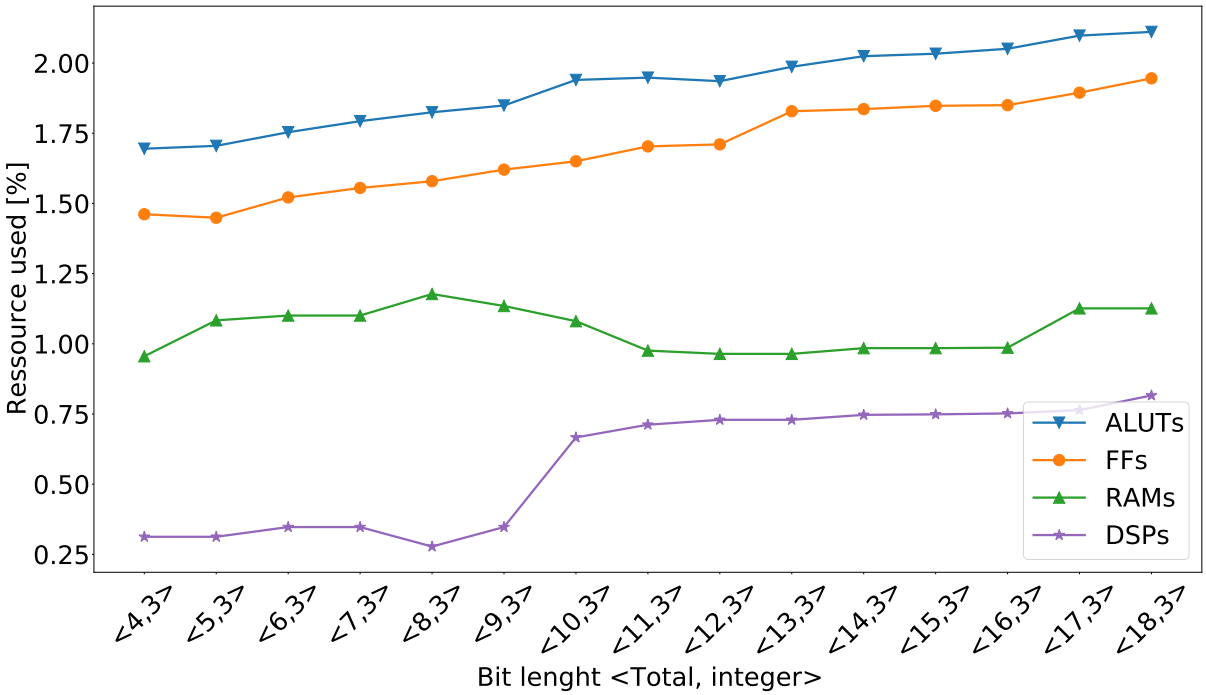


Figure 66 – Resource usage in a Stratix 10 FPGA as function of the Bit Width size for a Vanilla-RNN network at a target frequency of 400 MHz

3.9 Conclusion

The experiments conducted demonstrate that RNNs outperform the optimal filtering algorithm in energy reconstruction within the ATLAS LAr calorimeter, particularly in the overlapping region where multiple pulses are present. These neural networks have been specifically designed to minimize resource utilization while maintaining high performance. Among them, Vanilla-RNN emerges as a promising candidate capable of meeting the stringent requirements of the LASP firmware [2]. The utilization of HLS4ML offers a convenient solution for adapting and optimizing the neural networks' parameters to facilitate their implementation on FPGA platforms. Additionally, it proves to be a valuable tool for rapid prototyping. Notably, LSTM and RNN have been successfully implemented in HLS4ML for Quartus.

CHAPTER

4

FIRMWARE DEVELOPMENT FOR EVALUATING EMBEDDED RECURRENT NEURAL NETWORKS ON THE STRATIX 10 DEVKIT

The firmware implemented in HLS facilitates rapid initial prototyping and network architecture optimization. However, the HLS implementation falls short in achieving both the required frequency to accommodate 384 channels per FPGA and the mandated latency levels [2]. As a result, it does not meet the specified requirements as discussed in chapter 3. VHDL is employed to further enhance the HLS implementation and impose placement constraints on the physical components of the FPGA, thereby achieving compliance with the specified criteria for the Phase-II of the firmware as explained in reference [2].

This chapter introduces a testing method to validate the NN firmware on the hardware. To do that a test firmware, capable of providing input data to the NN firmware and to extract the output of the NN, is developed. Section 4.1 summarizes the VHDL implementation of the NN firmware and its characteristics. Details about the Intel Stratix 10 Development Kit used for the hardware test, including the essential hardware components that it encompasses, are described in Section 4.2. Section 4.3 explores the overall structure, methodologies, and techniques employed in each component constituting the test firmware structure. The structure presented in this section enables the retrieval of data generated by the NN from the FPGA. Section 4.4 focuses on validating the test firmware using the ModelSim simulation tool. The validation of the test firmware on the hardware is described in section 4.5. Section 4.6, describes how the output of the NN firmware is extracted from the board. Section 4.7 analyzes the performance of the NN by comparing the expected values and the NN results extracted from the FPGA. Finally, Section 4.8 provides the conclusion.

4.1 NN firmware in VHSIC Hardware Description Language (VHDL)

The NN system employed in this section involves the Vanilla-RNN operating with the sliding-window technique described in Section 3.4. The training is done in Keras, utilizing simulated input data from AREUS. The network is fed five sequential samples of the electronic pulse from an individual channel that is simulated with AREUS as described in section 3.1.1. These inputs are used to compute the corresponding transverse energy. These five samples span a time window encompassing five consecutive bunch crossings. The computed energy corresponds to the energy deposit at the second bunch crossing. While four of the samples span the peak region of the pulse originating from the energy deposition, the first sample is chosen to be before the start of the pulse. This temporal arrangement permits the mitigation of effects from overlapping pulses arising from preceding energy deposits. The architecture of the neural network comprises five successive RNN cells, followed by a dense layer, as depicted in Figure 67. Each cell is responsible for processing the input derived from one sample, pertaining to a single bunch crossing. Details about the NN implementation in VHDL can be found in reference [2].

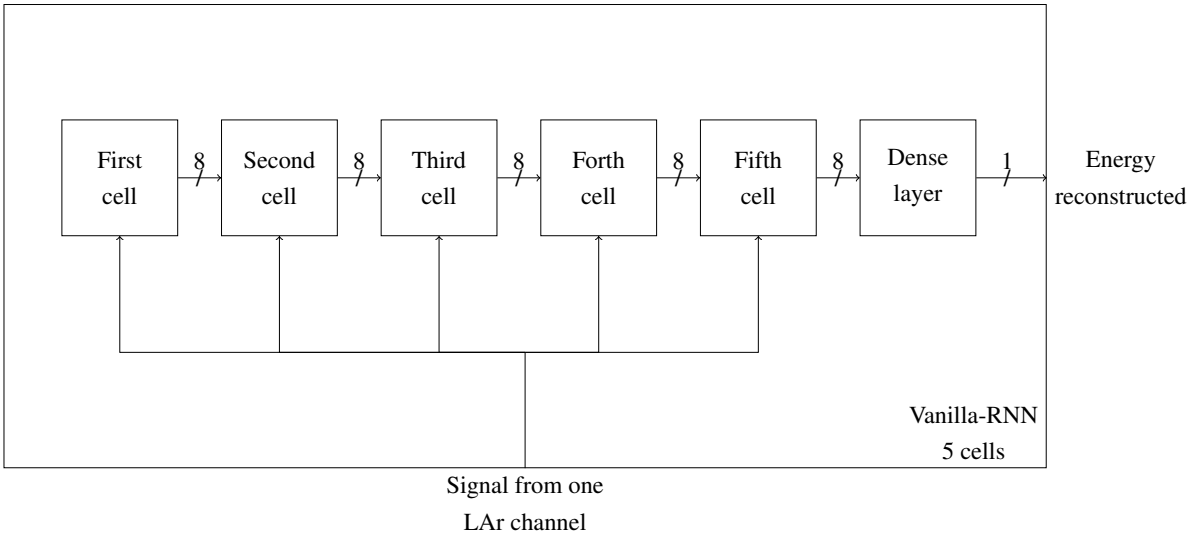


Figure 67 – Diagram depicting a Vanilla-RNN which consists of 5 cells, each having an internal dimension of 8 (representing the size of the state vector). These cells are sequentially arranged and connected to each other. The RNN is then followed by a dense layer. Each cell in the RNN takes input data and the state from the previous cell to compute a new state. The first cell solely relies on the input data since it lacks a preceding sample. The dense layer uses the state of the last cell to calculate the transverse energy.

In the Vanilla-RNN setup, various data categories are defined: input and output data, neural network weights, and intermediate computation data within the network blocks. The

internal and input/output groups utilize 19 bits, while the weights use 16 bits.

In Phase II, each FPGA is tasked with energy reconstruction for 384 distinct channels. The computations occur on-the-fly at a collision frequency of 40 MHz. The limited resources of the FPGA inhibit the fitting of 384 NN instances, requiring each neural network instance in the FPGA to handle multiple channels. In this RNN application, 28 instances are employed. Each instance has a multiplexing factor of 14, accommodating a total of 392 channels [2]. This setup requires a clock frequency of 560 MHz to effectively support the multiplexing factor.

4.2 Intel Development Kit

Figure 68 – The Stratix 10 GX development kit employed for evaluating the firmware comprising the neural networks.

The Intel Stratix 10 GX FPGA Development Kit have a similar FPGA as the LASP demonstrator boad. While the LASP demonstrator board is being developed and tested, the development kit board can be used to test the NN firmware.

In the context of FPGA-based design, the development kit provides invaluable capabilities for testing, ensuring the correct implementation, and verifying the desired outputs of the implementation of RNNs on the Stratix 10 GX FPGA. Crucial elements such as the clock crystal and low jitter programmable clocks ensure precise frequency, while the reset button aids in manual restarts for effective debugging. Furthermore, the Joint Test Action Group (JTAG) Universal Asynchronous Receiver/Transmitter (UART), along with Internet communication interfaces, streamlines debugging and data transfer between the FPGA and external devices. For efficient data extraction, the kit offers USB cables, facilitating the retrieval of internally calculated values from the FPGA.

4.3 General firmware test structure

The test firmware is a framework that allows the injection of data into the NN and the retrieval of transverse energies reconstructed by the hardware implementation of the RNN firmware in the FPGA . The output of the RNN firmware will pass through various key components before arriving to the computer terminal, as illustrated in the simplified diagram in Figure 69. The input data and the weights are sent from a RAM to the NN which computes the energy and send it to another RAM. This second RAM is read by the NIOS and the corresponding data is sent to the computer. The test firmware is designed with modularity in mind, facilitating the testing of different types of neural networks firmware.

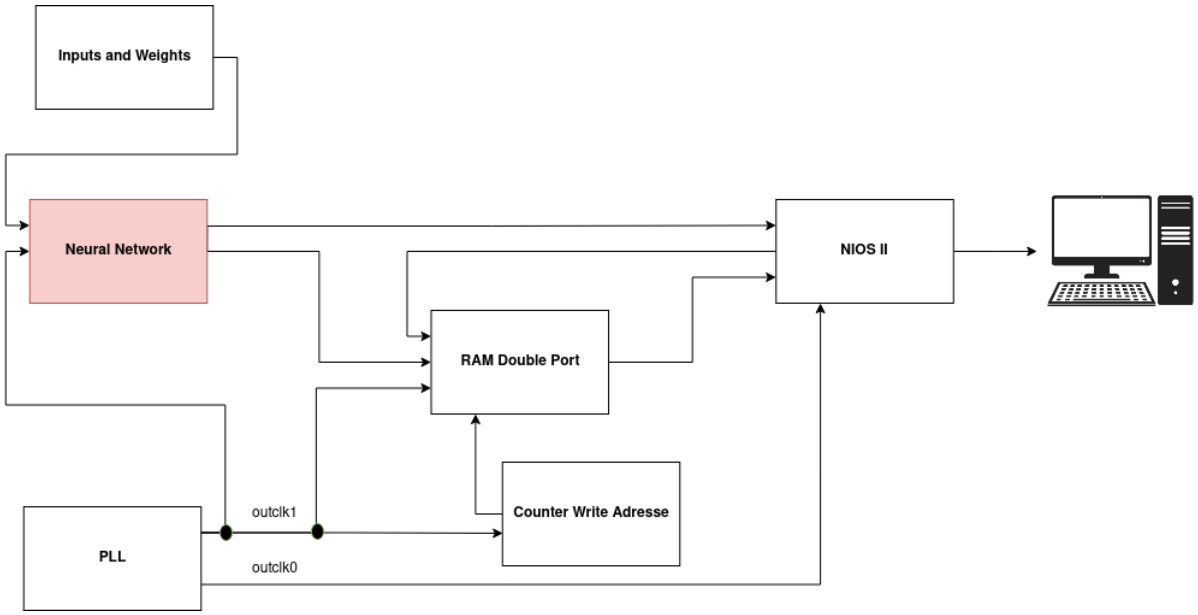


Figure 69 – Simplified schematic depicting the connections between the RNN and the test firmware.

The test firmware plays a crucial role in the RNN evaluation process, as it allows validating the execution of the RNN firmware on an actual microchip. This step can help discover bugs or vulnerabilities in the design. Additionally, hardware testing enables real-time measurement of firmware performance and the ability to conduct stress tests.

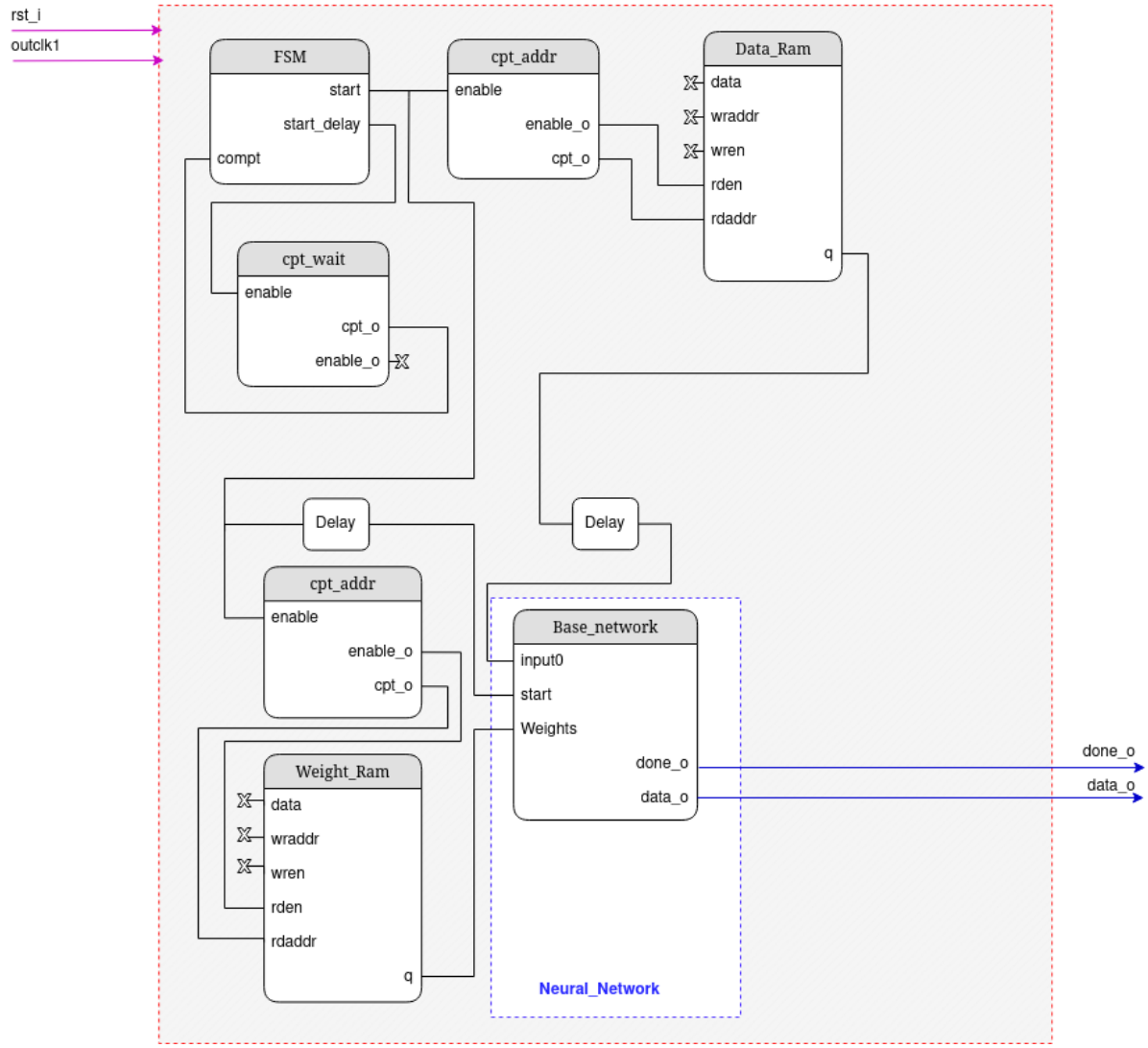


Figure 70 – The schematic illustrates the components of the test firmware that are used to input the data and the weights to the NN. It includes the core **base_network** block with a 5-cell Vanilla-RNN having 8 internal dimensions. Two RAM segments, **Weight_Ram** and **Data_Ram**, store the data. The **cpt_addr** counters manages RAM data transmission with precise timing. **Delays** ensure data synchronization, while **cpt_wait** briefly halts the **full state machine (FSM)** for seamless initiation. The **FSM** coordinates the system.

Figure 70 shows details about the firmware components used to send the input data and weights to the NN. The input and weight values are stored within hexadecimal files, which adhere to the Intel hexadecimal format. These hexadecimal values were generated through a two-step process: first, by transforming the weight and input values from a floating-point format to a fixed-point format, and subsequently, by converting them into the Intel hexadecimal representation. The RAM memory containing the stored electronic pulse samples can hold 259 words, each consisting of 19 bits. The input RAM is connected to the component “cpt_addr”, which, through the Finite State Machine (FSM), manages the control of RAM addresses. The component represented by “Weight_Ram” comprises a set of five RAMs. Each RAM stores a

specific type of weight (bias, kernel, recurrent kernel, dense bias, and dense kernel). All weights consist of 14 words, which correspond to the 14 channels that one instance of the network should treat with a multiplexing of 14. Only the first channel will be used in this test. The recurrent kernel contains 64 values, each represented by 16 bits, totaling 1024 bits per word. The kernel, bias, and dense kernel weights consist of 8 values, each represented by 16 bits, totaling 128 bits per word. The dense bias contains only one value, represented by 16 bits per word. All weights are connected to the address counter ("cpt_addr") component, which, through the FSM, manages the control of the weight RAM in the same way as the inputs. By using the "cpt_addr" component to retrieve the correct value from the memory, the weights and samples serve as input to the base network.

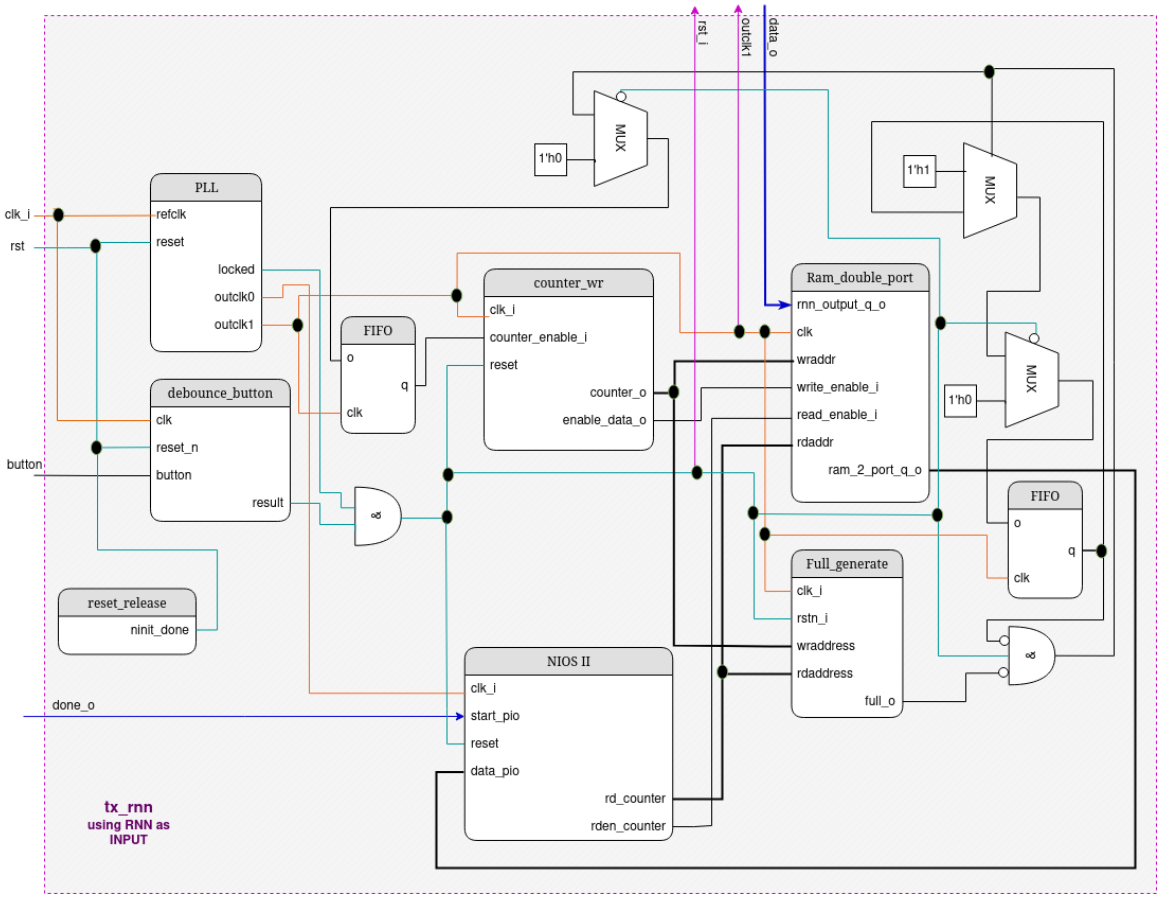


Figure 71 – The schematic illustrates the components of the test firmware that are used to extract the computed RNN values. It includes the **Ram_double_port** that allows the reduction of the output value frequency rate. The **counter_wr** manages RAM write operations with precise timing. The **FIFO** ensures data synchronization, while the **Full_generate** indicates if the RAM is full. The **NIOS II** controls the enable and read address of the RAM. The **PLL** creates two synchronous frequencies. The **debounce_button** and **reset_release** enable proper system reset.

Figure 71 shows a detailed block diagram of the test firmware used for recovering the RNN firmware values. In this diagram, the highlighted signals in blue correspond to the RNN outputs connected to the test firmware. These outputs consist of a 19-bit value and a

signaling component denoted as “done”, which serves to indicate the exact moment when the first transverse energy value is obtained by the RNN firmware. These two distinct signals are connected to a “double-port RAM” (Subsection 4.3.2) and the NIOS II processor (Subsection 4.3.1), respectively. The Nios processor and the “counter_wr” (Subsection 4.3.3) control the read and write addresses of the RAM, as well as the moment at which the values should be read and written. To prevent data in the RAM to be overwritten, the “full_generate" (Subsection 4.3.7) component generates the full signal which deactivates writing new values in the RAM.

The reset signal, depicted in light blue, is generated by the “reset_release”(Subsection 4.3.5) component and then propagated to the PLL and debounce button. The resulting signal is derived from a logical ‘AND’ gate between the debounce button (Subsection 4.3.6) and the PLL (Subsection 4.3.4) locked signal. This signal is then propagated to all other components that require a reset.

The clock signal, depicted in orange, is created from a crystal operating at 25 MHz and is connected to the PLL component, which subsequently generates two other frequencies: one at 70 MHz and the other at 560 MHz. The higher frequency is supplied to the NN firmware, the ”counter_wr”, “full_generate”, and the “ram_double_port”, while the lower frequency is supplied to the NIOS II. The signals depicted in purple represent the clock and reset signals that interface with the RNN firmware.

4.3.1 NIOS II

The Nios II [53] microprocessor was chosen for recovering data from the FPGA since it provides configurable sub-peripherals and the JTAG UART connection. Furthermore, the Nios II is designed to be space and energy efficient. The Nios II gen2 architecture is a 32-bit embedded processor design explicitly tailored for integration with Intel’s family of FPGAs. Notably, the Nios II gen2 framework incorporates libraries for custom instructions which enables specific development for the test firmware design.

Nios II gen2 exists in two distinct configurations: Nios II/f and Nios II/e. The Nios II/e core is optimized for minimal logic usage within FPGA contexts, making it particularly efficient for applications centered around low-cost FPGAs. On the other hand, the Nios II/f core prioritizes maximum performance, albeit at the cost of core size. Both configurations encompass essential resources, including:

- Available IPs with custom instructions.
- Data caches ranging from 512 B to 64 KB.
- Access to up to 2 GB of external address space.
- RAM designated to save software instruction variables.

- A JTAG debugging module.

In the pursuit of maximum performance, the utilization of the Nios II/f configuration was selected. The Nios II development process unfolds in two distinct phases: hardware generation and software development.

The hardware generation phase entails using the Qsys system integration tool [54], an integral component of the Quartus II package. Through a graphical user interface (GUI), it is possible to choose the desired Nios II features, and set and augment the embedded system with peripherals and I/O blocks such as memory controllers and parallel input/output (PIO). With the hardware specifications in place, Quartus II coordinates the synthesis, the placement, and routing operations to implement the entire system on the designated FPGA platform. As for software development management, the Embedded Design Suite (EDS) package needs to be acquired. Built upon the foundation of the Eclipse IDE, EDS encompasses a C/C++ compiler, debugger, and an instruction set simulator. EDS also facilitates the download and execution of compiled applications on the physical FPGA host.

4.3.1.1 Structure of the NIOS II Hardware

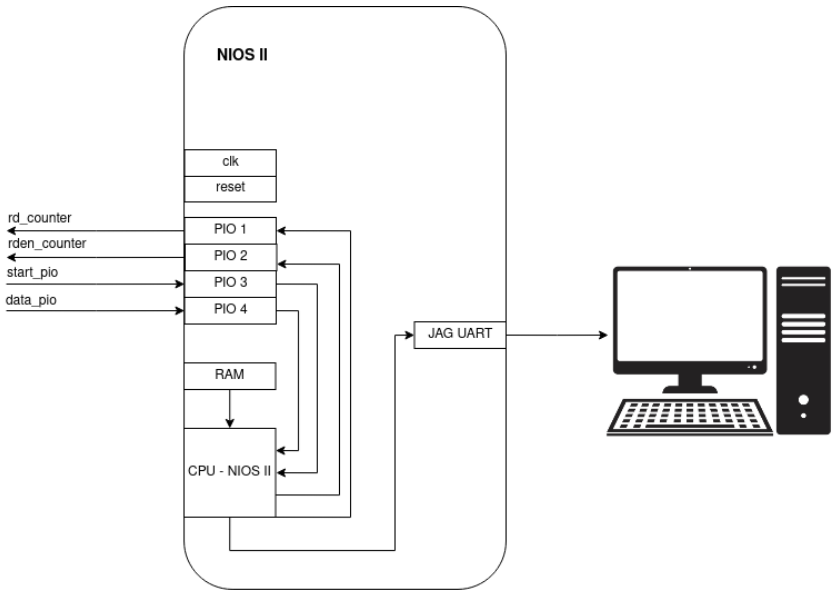


Figure 72 – Components of the NIOS II Hardware Framework.

The NIOS II hardware structure and its components used in this project is depicted in Figure 72. Its structure consists of the following components:

- **NIOS II Processor (CPU):** This processor core implements the Nios II instruction set, as described in the software structure.
- **On-chip Memory (RAM):** This memory enables the storage of variables defined in the software structure.

- **Parallel I/O (PIO):** The PIO allows parallelization of up to 32 bits for both input and output. The PIO IP component provides options for defining device parameters, including data size. In the hardware development, four PIOs were utilized: two PIOs in input mode to receive output data from the RNN and the 'done' signal, with bit widths of 19 and 1 respectively; two PIOs in output mode, sending the read address and enable signals related to the read of the double RAM, with bit widths of 8 and 1 respectively.

- **JTAG UART:** The JTAG target connection utilizes the micro USB cable to control data flow between the FPGA and the computer. This data is packaged using the UART protocol [55], allowing asynchronous serial communication between electronic devices. This approach supports sequential bit-by-bit transmission and reception of data, enabling the transfer of bits without requiring a shared clock signal between the devices.

- **Clock Bridge (clk):** This component synchronizes the clock frequency across all elements of the NIOS II system. To ensure proper functioning of the NIOS II core functions, the RNN clock frequency should be at least four times higher than the JTAG clock frequency [56]. Consequently, a clock frequency of 70 MHz was selected.

- **Reset Bridge (reset):** A global hardware reset signal that promptly forces the processor core to reset.

4.3.1.2 Structure of the NIOS II Software

After creating the NIOS II hardware, a Board Support Package (BSP) file, which contains the address of the components and the IPs of the system, is generated to produce a mapping that serves as an identifier for the software. The NIOS II software architecture, supported by libraries in the C language, enables the creation of custom instructions. The NIOS II processor's data bus is 32 bits wide, and its instructions are available to read and write in one byte, half-word (16-bit), or word (32-bit) data.

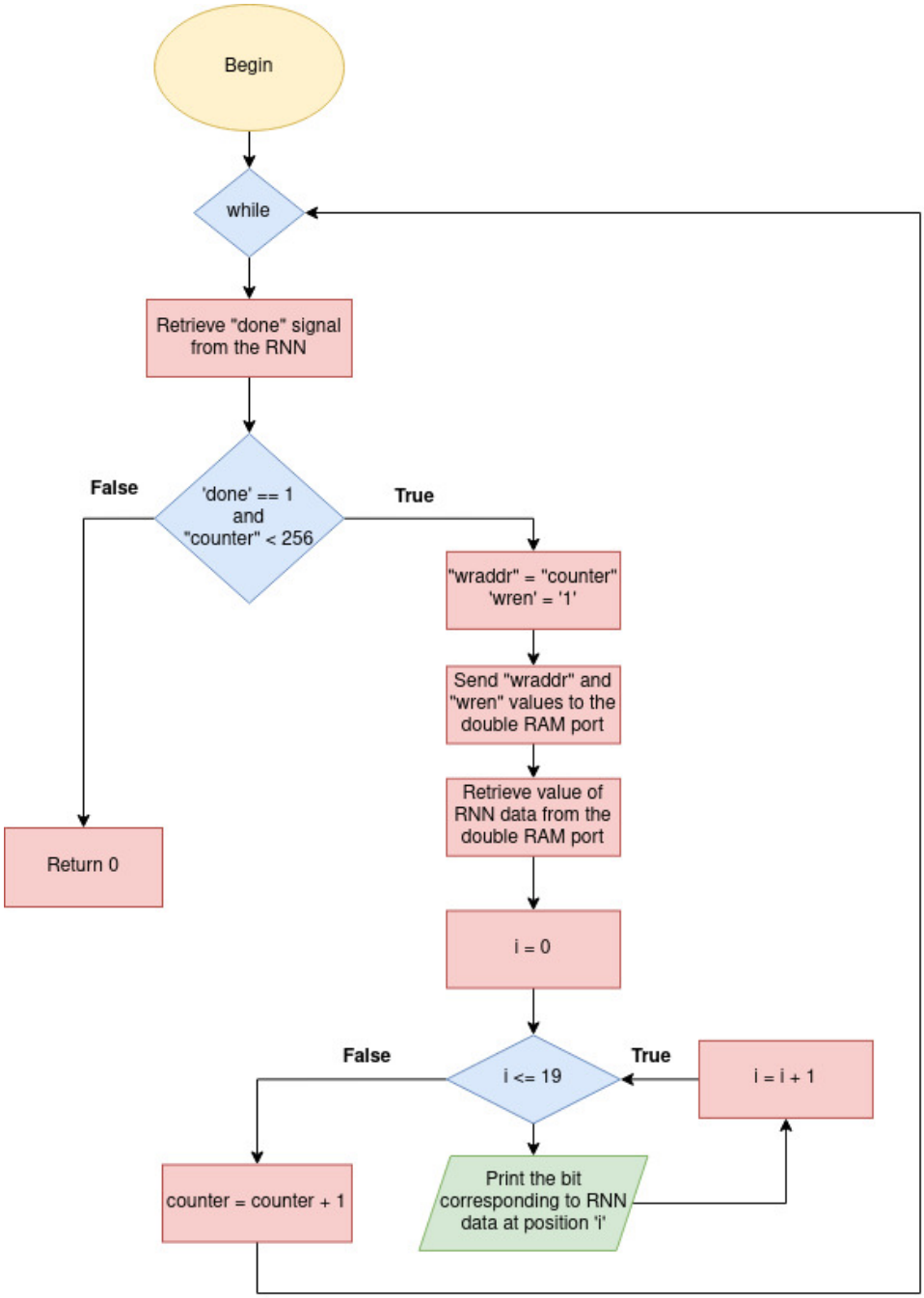


Figure 73 – Representation of the code flow diagram of the C code used to retrieve data from the RNN as well as to control the read enable and read address of the double-port RAM.

The software structure designed for the test firmware project, developed using the C language and using specialized libraries, is depicted in Figure 73. This structure provides the means to read the double-port RAM. Additionally, it transmits the read value through a computer terminal using a micro USB. This transmission permits the visualization of the obtained results.

4.3.2 Double-port RAM

The double-port RAM is a synchronous component that enables the concurrent execution of both read and write operations within the same clock cycle. This design unit seamlessly transitions between read and write operations. In the context of this application, the double-port RAM stores 256 words, each with the capacity of 19 bits. The addressing mechanism is implemented using counters. While the write side is controlled by a VHDL block, the read side is instantiated through the counter within the NIOS software. When combining both counter components in VHDL, data desynchronization was observed. Due to this issue, it was necessary to incorporate the read counter inside the NIOS II. The double-port RAM is filled with output values from the RNN. Upon activating the value retrieval through a terminal, these values are extracted from the double-port RAM.

4.3.3 Counter for the write address of the double-port RAM

This counter plays a crucial role in controlling specific events or states. In the context of test firmware implementation, we utilize an 8-bit counter, enabling it to represent binary values ranging from 0 to 255. This specific counter is engineered to generate the write address for a double-port RAM. Within this scenario, the counter serves as a controller, determining the precise memory address to be written at any given moment. This ensures the preservation and organization of data stored in the double-port RAM without incurring any data loss.

4.3.4 Phase-Locked Loop

A Phase-Locked Loop (PLL) is designed to synchronize and control the output frequency of a signal with respect to a reference signal. A PLL is utilized to generate two distinct frequencies: 560 MHz and 70 MHz, both derived from a single 25 MHz clock crystal reference. This process is achieved through a series of circuits and signal processing that keep the frequencies synchronized and in phase, ensuring no phase difference between them. The use of a PLL prevents potential timing discrepancies that may arise when using two separate clock crystals to generate these frequencies. When two distinct crystals are employed to produce different frequencies, there is a chance that their oscillation characteristics might not be perfectly synchronized. This misalignment could result in slight frequency variations between the crystals, leading to a gradual phase shift over time in the generated clocks. By utilizing a PLL to generate these two frequencies, one ensures a consistent phase relationship and synchronization between them.

4.3.5 Reset release

The role of the reset release component is to verify the successful completion of the Stratix 10 device setup programming process. This component, obtained from an IP from the

Quartus platform, generates a signal that is activated only when it is considered safe to release the reset signal throughout the entire device. Its primary function is to temporarily halt clocks and other interconnected components until this signal is initiated. Including an instance of the "reset release" in the project is imperative to ensure that the programming of the firmware is completed before starting to run the test firmware on the FPGA.

4.3.6 Debounce button

A physical reset button was incorporated to allow physical reset of the firmware. Nevertheless, physical buttons are prone to signal fluctuations caused by rapid and multiple signals occurring in quick succession. These instances can transpire during button presses, releases, or even minor physical disturbances. Such swift signal variations can result in imprecise readings or unintended system actions. To counter this issue effectively, the debounce button assumes a critical role. By implementing a debounce mechanism, this button introduces a delay that guarantees the registration of only steady and intentional signals after a button press or release event. Upon button press, the debounce circuit is engineered to momentarily disregard any additional signals that could emerge due to slight bouncing or button vibrations. Through the application of a 10 µs delay following signal stabilization, authentic inputs to the system are sent.

4.3.7 Full generate

The purpose of the “full generate” component is to ascertain whether the Double Port RAM is at its maximum storage capacity, thereby halting any further RAM write operations upon reaching this limit. This module is equipped with input ports for write and read addresses which are used in its architecture. The “full generate” architecture computes the disparity between the write and read addresses. This calculated difference is then stored within a dedicated signal, which is subsequently compared against the maximum value that the Double Port RAM is capable of storing. When this dedicated signal attains a value equivalent to the Double Port RAM’s maximum capacity, it serves as an indicator that the buffer has reached its full capacity. As a result, the “full” output signal is activated.

4.4 Validation with simulation

Simulation is a crucial step in the design and verification process of digital systems and electronic circuits. It enables the testing and validation of the aforementioned components before their physical implementation in the Stratix 10 Development Kit. Simulation helps identify errors, timing issues, signal conflicts, and other potential problems that could emerge during the system’s actual operation.

The TimeQuest timing analyzer, provided by Intel, was used to ensure that there are no timing violations. This tool is applied during the compilation stage, and the obtained slack was 0.026 ns for a frequency of 560 MHz and 0.115 ns for a frequency of 70 MHz. The validation of the components operation was performed using ModelSim [57], a widely recognized software designed for simulating digital circuits and transistor-level systems. Part of the Electronic Design Automation (EDA) tool family, ModelSim is developed by Mentor from the Siemens company. This software offers a comprehensive simulation platform, accommodating a variety of hardware description languages, including VHDL. ModelSim facilitates the creation of models, enabling simulations ranging from basic logic gates to entire processors and embedded systems. For ModelSim simulation, the use of a testbench is necessary. The testbench is designed to test the behavior of a digital design.

4.4.1 Validation with ModelSim

Simulation was conducted using ModelSim on VHDL components that were not auto-generated via Quartus IP. The objective was to validate the correct functionality of each individual element.

The address write counter's testbench aims to verify the correct generation of the addressing for the double-port RAM write operation. The testbench simulates both the system's reset signal ("rstn_i") and the RAM's write enable signal ("enable_data_o"). Figure 74 illustrates the various signals simulated by ModelSim. The outcome of this test can be observed through the "enable_data_o" signals, which should correspond to the input signal "rstn_i". The output signal contains as expected addresses ranging from 0 ("min_value") to 255 ("max_value"), represented by the "counter_o" signal. After achieving the "max_value","counter_o" returns to 0.

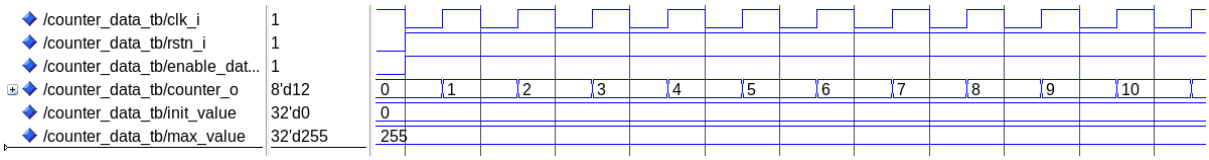


Figure 74 – Simulation in ModelSim of the counter write address component. The signal "clk_i" serves as the component clock, while "rstn_i" is an input signal for the component reset. The "enable_data_o" is an output signal that enables writing to the RAM double-port. The signals "init_value" and "max_value" define the range (initial and final) of addresses for "counter_o". The "counter_o" is responsible for handling write addresses within the double-port RAM and advances by one position during each clock cycle.

In Figure 75, the simulated module in ModelSim shows the stabilization of the button signal using the debounce button component. The test bench simulates the reset signal of the component and the instability of the button, represented in the figure by the signals "reset_n_tb" and "button_tb". The output signal waits for a predetermined time interval after the button

stabilizes, as can be seen in the “result_tb” signal. This ensures that the output reset is triggered only after this duration.

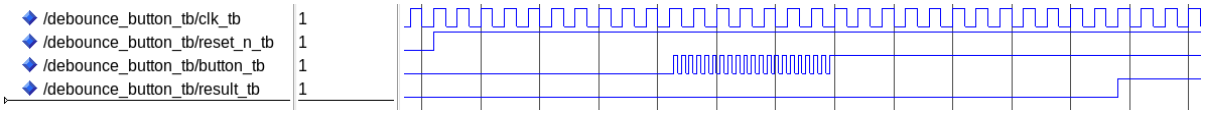


Figure 75 – Simulation in ModelSim of the debounce button component. The signal “clk_tb” serves as the component clock, while “reset_n_tb” is an input signal for the component reset. The “button_tb” represents the instability of the physical button when pressed, and the “result_tb” signal represents the state of the “button_tb” after it has stabilized.

Figure 76 illustrates the simulation of the “full_generate” component. The testbench manages the signals “wraddress_i” and “rdaddress_i”, which correspond to the write and read addresses of the RAM, respectively. The testbench also simulates the component’s reset using the signal “rstn_i”. The simulation outcome generates the output signal “full_o”, which indicates the moment when the RAM with 256 memory locations is completely filled. This signal is obtained by calculating the difference between “wraddress_i” and “rdaddress_i”, and if the difference is 255, the “full_o” signal becomes ‘1’.

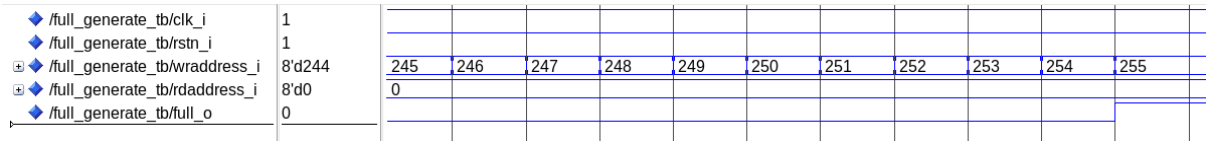


Figure 76 – Simulation in ModelSim of the full generator component. The signal “clk_i” serves as the component clock, while “rstn_i” is an input signal for the component reset. The signals “wraddress_i” and “rdaddress_i” are, respectively, the write and read address of the RAM double port . The signal “full_o”, indicates that the RAM double port is full.

4.5 Validation of the energy extraction firmware on the hardware

The validation of the test firmware was conducted by substituting the neural network’s input with a RAM containing a set of predetermined values, ranging from 0 to 255. Through utilization of the SignalTap tool and the read to the terminal, the resulting output was verified, which allows the validation of these values up to the Nios II level.

4.5.1 Design of the RAM

To verify the proper functioning of the test firmware, a RAM containing 256 values ranging from 0 to 255 was integrated into the system and replaces the NN. Additionally, an address counter was included, which enables the activation of the read enable for this RAM. This counter also holds the address of the position of the value to be transmitted, as depicted in

the schematic diagram of the system in Figure 77. The RAM was configured to inject an event every 560 MHz as, the same frequency of the RNN firmware.

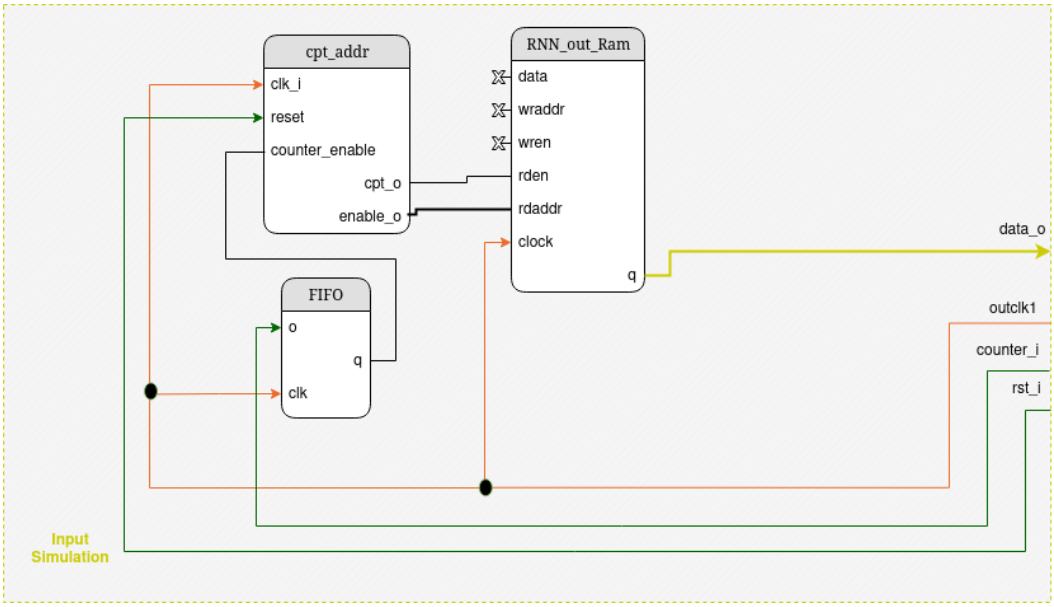


Figure 77 – The schematic depicts the components of the RAM replacing the NN block and containing the values to be read out by the test firmware. It includes: The **RNN_out_Ram**, which allows insertion of values through a Memory Initialization File (.mif). The **counter_addr**, responsible for managing the RAM read operations with precise timing, and the **FIFO**, which ensures signal synchronization between the counter and the data.

Figure 78 depicts a schematic similar to that of Figure 71, with the distinction that the inputs are sourced from the RAM. As explained in section 4.1, a ‘done’ signal is set to ‘1’ upon obtaining the first reconstructed energy. To simulate this ‘done’ signal emitted by the RNN, a FIFO was added to the test firmware, as illustrated in Figure 78.

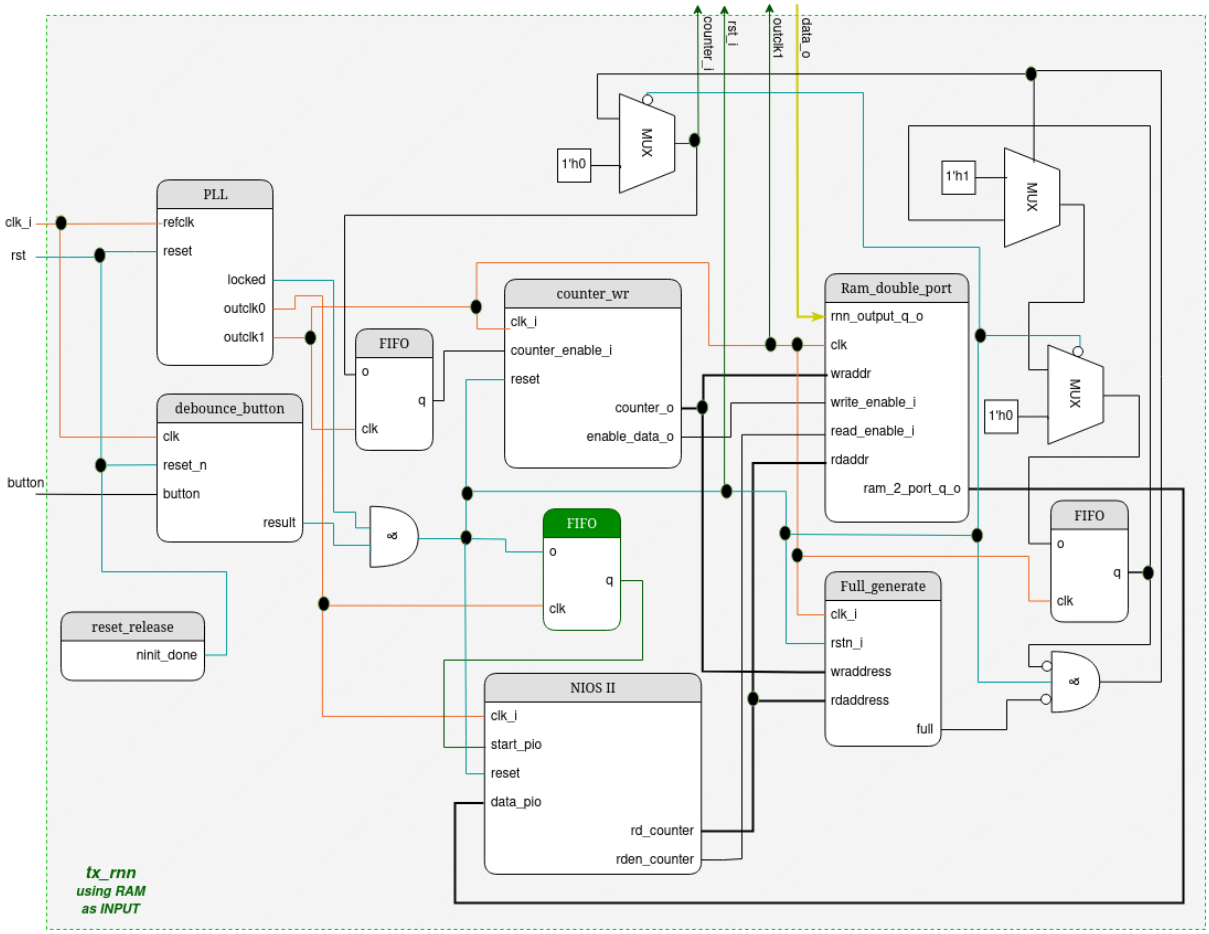


Figure 78 – The schematic depicts the components of the necessary test firmware implementation with a RAM replacing the RNN input. The arrangement closely resembles Figure 71, differing only in the origin of the "data_o" from a RAM, and the replacement of the "done" signal with a FIFO in green.

4.5.2 Validation with SignalTap

SignalTap is a feature incorporated within Intel's Quartus Prime software. Functioning as a potent tool, it facilitates debugging and signal analysis for FPGA-based hardware projects. It allows for viewing internal signals in real time, without the need for physical instrumentation or modifications to the original design. The SignalTap feature empowers one to choose the signals one intends to monitor. The data that is captured can be conveniently showcased through waveform, tables, and various other visualization formats. Moreover, SignalTap offers the functionality to establish triggers. These trigger conditions grants the capability to initiate signal capture solely when particular predefined events transpire. This streamlined approach eases the process of concentrating on events that hold significance, thus enhancing the precision and efficiency of the analysis. Since SignalTap is an effective debugger, it was employed to verify and validate the hardware implementation of the test firmware.

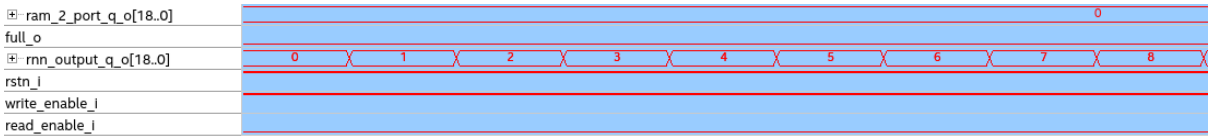


Figure 79 – Real-time validation of the test firmware using SignalTap with zoom in. The frequency utilized for this simulation was 560 MHz for signals "full_o", "rstn_i", "write_enable_i", "rnn_output_q_o" signals and 70 MHz for signals "ram_2_port_q_o", "read_enable_i". The signal "write_enable_i" is enabling the writing while the "read_enable_i" maintaining deactivated the reading of the double-port RAM. The "rnn_output_q_o" is the data come from the "RAM_out_Ram" (Figure 77). The "ram_2_port_q_o" sends the output values from the "RAM_double_port" (Figure 78) to the Nios II processor. The "full_o" signal indicates the moment when this RAM is full, deactivating the "write_enable_i" signal. The "rsrn_i" signal is used to reset the full system.

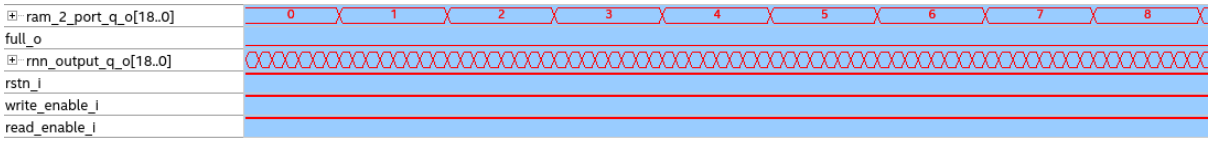


Figure 80 – Real-time validation of the test firmware using SignalTap with zoom out. The frequency utilized for this simulation was 560 MHz for signals "full_o", "rstn_i", "write_enable_i", "rnn_output_q_o" signals and 70 MHz for signals "ram_2_port_q_o", "read_enable_i". The signals "write_enable_i" and "read_enable_i" are enabling the writing and reading of the double-port RAM, respectively. The "rnn_output_q_o" is the data coming from the "RAM_out_Ram" (Figure 77). The "ram_2_port_q_o" sends the output values from the "RAM_double_port" (Figure 78) to the Nios II processor. The "full_o" signal indicates the moment when the RAM is full, deactivating the "write_enable_i" signal. The "rsrn_i" signal is used to reset the full system.

Figure 79 and 80 show several signals with SignalTap. The signals include the output of the NN (*rnn_output_q_o*) before entering the double port RAM ("*RAM_out_Ram*") and after exiting this RAM on its way to the NIOS (*ram_2_port_q_o*), as well as their enable signals for reading and writing. The global system reset signal and the indicator for the completion of the double-port RAM operation are also highlighted. Furthermore, the acquisition process occurs when the "*read_enable_i*" signal is activated.

In Figure 79 and 80, the input values follow the sequential arrangement defined in a 256-position MIF file originating from the "*RAM_out_Ram*", which replaces the NN as explainned in section 4.5.1. The input and output values of the dual-port RAM operate at frequencies of 560 MHz and 70 MHz, respectively. Both figures represents the same simulation with different timing and zooms. The figures provides visual representations of how the "*write_enable_i*" and "*read_enable_i*" signals control the operation of the double-port RAM. In Figure 79, the "*write_enable_i*" signal enables the writing process, while the "*read_enable_i*" signal disables the reading process of the double-port RAM. On the other hand, Figure 80 illustrates a scenario where both the "*write_enable_i*" and "*read_enable_i*" signals are active, allowing both writing and reading operations to take place simultaneously in the double-port RAM. In both figures,

it is also possible to verify that no data was ignored or modified during the simulation, thus demonstrating the correct functionality of the firmware.

In figure 80 is also possible to verify the frequency difference between the input and output signals. The values of the "*ram_2_port_q_o*" signal are sampled every 8 clock cycles, whereas "*rnn_output_q_o*" is sampled every single clock cycle. This demonstrates proper operation at the desired frequency.

4.6 Extraction of NN values

To perform the extraction of values from the RNN, as discussed in Section 4.1, the integration of the RNN firmware with the test firmware module was carried out. The RNN firmware has 259 samples as input values obtained from AREUS simulation, which, given this data set's size, generate 256 transverse reconstructed energy. These specific values were chosen to align with the storage capacity of the dual-port RAM.

The process of programming the firmware onto the FPGA was conducted through the physical connection established between the FPGA and the computer, using a micro USB cable. This physical connection not only enabled programming but also served as a channel for data traffic, essential for data retrieval later. Once the FPGA is programmed, the reset button is pressed to initiate the energy reconstruction process. When the first value of the reconstructed transverse energy is obtained, the writing of values to the double-port RAM starts. The writing to the double-port RAM is performed every 14 clock cycles at a frequency of 560 MHz to be able to obtain an NN computed energy corresponding to one channel with a frequency of 40 MHz each.

Reading the RAM using the Nios begins by loading the developed Nios II software onto the Nios II processor. Once this step is completed, then it's possible to open the Nios II terminal. The processor is instructed to immediately start executing the program previously loaded onto the Nios II processor right after opening the terminal session, allowing the terminal output to be recorded in a log file while being displayed on the terminal simultaneously. This process needed to be repeated 8 times in order to obtain 2048 values.

4.7 Comparison between extracted values from hardware and simulation

The transverse energy reconstruction achieved using the Stratix 10 development kit hardware was first compared bit by bit to firmware simulation and they match perfectly. Then the values were compared to the obtained ones from the Keras implementation. A small discrepancy is expected due to fixed point usage and truncation in the NN firmware as explained in [2].

Figures 81 and 82 shows energy difference between the RNN firmware and Keras. Figure 81, encompassing all collected energies, exhibits a Root Mean Square (RMS) of 0.55 MeV, with a mean of -1.2 MeV. In contrast, Figure 82 focuses exclusively on energies surpassing 240 MeV which corresponds to the 3σ noise level. This threshold facilitates noise-free energy analysis, resulting in an RMS of 0.50 MeV and a mean of -1.3 MeV. Both graphs presents an RMS which is smaller than the noise level of 80 MeV. Figure 81 exhibits a peak at zero. This phenomenon arises when the reconstructed transverse energy is zero in both Keras and the firmware.

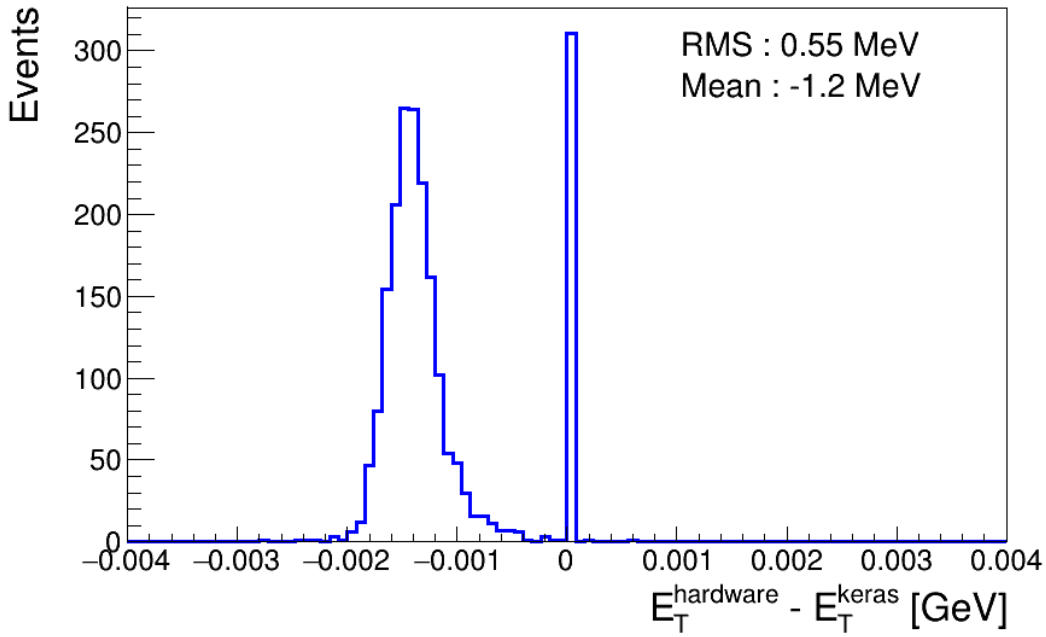


Figure 81 – Difference between the transverse energy computed with the firmware and the one computed with Keras.

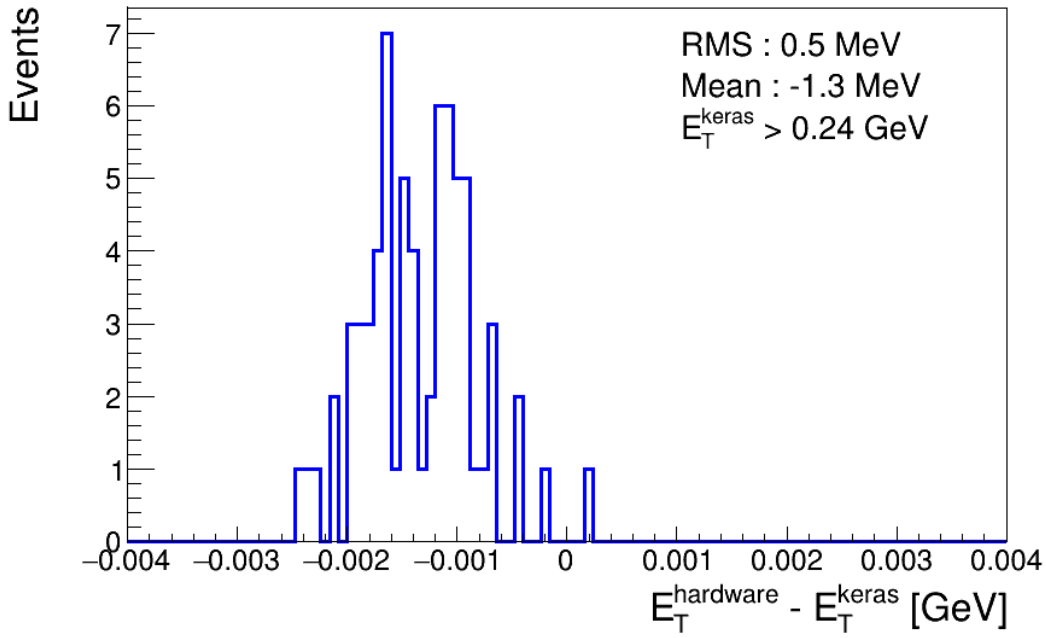


Figure 82 – Difference between the transverse energy computed with the firmware and the one computed with Keras. A cut on the Keras energy of 240 MeV is performed to select values 3σ above the expected electronic noise level in the calorimeter.

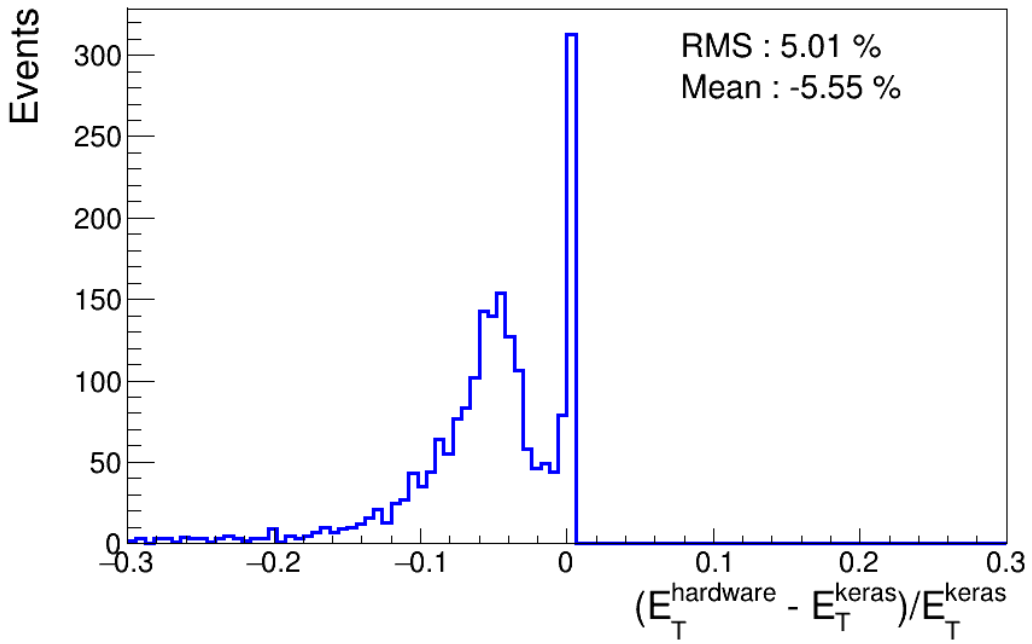


Figure 83 – Relative difference between the transverse energy computed with the firmware and the one computed with Keras.

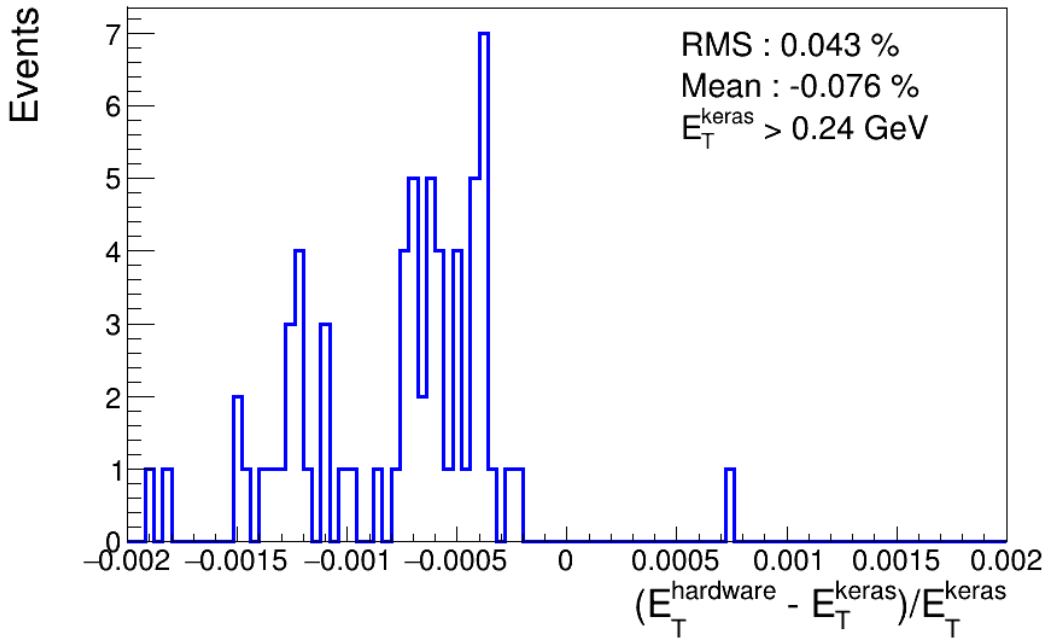


Figure 84 – Relative difference between the transverse energy computed with the firmware and the one computed with Keras. A cut on the Keras energy of 240 MeV is performed to select values 3σ above the expected electronic noise level in the calorimeter.

Figures 83 and 84 illustrate the relative energy difference between the RNN firmware and Keras, relative to the Keras energy. Figure 83, considering all recorded energies, presents an RMS of 5.01 % and a mean of -5.55 %. Meanwhile, Figure 84 selects energies exceeding 240 MeV, ensuring a clean energy analysis, and yields an RMS of 0.043 % along with a mean of -0.076 %. These results reproduce the expected firmware resolution of less the 0.1 % with the 3σ cut from the simulation shown in reference [3].

The deviation of the mean from 0, evident in the four plots, stem from the utilization of the truncation on 19 bits in the internal calculation. Furthermore, the attained resolution is impacted by loading the converted values (floating to fixed point representations) for both input and weight values.

4.8 Conclusion

A test firmware has been created to verify the functionality of the NN implementation on the hardware platform. This test firmware is versatile and can test different types of NNs. It provides the input values to the NN and extracts the calculated energy values. The NN firmware has been validated and the difference with Keras was found to be less than 0.1% for energies above the noise threshold, which perfectly agrees with expected values from firmware simulation.

It's worth noting that the validation of the test firmware was carried out using a restricted

array of values. Each execution is restricted to extracting 256 values. AN important additional check is needed to validate firmware stability over long periods of the time at 40 MHz. This can be done by directly checking the computed energies in the FPGA by comparing it to expected values stored in a RAM. Both the NN and test firmware should also be ported to the Agilex FPGAs before being able to test them on the final LASP board.

CHAPTER

5

CONCLUSION AND FUTURE PERSPECTIVES

The high luminosity of the HL-LHC will lead to collisions with high pileup resulting in a reduction in the accuracy of the energy reconstruction in the ATLAS LAr calorimeter when utilizing the optimal filtering algorithm. To fulfill the goals of the ATLAS experiment, the development of innovative energy reconstruction methods is imperative. The Phase II upgrade, scheduled for 2026-2028, involves replacing the LAr calorimeter readout system. This upgrade entails the replacement of the LAr readout electronics with cutting-edge FPGAs, providing more processing capabilities and allowing the implementation of neural networks for the energy reconstruction. These energy reconstruction algorithms are set to be integrated into the Phase II LASP boards, each containing two FPGAs.

Vanilla-RNN and LSTM architectures have been developed to reconstruct the energy deposited in LAr cells and are shown to exhibit enhanced performance compared to the optimal filtering algorithm within the demanding circumstances of the HL-LHC. RNNs were crafted to precisely reconstruct the energy of overlapping pulses, addressing a domain where the optimal filtering algorithm shows vulnerabilities.

This thesis presents the implementation of the developed neural network in firmware using the HLS language and its implementation in the HLS4ML library. Furthermore, it also describes the design of a test firmware, that was used to validate the neural network implementation in the hardware.

The HLS4ML library translates neural networks developed with common machine learning tools (such as Keras) to HLS language that can be used to generate firmware. It streamlines parameter calibration and optimization for neural networks to suit FPGA frameworks, thereby expediting the prototyping phase. However the HLS4ML package did not support RNN architectures and INTEL FPGAs. I implemented the Vanilla-RNN and LSTM in HLS4ML in a way that aimed at curbing resource usage on FPGAs while upholding performance standards. This implementation targeted INTEL FPGAs with Quartus HLS. I used this implementation to

scan the neural network parameters and find the optimal settings that can fit within the FPGA resources. However the HLS, and thus HLS4ML, was found to lack the flexibility to generate a firmware that fits the LASP specification. Thus a firmware in VHDL is developed to further optimize the HLS firmware and to meet the specifications. Although the development in HLS where instrumental for fast optimization during the prototyping phase.

To validate the RNN firmware on FPGAs, I created and deployed a test firmware targeting the Intel Stratix 10 development kit. This test firmware provides the input values and the weights to the RNN firmware. It also allows the extraction of the energy reconstructed by the RNN firmware by the means of a system on chip (NIOS II) and a JTAG connection. The test firmware was used to validate the NN firmware and the extracted value from the FPGA were found to match the expected values from simulation.

In conclusion, RNNs emerge as promising contenders for the endeavor of energy reconstruction within the LAr calorimeter throughout the HL-LHC era. They outperform the optimal filtering algorithm and can be seamlessly integrated into FPGA devices. The firmware of the Stratix10 FPGA should be adapted to the Agilex FPGA, which will serve as the final FPGA for the LASP board. Automating the process of generating VHDL code could significantly simplify the firmware developed process. Further enhancements in HLS should also be investigated since the language is recent and in rapid expansion.

BIBLIOGRAPHY

[1] H. Abreu, M. Aharrouche, M. Aleksa, L. Bella, J. Archambault, S. Arfaoui, O. Arnaez, E. Auge, M. Aurousseau, S. Bahinipati, J. Bán, D. Banfi, A. Barajas, T. Barillari, A. Bazan, F. Bellachia, O. Beloborodova, D. Benchekroun, K. Benslama, et al., *Performance of the electronic readout of the ATLAS liquid argon calorimeters*, Journal of Instrumentation **5** (2010) , https://dx.doi.org/10.1088/1748-0221/5/09/P09003. Citations on pages 22 and 66.

[2] G. Aad, T. Calvet, N. Chiedde, R. Faure, E. Fortin, L. Laatu, E. Monnier, and N. Sur, *Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment*, Journal of Instrumentation **18** (2023) P05017, https://dx.doi.org/10.1088/1748-0221/18/05/P05017. Citations on pages 23, 67, 98, 99, 100, 101, and 116.

[3] G. Aad, A.-S. Berthold, T. P. Calvet, N. Chiedde, E. Fortin, N. Fritzsche, R. G. Hentges, L. A. O. Laatu, E. Monnier, A. Straessner, and J. C. Voigt, *Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters*, tech. rep., CERN, Geneva, Jul, 2021. https://cds.cern.ch/record/2775033. Citations on pages 24, 25, 69, 78, 79, 80, 81, 82, and 119.

[4] S. L. Glashow, *Partial-symmetries of weak interactions*, Nuclear Physics **22** (1961) 579–588, https://www.sciencedirect.com/science/article/pii/0029558261904692. Citation on page 33.

[5] A. Salam and J. Ward, *Electromagnetic and weak interactions*, Physics Letters **13** (1964) 168–171, https://www.sciencedirect.com/science/article/pii/0031916364907115. Citation on page 33.

[6] S. Weinberg, *A Model of Leptons*, Phys. Rev. Lett. **19** (1967) 1264–1266, https://link.aps.org/doi/10.1103/PhysRevLett.19.1264. Citation on page 33.

[7] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. Abdel Khalek, et al., *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B **716** (2012) 1–29,

https://www.sciencedirect.com/science/article/pii/S037026931200857X. Citation on page 33.

[8] I. J. R. Aitchison and A. J. Hey, *Gauge Theories in Particle Physics: A Practical Introduction, Non-Abelian Gauge Theories: QCD and The Electroweak Theory*, vol. 2. CRC Press, 2013. https://library.oapen.org/handle/20.500.12657/50884. Citation on page 35.

[9] M. J. Shroff, *A data injector for the High Luminosity LHC ATLAS Liquid Argon Signal Processor*, https://cds.cern.ch/record/2753823, Presented 18 Aug 2020. Citation on page 35.

[10] N. Kumari, *Unravelling the top-Higgs coupling with the ATLAS experiment at LHC*, https://cds.cern.ch/record/2846001, Presented 16 Nov 2022. Citation on page 35.

[11] P. D. Group, *Review of Particle Physics*, Physical Review D **98** (2018) 030001, https://journals.aps.org/prd/abstract/10.1103/PhysRevD.98.030001. Citation on page 35.

[12] D. J. Griffiths, *Introduction to Elementary Particles*. John Wiley & Sons, Nova Iorque, 1999. http://nuclphys.sinp.msu.ru/books/b/Griffiths.pdf. Citations on pages 36, 37, and 38.

[13] C. N. Yang and R. L. Mills, *Conservation of Isotopic Spin and Isotopic Gauge Invariance*, Phys. Rev. **96** (1954) 191–195, https://link.aps.org/doi/10.1103/PhysRev.96.191. Citation on page 36.

[14] Y. Kosmann-Schwarzbach, *The Noether Theorems: Invariance and Conservation Laws in the Twentieth Century*. Springer, 2011. Citation on page 38.

[15] F. Englert and R. Brout, *Broken Symmetry and the Mass of Gauge Vector Mesons*, Phys. Rev. Lett. **13** (1964) 321–323, https://link.aps.org/doi/10.1103/PhysRevLett.13.321. Citation on page 39.

[16] L. Del Debbio, *Parton Distributions in the LHC Era*, EPJ Web Conf. **175** (2018) 01006. Citation on page 40.

[17] N. K. Vu, *Search for SUSY Electroweak production at LHC Run 2 and study of CMOS sensor for ITk replacement in second-half of HL-LHC*, https://amu.hal.science/tel-03551191. Citation on page 43.

[18] E. Mobs, *The CERN accelerator complex - 2019. Complexe des accélérateurs du CERN - 2019*, General Photo. Citation on page 43.

[19] ATLAS Collaboration, *Luminosity Plots from the October 2018 Public Results (Run 2)*, n.d. https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2#Luminosity_Plots_from_the_Octobe. Accessed on: Juin 2023. Citation on page 45.

[20] CERN AC, *Schema du detecteur ATLAS, une experience proposee pour le LHC. A noter la taille des personnes autour du detecteur*, General photo, Mar., 1998. Citation on page 47.

[21] J. Pequenao, *Computer generated image of the ATLAS inner detector*, General photo, Mar., 2008. Citation on page 49.

[22] ATLAS Collaboration, *Experiment Briefing: Keeping the ATLAS Inner Detector in perfect alignment*, General photo, July, 2020. Citation on page 49.

[23] A. Collaboration, *Approved Plots of the Tracking Combined Performance Group*, http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/IDTR-2015-007. Citation on page 52.

[24] A. Collaboration, *Track and Vertex Reconstruction in the ATLAS Inner Detector*, Tech. Rep. ATL-PHYS-PUB-2022-033, CERN, 2022. https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2022-033/. Citation on page 52.

[25] ATLAS Collaboration, G. Aad et al., *Operation and performance of the ATLAS semiconductor tracker in LHC Run 2*, JINST **17** (2022) P01013, arXiv:2109.02591, https://cds.cern.ch/record/2780336. Citation on page 54.

[26] A. Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider: A Description of the Detector Configuration for Run 3*, 2023. Citation on page 54.

[27] ATLAS Collaboration, *ATLAS Public Results*, https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TRTPublicResults. Accessed on: Juin 2023. Citation on page 55.

[28] J. Pequenao, *Computer generated image of the ATLAS Liquid Argon*, https://cds.cern.ch/record/1095928. Citation on page 56.

[29] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3** (2008) S08003, https://cds.cern.ch/record/1129811. Citations on pages 57 and 59.

[30] J. Pequenao, *Computer generated image of the ATLAS Muons subsystem*, Cern-ge-0803017, Mar., 2008. CDS: https://cds.cern.ch/record/1095929. Citation on page 60.

[31] CERN, *ATLAS Experiment Data at CERN*, https://cds.cern.ch/record/2872309, 2023. Citation on page 62.

[32] A. Collaboration, *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System*, tech. rep., CERN, Geneva, 2017. https://cds.cern.ch/record/2285584. Citation on page 63.

[33] ATLAS, *ATLAS Liquid Argon Calorimeter Phase-II Upgrade: Technical Design Report*, https://cds.cern.ch/record/2285582. Citations on pages 64 and 70.

[34] W. Cleland and E. Stern, *Signal processing considerations for liquid ionization calorimeters in a high rate environment*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **338** (1994) 467–497, https://www.sciencedirect.com/science/article/pii/0168900294913323. Citation on page 66.

[35] Madysa, Nico, *AREUS: A Software Framework for ATLAS Readout Electronics Upgrade Simulation*, EPJ Web Conf. **214** (2019) 02006, https://doi.org/10.1051/epjconf/201921402006. Citation on page 67.

[36] *Development of readout and trigger electronics for the ATLAS Liquid Argon Calorimeters*, https://tu-dresden.de/mn/physik/iktp/arbeitsgruppen/experimentelle-teilchenphysik/forschung/detektorentwicklung?set_language=en, n.d. Accessed: April 29, 2023. Citation on page 68.

[37] W. S. McCulloch and W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, The Bulletin of Mathematical Biophysics **5** (1943) 115–133. Citation on page 71.

[38] P. J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, USA, 1994. Citation on page 71.

[39] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998) 2278–2324. Citation on page 71.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research **15** (2014) 1929–1958, http://jmlr.org/papers/v15/srivastava14a.html. Citation on page 71.

[41] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, *Spiking Neural Networks and Their Applications: A Review*, Brain Sciences **12** (2022), https://www.mdpi.com/2076-3425/12/7/863. Citation on page 71.

[42] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., USA, 1994. Citation on page 72.

[43] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org. Citations on pages 73 and 80.

[44] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv:1609.04747 [cs.LG]. Citation on page 74.

[45] R. Pascanu, T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks*, vol. 28 of *Proceedings of Machine Learning Research*. PMLR, Atlanta, Georgia, USA, 2013. https://proceedings.mlr.press/v28/pascanu13.html. Citation on page 74.

[46] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, Journal of Machine Learning Research - Proceedings Track **9** (2010) 249–256. Citation on page 74.

[47] X. Glorot, A. Bordes, and Y. Bengio, *Deep Sparse Rectifier Neural Networks*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics **15** (2011) 315–323, https://proceedings.mlr.press/v15/glorot11a.html. Citation on page 74.

[48] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997) 1735–1780. Citation on page 76.

[49] F. Chollet, *Keras*, https://keras.io/. Citation on page 78.

[50] TensorFlow, *TensorFlow*, https://www.tensorflow.org/. Citation on page 78.

[51] L. A. O. Laatu, *Development of artificial intelligence algorithms adapted to big data processing in embedded (FPGAs) trigger and data acquisition systems at the LHC*, https://cds.cern.ch/record/2875588, Presented 03 Oct 2023. Citation on page 80.

[52] N. Chiedde, *Implementation and optimizations linked to Simple-RNN and LSTM for Quartus*, https://github.com/fastmachinelearning/hls4ml/pull/575, GitHub. Citation on page 85.

[53] Intel Corporation, *Nios II Classic Processor Reference Guide*, https://www.intel.com/content/www/us/en/docs/programmable/683620/current/nios-ii-f-core-30732.html. Citation on page 105.

[54] Intel Corporation, *Qsys System Design Tutorial*, https://www.intel.com/content/www/us/en/docs/programmable/683378/current/qsys-system-design-tutorial.html. Citation on page 106.

[55] Intel Corporation, *Embedded Peripherals IP User Guide, JTAG UART Core*, https://www.intel.com/content/www/us/en/docs/programmable/683130/21-4/jtag-uart-core.html. Citation on page 107.

[56] Intel Corporation, *Nios II Processor Reference Guide*, https://www.intel.com/content/www/us/en/docs/programmable/683836/current/introduction.html. Citation on page 107.

[57] Intel Corporation, *ModelSim Intel FPGAs Standard Edition Software*, https://www.intel.com/content/www/us/en/software-kit/750368/modelsim-intel-fpgas-standard-edition-software-version-18-1.html. Citation on page 111.