Real-Time Trigger and online Data Reduction based on Machine Learning Methods for Particle Detector Technology

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS (Dr.-Ing.)

von der KIT-Fakultät für
Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Inform. Steffen Bähr

geb. in Kaptshagai

Tag der mündlichen Prüfung:
16.07.2020

Hauptreferent: Prof. Dr.-Ing. Dr. h. c. Jürgen Becker
Korreferent: Prof. Dr. Ivan Peric

Real-Time Trigger and online Data Reduction based on Machine Learning Methods for Particle Detector Technology

1. Auflage: Juli 2021

Thank you to all the people that spent slices of their time with me throughout my life
Nothing is more precious to me

Abstract

Modern particle accelerator experiments are producing immense amounts of data online during their operation. Storing the entire amount of generated data is quickly exceeding reasonable budgets for the data readout infrastructure. This problem is traditionally addressed by using a combination of trigger and data reduction mechanisms that are located close to the respective detectors to facilitate a reduction of the data rates as early in the process as possible. Meanwhile, traditional approaches to these systems are struggling with achieving an efficient reduction for modern experiments such as Belle II. The reason for this lies in the complex observed distributions of background, or unwanted, events. This situation is enhanced by the unknown characteristics of both the accelerator and detector before reaching high luminosity operation. A robust and flexible algorithmic alternative is thus required to address this problem. This can be provided by using an approach based on machine learning. Since such trigger and data reduction systems are operated under tight constraints such as small latency budgets, a high number of required data transmission connections and general real-time processing, Field Programmable Gate Array (FPGA)s are used as a technological basis for the implementation. Within this thesis, several approaches based on machine learning methods were developed for FPGAs to fulfil the challenges present at the Belle II experiment. These systems are presented and discussed throughout this thesis.

The primary application case addressed within this thesis represents the suppression of particle tracks that originate from outside of the interaction point at which particles collide. This suppression is performed by estimating a particle track's three-dimensional parameters in real-time. The system that was developed for this task is the neural z-Vertex trigger. It has to estimate the requested parameters within a hard latency budget of $5\mu s$ that is allocated to the entire first level trigger system. This latency has to be achieved while achieving a reasonable accuracy for estimating the parameters to avoid any loss of important physics data. The algorithmic basis of this trigger system is a multi-layer perceptron coupled with a detector-specific preprocessing that reduces the overall processing complexity. For this, a flexible architecture was developed, which is realizing all of the required functional aspects. The architecture is accompanied by a design flow that provides a semi-automated generation of the firmware that facilitates quick updates of the internal processing in case of changing conditions within the experiment. Validation is promoted at several stages, which includes a slow control infrastructure that enables online monitoring of the current status, while data quality monitoring is used to evaluate the system's performance during operation. All aspects are culminating in the generation of two system setups that are used for the experiment's operation. While the former is allowing an early evaluation, without the presence of the complete data readout infrastructure, the later setup is used during physics operation and estimating the desired track parameters. Especially the later system shows that all requirements are fulfilled for both integration and quality of the estimation. This achievement is underlined by the presen-

tation of data samples taken from the experiment's early runs. These data samples show the general correctness of the approach and the ability of the employed neural network to estimate the z-Vertex. In conclusion, this work represents the first z-Vertex estimating trigger system based on neural networks that is operational at a modern state-of-the-art experiment.

Further optimizations of the trigger's performance are achieved by addressing the surrounding sub-systems within the first level trigger system. These optimizations are supplementing the neural z-Vertex trigger but are on their own significantly improving the general trigger performance. Two approaches are presented, which were transitioned to operation on an FPGA within this thesis. These approaches are a revised track segment finder for the experiment's central drift chamber and a Hough-based 3D-Track estimator using a weighting scheme based on Bayes' theorem. The former is addressed by a state machine-based revision of the original track segment finder. The revised system is capable of increasing the overall efficiency substantially, as is shown by data taken from the experiment. It is already operational within the experiment, and qualitative statements are made using collision data. The Hough-based estimator, on the other hand, is currently in the prototyping stage of its development. The system's feasibility for future application is shown for this stage, with the final integration being planned after the installation of anticipated upgraded FPGA platforms. For this, general integration strategies, together with their individual trade-offs, are explored and discussed.

Finally, stepping away from trigger systems, the task of online data reduction is addressed with a discussion about the development of an online cluster analysis. This system is capable of classifying observed particles by only using the data generated at the pixel detector of Belle II. The classification is performed by an FPGA-based implementation of the NeuroBayes algorithm, which thus far was only used on traditional computer systems. The developed system is capable of achieving high efficiency for the classification of pixel detector data while fulfilling the strict real-time requirements set for both throughput and latency. Exploring the boundaries of the system's reachable performance, it is further extended in into the PCIe-based high-throughput demonstrator, for which it is shown that the bottleneck is due to the limitations of data transmission and not caused by the internal processing architecture.

Zusammenfassung

Moderne Teilchenbeschleuniger-Experimente generieren während zur Laufzeit immense Datenmengen. Die gesamte erzeugte Datenmenge abzuspeichern, überschreitet hierbei schnell das verfügbare Budget für die Infrastruktur zur Datenauslese. Dieses Problem wird üblicherweise durch eine Kombination von Trigger- und Datenreduktionsmechanismen adressiert. Beide Mechanismen werden dabei so nahe wie möglich an den Detektoren platziert um die gewünschte Reduktion der ausgehenden Datenraten so frühzeitig wie möglich zu ermöglichen. In solchen Systeme traditionell genutzte Verfahren haben währenddessen ihre Mühe damit eine effiziente Reduktion in modernen Experimenten zu erzielen. Die Gründe dafür liegen zum Teil in den komplexen Verteilungen der auftretenden Untergrund Ereignissen. Diese Situation wird bei der Entwicklung der Detektorauslese durch die vorab unbekannten Eigenschaften des Beschleunigers und Detektors während des Betriebs unter hoher Luminosität verstärkt. Aus diesem Grund wird eine robuste und flexible algorithmische Alternative benötigt, welche von Verfahren aus dem maschinellen Lernen bereitgestellt werden kann. Da solche Trigger- und Datenreduktion-Systeme unter erschwerten Bedingungen wie engem Latenz-Budget, einer großen Anzahl zu nutzender Verbindungen zur Datenübertragung und allgemeinen Echtzeitanforderungen betrieben werden müssen, werden oft FPGAs als technologische Basis für die Umsetzung genutzt. Innerhalb dieser Arbeit wurden mehrere Ansätze auf Basis von FPGAs entwickelt und umgesetzt, welche die vorherrschenden Problemstellungen für das Belle II Experiment adressieren. Diese Ansätze werden über diese Arbeit hinweg vorgestellt und diskutiert werden.

Der primäre Anwendungsfall, der in dieser Arbeit adressiert wird, ist die Unterdrückung von Teilchenspuren, welche ihren Ursprung außerhalb des Kollisionspunkts desExperimentes haben. Diese Unterdrückung wird anhand einer Schätzung der dreidimensionalen Spurparameter zur Laufzeit anhand von Echtzeit-Datenverarbeitung durchgeführt. Diese Aufgabe wird von dem neuronalen z-Vertex Trigger übernommen. Dieses System muss die Schätzung der benötigten Spurparameter innerhalb des harten Latenzbudgets von $5\mu s$ berechnen, welches für das gesamte Triggersystem der ersten Stufe angesetzt ist. Diese Latenz muss eingehalten werden während eine ausreichend gute Genauigkeit für die Schätzung der Parameter erzielt werden muss. Die algorithmische Basis für dieses Triggersystem ist ein mehrschichtiges Perzeptron zusammen mit einer auf das Experiment ausgelegten Vorverarbeitung welche die interne Komplexität des Perzeptrons verringert. Hierfür wurde eine flexible Architektur entwickelt, die alle benötigten funktionalen Aspekte umsetzt. Die entwickelte Architektur wird von einem Entwurfsfluss begleitet, welcher eine semi-automatisierte Generierung der Firmware erlaubt, um schnelle Aktualisierungen der internen Verarbeitung, gemäß der sich ändernden Bedingungen unter denen das Experiment stattfindet, zu erlauben. Die Validierung der Architektur wird durch mehrere Mechanismen unterstützt. Diese bestehen aus der Unterstützung der Slow Control, zum Online-Monitoring des aktuellen Zustands der Hardware, und des

Data Quality Monitorings, welches zur Evaluierung der Vorhersage-Fähigkeit des Triggers zur Laufzeit genutzt wird. Alle betrachteten Aspekte werden schließlich kombiniert um zwei Systeme aufzusetzen, welche während des Betriebs des Experimentes verwendet wurden. Während eines der Systeme es erlaubt den Trigger in einem frühen Entwicklungszustand des Experimentes einzusetzen, ohne das Vorhandensein der kompletten Infrastruktur der Datenauslese, kann das andere System zum Einsatz mit Teilchenkollisionen verwendet werden. Anhand des Systems zum Einsatz während Kollisionen wird gezeigt das alle an den Trigger gesetzten Anforderungen erfüllt werden. Dies wird durch eine Evaluierung der Leistungsfähigkeit anhand von aus dem Experiment genommenen Daten unterstützt. Die gezeigten Daten zeigen hierbei die allgemeine Korrektheit des Triggeransatzes und dessen Fähigkeit ,unter Nutzung neuronaler Netze, den z-Vertex einer Teilchenspur zu schätzen. Zusammenfassend repräsentiert diese Arbeit die Beschreibung des ersten Triggersystems auf Basis von FPGAs welches den z-Vertex mit Hilfe neuronaler Netze während des Betriebs des Experiments hinreichend genau schätzen kann.

Weitere Optimierungen der Vorhersagefähigkeit des Triggersystems werden, durch zusätzliche Ansätze zur Überarbeitung der weiteren Teil-Triggersysteme der ersten Stufe, eingeführt. Diese werden dabei den zuvor vorgestellten z-Vertex Trigger unterstützen. Zwei Ansätze werden dabei diskutiert, welche im Rahmen dieser Arbeit auf FPGAs umgesetzt wurden. Die umgesetzten Ansätze sind zum einen ein überarbeiteter Spursegment-Finder und eine 3D Spurparameter-Vorverarbeitung basierend auf der Hough Transformation in Kombination mit einer Gewichtung basierend auf dem Theorem von Bayes. Der Spursegment-Finder basiert dabei auf einer Umstrukturierung der Architektur unter der Nutzung von Zustandsautomaten. Dieses System ist in der Lage die Leistungsfähigkeit der ersten Vorverarbeitungsstufe der Daten der Driftkammer zu erhöhen. Das System wird anhand von Daten aus dem Experiment untersucht, wobei generell höhere Effizienzen zur Erkennung von Teilchenspuren erzielt werden. Die zusätzliche 3D Vorverarbeitung ist währenddessen in einer prototypischen Entwicklungsphase. Dessen Eignung zum zukünftigen Einsatz im Experiment wird innerhalb dieser Arbeit gezeigt und durch einen Plan zur Integration in das Experiment unterstützt.

Der letzte Abschnitt dieser Arbeit adressiert die Online-Datenreduktion des Experiments durch die Umsetzung einer Online-Clusteranalyse auf FPGAs. Dieses System ist in der Lage beobachteten Teilchen in Untergrund Ereignisse oder gesuchte Pionen zu klassifizieren, in dem nur Daten des Pixeldetektors verwendet werden. Dies wird erzielt in dem der NeuroBayes Algorithmus auf FPGAs umgesetzt wird. Dieser Algorithmus wurde bisher nur auf traditionellen Rechensystemen verwendet. Das hier entwickelte System ist in der Lage eine hohe Effizienz bei der Klassifizierung zu erzielen, während alle Echtzeitanforderungen an den Durchsatz und die Latenz erfüllt werden. Die Leistungsfähigkeit der entwickelten Architektur wird zusätzlich untersucht in dem ein PCIe-basierter Demonstrator vorgestellt wird bei dem der Flaschenhals nicht bei der Architektur, sondern an der verfügbaren Bandbreite ist.

Preface

Diese Dissertation beschreibt den Ausflug eines Ingenieuren bzw. Informatikers in die Welt der Teilchenphysik. Dementsprechend ist nicht die dahinter liegende Physik im Mittelpunkt sondern die Umsetzung von FPGA-basierten Systemen zum Einsatz innerhalb der Datenauslese-Umgebung eines der größten solcher Experimente. Da solche Umgebungen in der Regel massive Ausmaße annehmen, verweise ich begleitend zu diesem Werk auf die diverse Literatur der Kolloboration des Experiments. Während meines eigenen Studiums des Experiments, war ich durchweg beeindruckt von der Vielfältigkeit der innerhalb von Belle II verfolgten Arbeiten. Für jemanden der die entsprechenden Personen tatsächlich trifft und mit ihnen darüber diskutieren kann, ist es natürlich noch einfacher sich in die Thematiken einzufinden, jedoch sind die verschiedenen Werke auch für sich bereits mehr als lohnenswert. Ausserdem waren das mit die faszinierendsten Menschen die ich getroffen habe, es lohnt sich herauszufinden was sie zu sagen bzw. schreiben haben.

Zusätzlich möchte ich mich noch einmal bei allen Personen bedanken, die mir ihre Zeit geschenkt haben. Eltern, Bruder und Familie, die mir immer geholfen und untersützt haben, meinen Quatsch auch schon am längsten mitmachen mussten, ich hoffe sie halten noch mehrere weitere Jahre aus. All meine Freunde aus Mannheimer Schulzeiten Andreas, Dennis, Michael, Timo .. die auch meine ältesten Freunde sind, mir die Schulzeit überhaupt erst erträglich gemacht haben, vor allem den Französisch-Unterricht, aber auch später in der Zivi-Zeit und vieles mehr. Sie sind auch heute auch trotz aller Änderungen, die ich als Mensch durchlaufen habe bereit ihre Zeit mit mir zu verbringen. Alle meine Kollegen aus Belle II, vor allem die Kollegen aus München Christian, Sebastian, Felix und Sara, die Kollegen aus Korea InSoo, Cheoulhun, JaeBak, KyungTae, SungHyung, Youngjun und die des KEK Iwasaki-san, Yun-Tsung, Nakazawa-san, Koga-san, mit denen ich eng zusammengearbeitet (und Belle II gefeiert) habe. Meine Arbeitskollegen der Gruppen Becker, Sax, Müller-Glaser und Stork, es war ein großes Glück in einem Arbeitsumfeld mit vielen tollen Menschen arbeiten zu können. Vor allem Kai, Lidia, Florian, Augusto, Arthur, Houssem, Oliver, Tim, Stephan, Stefan, Jan, Kevin, Steffi, Tobias, Lukas, Johannes, Jijing, Simon S. mit denen ich viele Sachen unternehmen konnte. Hannes, Simon, Fabian, Harald und Michael mit denen ich viel unternommen habe wie Kino, Reisen, Sportveranstaltungen, die mir aber immer zugehört und geholfen haben. All die super Studenten die ich betreuen durfte in Abschlussarbeiten oder Vorlesungen. Vor allem Kai, Adam, Julian bei denen die Betreuung am meisten Spaß gemacht hatte. Falco und Viet aus der alten ARAMiS-Gruppe, die mir vor allem am Anfang das Leben am Institut einfacher und angenehmer gemacht haben, viel geholfen haben, immer da waren auch nur für ein Gespräch, oder auch zu privaten Sachen bereit waren. Den Kollegen der Werkstatt und der Sekretariate.

Die wichtigsten Personen für meinen Erfolg zur Promotion, abseits von mir selbst, möchte ich hier zusätzlich Danken. Oli für seine Betreuung von mir als er damals PostDoc war, vor allem als ein jemand mit dem man über alles reden konnte. Er war auch einer der Menschen die auf eine Art und Weise kritisch mit mir reden konnten, so dass ich anschliessend wusste was ich verbessern sollte. Timo, bereits für seine Betreuung meiner Diplomarbeit, aber dann vor allem für die gemeinsame Zeit als Büro-Nachbarn. In den letzten Jahren habe ich wohl mit keinem Menschen mehr Zeit verbracht als mit ihm, eine bessere Person dafür gibt es aber auch nicht. Prof. Becker für die jahrelange Betreuung, die gemeinsame Zeit durch die HSO-Vorlesung, aber auch generell für die gemeinsame Zeit z.B. bei verregneten Baseball-Spielen mit neugewonnen amerikanischen Kollegen, die "SaufN'Spiel" erklärt haben wollten. Schließlich meine Freundin, die mir vor allem gegen Ende sehr geholfen diese Schrift auch mal zu Ende zu schreiben. Ohne ihre durchgängige Motivation, Geduld und Unterstützung würde ich heute noch daran schreiben.

Karlsruhe, Juli 2021
Steffen Bähr

Contents

1. Introduction

Particle accelerator experiments are one of the cornerstones for humanity's efforts to advance its knowledge about the fundamental laws of physics that are underlying our world. They provide the experimental setting for generating and observing physics processes that are otherwise hidden from us and only assumed to be existing through theoretical models. The basis behind exploring new physics beyond our knowledge is to recreate conditions under which never before seen particle decays are revealed. The goal is to compare their observed behaviour with previously established theoretical models that are describing the nature of particles. Advances typically require the creation and observation of very rarely occurring particle decays. There is always knowledge to be gained when observing these, irrespective of them matching or deviating from the theory of established models as they are either validated or indticate the extistence physics beyond our knowledge. The best-known discovery in recent years has been the observation of particles corresponding to the predicted behaviour of the Higgs boson [27]. This discovery's reach spread far beyond the community of particle physics as most of the world's population, with an interested in science and access to the respective information, followed the progress of this discovery. While its experiment, is without a doubt, the most well known around the world, other less known efforts led to similarly significant advancements towards new physics. The most important examples within the context of this thesis are the Belle and BarBar experiments from 2001. Both were able to record never before seen particle decays, that paved the way for experimental proof of the Charge and Parity (CP)-violation, which is one of the assumed major physics mechanisms that lead to the asymmetry between matter and antimatter as it is present in our world [12, 4].

New insights are typically gained in modern experiments by the colliding particles at extremely high energies to improve the chances of generating the targeted rare particles and their decays. Improvments in the generation of collisions is achieved by further development of the particle accelerator used together with the experiment. This development is mainly focused around tuning of the energy and increasing its luminosity. Both are designed depending on the scientific context of the experiment. The development of these two characteristics over the last years is shown in figure 1.1 for various experiments [102]. Most importantly here is to note that the current world record in luminosity is held by the KEK-B-factory (KEKB) accelerator, which is the predecessor of the accelerator that is used by the Belle II experiment. The Large Hadron Collider (LHC) is meanwhile producing the by far highest energies and are even planned to be surpassed by its upcoming upgrades.

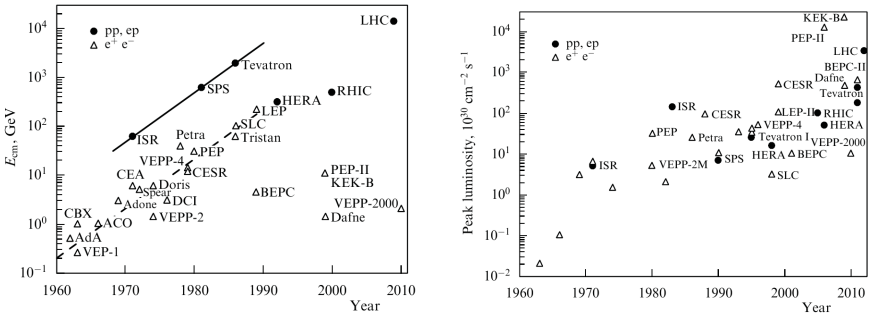


Figure 1.1.: Developments of both energy (a) and luminosity (b) in particle accelerator experiments over the last 50 years [102].

Detectors are installed around the interaction point at which particles are colliding to allow the observation of the resulting new particle decays. These detectors are designed to record information about the behaviour of the newly generated particles. Examples of recorded information are the energy and path of the particles through the detector's space. These detectors are typically heterogeneous in their design, as they are specialised to fulfil specific tasks, for example, high-resolution detection of the energy as close as possible to the interaction point. The complete information about the observed event is then reconstructed by combining the data of all the installed detectors. The accuracy of the reconstruction is meanwhile highly sensitive to the specific characteristics of the used detectors. Properties such as the distance of the detector to the interaction point, the spacing between adjacent sensors, and the capability to accurately digitize deposited energy of a particle are essential to achieve a high precision reconstruction. Witch each new experiment more technologically advanced detectors are employed to offer better recording and reconstruction. Typical approaches for improvement are the usage of more detector layers around the interaction point, more and denser located sensors within each of these layers, improvement of a sensor's readout characteristics as well as placing the detectors as close as possible to the interaction point. These developments is highly driven by the increase in both energy and luminosity provided by the accelerator. Examples of most recent developments that are about to shatter previously established high-end characteristics in particle physics experiments are the Gigatracker [31], with its amount of present channels, or upgrade of the ATLAS [34] experiment, with its extreme data rates. However, these are detectors that are currently in development and years will pass before operation will start.

With an ever-larger number and higher density of sensors integrated into the detectors, modern experiments are possessing never before seen quantities of readout channels. As the number of channels is typically scaling with the data rates that are read out from the detector, this necessitates significant improvements of the infrastructure used for the data readout. To support the data rates generated by so many channels, complex and expen-

sive data acquisition systems have to be developed. As most of these experiments are operated with consecutive collisions, the data readout is mostly designed for pipelined operation. In order to avoid any occurrence of dead time, in which the detector cannot be read out due to congestion within the readout, recorded data has to be processed as quickly as possible to keep up with its generation. While the before-mentioned experiments are providing a glimpse into the future of the extreme anticipated data rates ahead, they are already immense at current modern experiments. One such example is the Belle II particle accelerator experiment at the High Energy Accelerator Research Organization (KEK) in Japan, which aims at achieving a world record luminosity. It, for example, employs a novel Pixel Detector, the innermost detector in the experiment, which on its own is generating data rates of up to 1 MByte/event, showing the extreme pressure that is going to be put on its data readout systems.

While the ever-increasing amount of data generated at such experiments is enabling a more precise analysis of the underlying physics processes, the generated data rates can only be transferred from the detectors away to the data centres with great effort and cost for the transmission infrastructure. Most modern experiments aim at reducing the generated amount of data before it is even being sent to these data centres to avoid excessive cost. The most efficient solution is typically achieved by reducing the data rate to be sent as early in the readout system as possible, thus avoiding the need of installing an expensive high-capability transmission infrastructure. Modern experiments are aiming at moving the systems tasked with this reduction as close as possible to the detector's sensors due to the extreme data rates that are expected.

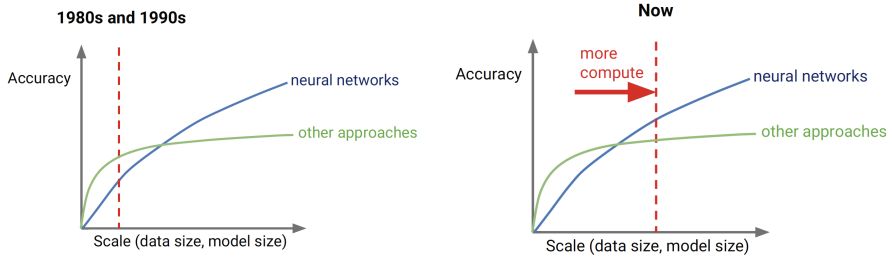
Traditionally two separate approaches are employed for the overall data rate reduction. One is the suppression of the total amount of data to be sent for each event to be stored. The other approach is aiming at the reduction of the number of events that are to be stored. By using combinations of these approaches, the total data rate to be stored can be efficiently reduced. Suppression of events is traditionally the responsibility of the trigger systems in such experiments. These systems are tasked with deciding whether data that is read out from the detectors is to be sent along for further analysis or is to be discarded to reduce the rate. Such trigger systems are typically designed in a multi-staged and pipelined architecture, with each stage having different requirements in terms of latency, resources, and throughput. The stages are organized in levels, with the system closest to the detector typically being the level 1 trigger.

Both mechanisms are based on the fact that only a small fraction of the overall data is actually essential for subsequent physics analyses. New physics is found through rarely occurring events, while common processes are not having a high contribution towards advancement. The general idea is subsequently to store only a predefined subset of the observed events, those which are of importance to the experiment's underlying physics goals. A typical example of such data reduction is the usage of a track trigger. These triggers are estimating the curvature of an observed particle track with the goal to determine the type of particle that was observed. This estimation is then used for discrimination between important and unwanted events. The problem of exceddingly high data rates can then be solved by performing classification tasks during online operation.

The unique circumstances of an experiment's operation typically enforce strict and tight requirements on the implementation of such systems. Successive collisions require both

low latency trigger decision making and matching of the detector's data rates. Additionally, even though not representing the entire detector's data, the amount of data to be received in order to be able to make trigger decisions is enormous with respect to common processing systems. These requirements are typically leading to the employment of specialised hardware systems. While Application Specific Integrated Circuit (ASIC)s can achieve the best results, these tasks are often carried out by FPGA platforms. These can provide the desired high amounts of high-speed Input/Output (IO), deterministic real-time processing and potential for adaption after their integration.

1.1. Motivation



(a) (b)

Figure 1.2.: Comparison of neural network-based processing with traditional approaches over the last years in terms of their accuracy relative to the scale of data to be processed [23]. While traditional approaches were representing the best solution in the past (a), the current increase of available computing power favoured the usage of neural networks as they are scaling better due to for example their inherent degree of parallelism (b).

Keeping outgoing data rates in check has been a challenging task in the domain of high energy physics experiments for a long time. Nowadays, it is rather another class of the overall big data challenge our modern society is currently facing, as most of the services used today rely on large scale collection and processing of data. The task is basically the same for all domains, in which raw data itself is not usable. However, when processed correctly, significant insight into the underlying processes can be gained. The current trend for the processing approaches in the wake of these massive data collections is the usage of fast, efficient, and low maintenance approximation of the desired solutions instead of their mathematically exact counterparts. This is where approaches based on machine learning ascended in both industry and science to tackle big data processing challenges. The development is graphically shown in figure 1.2, which illustrates the achieved accuracy of a processing solution depending on the scale of data to be processed [23]. While traditional approaches were achieving better results in the past, nowadays, neural net-

works are providing an overall better solution with regards to the amount of data to be processed.

The principle of neural networks is to use an algorithm that configures or learns parameters to solve any arbitrary processing problem. The processing operations to be performed are not custom-generated by an algorithm engineer but are instead derived from the configuration of very flexible operators called neurons. This configuration is performed in a way that leads to the correct classification of a select number of test data sets. Even though sometimes being excessive in their demand for processing power, these approaches are capable of achieving very good results without investing too much effort in the research of an exact algorithm. At the same time, these machine learning solutions often prove to be behaving quite well even in the presence of data sets that were never seen before,. They are thus often robust against unknown noise. Recent scientific developments in machine learning solutions solved its inherent problem of high resource demand. By combining traditional processing solutions with the universal processing capabilities of these algorithms, the required processing was significantly reduced, resulting in widespread employment even on traditionally constrained platforms such as embedded systems.

The main motivation of this thesis is to apply approaches based on machine learning to the data rate problems present in modern particle accelerator experiments, in particular for the real-time trigger at level 1 and online data reduction. Although this general approach has been done before it was on a much smaller scale, for example, at higher-level trigger systems with much more forgiving requirements. Meanwhile, when considering the offline processing domain of an experiment, in which close to infinite processing resources are present, these algorithms already proved to be a viable solution as they were and are still used for classification and identification of particle decays. For example, the Belle experiment made large-scale use of the NeuroBayes algorithm to analyse generated data offline. Nowadays, ever more experiments are looking into using machine learning for data analysis. Its feasibility is investigated as part of the TrackML challenge hosted by the European Organization for Nuclear Research (CERN) [20] at which computing tasks are openly announced to the public and supported by a prize money pool for the best solutions. While those are representing popular offline solutions, employment at the online processing stage of an experiment is rather uncommon. Despite their assumed suitability, such approaches have not yet been used in any First Level (L1) trigger system thus far.

In the past, the high demand for computing power set an insurmountable hurdle for large scale usage of machine learning solutions. The situation significantly improved with the development of computationally less intensive alternatives. However, its ascend is still driven by the increase in available amount of processing power. With the further reduction of the technology node in present modern semiconductor technology and the associated increase in integration densities, today's hardware platforms are possessing a massive computing capability that is at the same time cheaply available. Especially machine learning algorithms are profiting from the increased pool of available computing resources. Most of today's high-end mobile devices already contain dedicated neural network-based hardware accelerators, e.g., used for image optimization.

A realization of a neural network-based trigger system on FPGAs at L1, however, does not yet exist thus far. Besides the necessary studies to be conducted in order to find a suitable machine learning solution together with a topology and potential preprocessing,

both the implementation and integration within such tight requirements are not easy to realize. All processing has to be carried out in real-time and within a low latency budget. At the same time, many high-speed IOs have to be supported in order to receive and send the required data within the entire trigger system. Since the algorithms have to be adaptable to the observed behaviour of the experiment, concepts have to be created to update the employed algorithms in the field. In addition, such a trigger can only be used with a complementary online monitoring approach that ensures correctness.

Traditionally, the biggest hurdle to be overcome when implementing for FPGAs is the transition of the algorithm to the dedicated logic processing structures used in this technology. Both software and hardware development, which are based on very different paradigms, face each other at this point. FPGAs are often still programmed with a dedicated Hardware Description Language (HDL), which implements the concepts of parallelism in a fundamentally different way. The development for FPGAs thus requires in-depth knowledge of the technology in order to find an optimized solution that can meet the strict overall requirements dictated to L1 trigger systems.

1.2. Contribution

While algorithms from the class of machine learning, especially neural networks, are nowadays a popular occurrence within the FPGA community in the domain of real-time classification problems, they are only a rare sight for L1 trigger or online data reduction systems used in modern particle detector experiments. This is mostly due to the strict requirements to be fulfilled, mainly a fixed throughput coupled with a hard latency budget. Herein lies is the core of this thesis's contribution, as it represents the documentation of the first and at the moment only neural network-based L1 trigger system that is integrated, fully functional, and delivers tack estimations that are vital to the success of its targeted physics experiment. The contributions do not stop here, as two optimizations for the entire track trigger system are presented and discussed. They were primarily developed to further boost the trigger system based on neural networks. However, even in stand-alone, they are representing a substantial upgrade over the initial design. The usage of these kinds of algorithms in the context of particle accelerator experiments is further explored with the prototypical development of a neural network-based online data reduction system.

This thesis presents the design of the neural z-Vertex trigger for the Belle II experiment. It contains an integration strategy that fulfils all interface requirements, allowing to cover the entire space of the experiment's drift chamber. A flexible FPGA-based architecture is presented and discussed that is capable of achieving all of the resource, latency, and throughput requirements. It is designed to be easily adaptable for usage with newly trained neural networks in order to reflect the current operational status of the experiment or to be scaled to different requirements. Hardware-based implementations for both slow control and data quality monitoring are presented that contribute towards ensuring correctness. These are the cornerstones of proving correct functional operation. At last this system is capable of achieving not only all non-functional requirements, it can recreate the software-based implementation of the neural trigger concept almost entirely,

not only in simulation but during runtime operation as supported by the data samples that are presented within this thesis.

The upgrade of the L1 trigger system is meanwhile centred around the optimization of the basic processing step of the system, the TSF. Data presented within this thesis shows that the approaches developed here are increasing the readout efficiency significantly. Further improvements of the entire trigger system are investigated by the development of a prototypical Hough-based 3D track finding as a preprocessing approach on FPGAs that is using a weighting scheme based on Bayes theorem. The transition towards integration into the operation of the experiment is investigated. This investigation focuses on the examination of the possible opportunities and consequences of integration into the trigger system.

Lastly, this thesis presents the a FPGA-based realization of the NeuroBayes algorithm, which was initially used for particle classification in server farms. It is used for the prototypical implementation of an online data reduction system, which allows the identification of defined types of particles by using the pixel detector of Belle II. The limits of the implementation are investigated by the development of a high throughput demonstrator that is independent of the restrictions set by the experiment's integration requirements.

1.3. Outline

The contents of this thesis are presented according to the following structure. Chapter 2 presents the fundamentals for understanding the described systems and approaches. First, the Belle II experiment is presented in section 2.1. For this, general information about the experiment and its goals are provided. This is followed by an overview of the detector concepts developed for the experiment and accompanied by an introduction to its respective sub-detectors. The focus is then shifted towards the experiment's trigger system in section 2.2. The core of the examination represents the trigger system that is operating with the data from the Central Drift Chamber, which is the targeted detector for the trigger approach developed within this thesis. After establishing the fundamentals of the trigger system, the Data Aquisition (DAQ) of the experiment is presented in section 2.3. It also covers the general approach to storing data within the experiment's infrastructure. This will focus on topics relevant to the development of the systems targeted within this thesis. It features the employed general solutions for data reduction methods that are developed for the Pixel Detector. Supporting mechanisms such as the slow control and data quality monitoring will be presented, as these are mandatory for all systems. This concludes the discussion of the topics about the Belle II experiment, as such section 2.4 is representing an introduction to FPGAs. This will cover the general architecture of modern FPGAs as well as the general design flow that is used for development for such systems. Since machine learning and especially neural networks are forming the algorithmic basis for the systems developed here, the basics of these methods are presented in section 2.5.

Chapter 3 presents the state-of-the-art relevant to this thesis. It is divided into the two areas of data reduction and trigger techniques in the environment of particle accelerator experiments together with a brief discussion about the development of dedicated

hardware-based accelerators for neural networks. It begins with the related track trigger approaches of other experiments in section 3.1 that are either using machine learning or solving similar problems compared to the systems developed in this thesis. In section 3.3 the focus is put on modern solutions for implementing and inferring neural network architectures, covering both academic and commercial solutions.

Chapter 4 provides a more general view on the requirements present at the applications that are this thesis's focus. A discussion about the special requirements of data reduction and trigger systems within modern particle accelerator experiments and the implications of using machine learning methods is provided in section 4.1. Considering these requirements, a general architecture template for developing such systems is provided in section 4.2 together with applied design principles. This template is used throughout all neural network-based developments in this thesis. As all approaches require a similar development process, the design flow template used throughout this thesis is presented in section 4.3. While each system is requiring some parts to be custom-designed for the respective application, all are sharing the usage of neural networks. As such a general investigation of the possibilities for realizing these on modern FPGAs is presented in section 4.4. The fundamental strategies for realizing neural networks presented in this section are used throughout the targeted applications discussed later on.

The main system developed in this thesis is the neural z-Vertex trigger of the Belle II experiment. It will be presented and discussed throughout chapter 5. First, the experimental motivation behind the system is presented in section 5.1. This includes a brief physics point of view but will also focus on the algorithmic development and evaluation of its performance. The section is concluded with an examination of the specific requirements to be fulfilled by the system. After establishing the requirements, the focus is put on the realization of the system throughout section 5.2. Its presentation is starting with an investigation of the possible integration strategies into the overall trigger system. This covers the connection to all other sub-components and the realization of the communication protocols to be supported, including the approach to covering the entire space of the central drift chamber. The presentation of the general integration strategy is then concluded by the selection of a suitable FPGA platform that is fulfilling all requirements. The design of the detector-specific preprocessing is subsequently presented in section 5.2.4 together with figures of merit for each individual module that was developed. The section is concluded by the investigation of configurations to be used for the Multi Layer Perceptron that culminates in the system's fulfilment of all requirements. After establishing the realization of all functional elements, the focus is shifted to tools and mechanisms supporting the operation of the system, which are presented in section 5.3. This includes the concrete design flow used to infer the architecture in the presence of new network weights. Monitoring and validation mechanisms, together with their hardware-based implementation, are then presented in the subsequent sections. Section 5.4 is then focussing on the operational systems used within the experiment, starting with setups that facilitate in detail hardware testing. The culmination of all conceptual and design tasks is then presented within section 5.4.2. The two most important realizations of the neural z-Vertex trigger are presented here. Especially the final system used during physics operation is discussed in section 5.4.2.2. The achieved final characteristics of the system are presented here. This covers all non-functional aspects such as latency, throughput, and resources, but also a functional analysis of the system's capability to estimate the desired z-Vertex.

Both analyses are presented with data and measurements derived from operation within the experiment from the 2019 and 2020 runs. The chapter is then concluded with an outlook into upgrade opportunities for the system in section 5.4.4.

The following two chapters are then presenting general upgrades to the trigger system of the drift chamber that will improve overall performance. The beginning is chapter 6, which describes the prototypical development of the Hough based 3D track finder, that is planned as an upgrade. The chapter's structure is similar to the previous chapter starting with the functional description of the system presented together with its specific requirements in section 6.1. Following this, the prototypical realization is presented in section 6.3, starting with integration aspects and description of all hardware modules that were developed. The accompanying design flow is presented in section 6.3.4. The chapter is concluded by presenting strategies for the final integration in section 6.4.

An upgrade in smaller scope is the main topic of chapter 7. It features an upgrade of the track segment finder that is used by all sub-components of the central drift chamber's trigger system. The chapter begins with a dissemination of the initial system responsible for this task in section 7.1. A description of the upgraded approach chosen for the upgrade as well as the implementation are featured throughout section 7.2. The topic is concluded in section 7.3 with a presentation of the developed test setups as well as an evaluation of both functional and non-functional characteristics, the former being based on data taken from the experiment.

The last major topic of this thesis is the online data reduction approach based on the NeuroBayes algorithm. It is featured throughout chapter 8. Section 8.1 introduces the physics-based motivation for the usage of this system, together with an analysis of the requirements to be met. The realization of the system is described in section 8.2, starting with all aspects regarding the system's integration. The developed architecture and in-detail description of each module is part of section 8.2.2. The used design flows are afterwards discussed in section 8.2.3. The scope of a data reduction system based on the NeuroBayes is then extended beyond the provided application case of Belle II within section 8.3. Here, a high-throughput demonstrator is presented that explores the possible performances that can be achieved by the previously developed architecture.

Concluding the entire thesis, chapter 9 is discussing the general achievements and providing an outlook into future systems.

2. Fundamentals

2.1. The Belle II Particle Accelerator Experiment

Particle physics is one of the cornerstones of humanity's understanding of our universe. In its most general form, as an understanding of the composition of the world from the smallest inseparable parts or atoms, it is part of the basic knowledge of every human being of the educated world. While this area is familiar to most, it is only the most abstract basic concept. History repeatedly showed that particles considered as inseparable are again composed of other particles and phenomena that were not previously observed. In modern physics, the original concept of the inseparable atom has been refined into the standard model, which is used as the current reference. This model consists of a multitude of different particles, which is called a particle zoo due of its diversity. For each particle, it describes classical properties such as charge and mass, which also belong to the general understanding of particle physics. However, it also contains more abstract properties such as strangeness or charm, the effects of which require a deeper understanding. To validate the model, the particles described in it are generated in an experiment and their properties are measured. The preferred method for this is the operation of particle colliders in which selected particles are accelerated under high energies and brought to collision to create the desired new particle. For detection and measurement, a detector is built around the interaction point. In modern particle physics, such a detector consists of several sub-detectors with different properties that are tasked with measuring a specific properties of the observed particles. The thematic core of this thesis is based on the particle detector in operation at the Belle II experiment.

The standard model can be divided into three categories of particles: leptons, quarks, and gluons. Part of the leptons are, for example, the electrons but also their counterpart, the positrons, which have a positive instead of a negative charge. The two accelerated beams that are brought to collision at Belle II consist of electrons and positrons respectively. Since positrons have a rather rare occurrence in our observable environment, they are categorized as antimatter. Meanwhile, particles that are common to us, like electrons, are part of matter. Antimatter does not only consist of positrons, rather each particle of matter always has a corresponding counterpart. They differ in at least one characteristic such as charge but can differ in for example possessing an opposite spin as well. Here, too, particle accelerators are used to generate the energies needed to create matter/antimatter pairs. The question of why our observed universe is dominated by matter represents the core of the scientific objective of the Belle II experiment.

In addition to electrons, the group of leptons consists of muons and tau particles. However, these are occurring only rarely, since they are not stable and quickly decay into other particles. The different manifestations of the leptons are called flavour, a term often used

in discussions around the physics of Belle II. So far little is known about tau particles and their antimatter counterpart. One of the objectives of Belle II is to detect these particles. Since their occurrence is so rare a high luminsoity coupled with a high-efficiency trigger is mandatory to have a chance of actually recoring them with meaningful statistics.

Besides leptons, quarks are also of interest in the experiment. They make up the largest part of the mass and by a combination of quarks with anti-quarks are forming most of the known particles. Stable combinations are hereby called hadrons. These can, in turn, be distinguished between baryons and mesons. Baryons are exclusively made up of combinations of three quarks with protons probably being the most well known members. Mesons are more uncommon, they always consist of a matter-antimatter pair. A part of the mesons are the B mesons, or sometimes called Beauty. These are also eponymous of the Belle experiment since it aims to create B meson pairs from collisions.

Particle decays takes place according to the defined laws of physics. One of the better known is energy conservation. In particle physics, the initial mass, for example, two colliding particles, cannot decay into products that have a higher overall mass. It has been shown that laws of conservation are the result of prevailing symmetries.

For Belle II, the symmetries of charge conjugation and parity are the most important in order to understand the goals of the experiment. Charge conjugation is the symmetry that is easier to understand. It describes that the antimatter counterpart of matter has the reversed charge while the other properties are preserved. While it is preserved experimentally for most of the occurring interactions, this is not the case for the weak interaction. Parity meanwhile describes a spatial symmetry between related particles. To exemplify this, two coordinate systems are defined for the observation of a particle decay, one is defined with (x,y,z) and the other being a reflection with (-x,-y,-z). Parity is now obtained when a decay appears the same in both systems with respect to the coordinate system. The decay appears to be mirrored in the opposite system. Here it has been observed that the symmetry of parity is violated for the weak interaction. The combination of these two conditions is also called the violation of the CP symmetry. This violation is the main motivation of the Belle II experiment and requires more detailed analyses based on experimental data.

The Belle II experiment [7] with its two use cases OCA and NNT provides the context for this thesis. This chapter presents the background information about the experiment. The detector built for the experiment is described together with all its sub-detectors in section 2.1.1. Subsequently, the most important detectors for the use cases discussed in this thesis are presented in section 2.1.2. These detectors provide the input data for the systems developed here and are used for the identification of particle tracks. The following section 2.2 focuses on the trigger system of the experiment, especially the CDC. The basics for understanding the online data reduction are presented in section 2.3.1 in which the data reduction system for the VXD is presented.

2.1.1. Overview of the Experiment

Belle II is an experiment connected to the SuperKEKB particle accelerator, which is located at Tsukuba, Japan. At this experiment, two separate particle beams of electrons and

positrons, e+e-, are accelerated and collided at its interaction point. Different energies are used for both of the accelerated beams, thus it is an asymmetric accelerator. Positrons are stored in the storage ring at an energy of $4GeV$, while electrons are stored at $7GeV$. Compared to its predecessor Belle II it is intended to reach a peak luminosity of $8*10^{35}$, which corresponds to a forty-fold increase of the peak luminosity of $2.1*10^{34}$ of the predecessor Belle. Technologically this increase is achieved mainly by employing a so-called "Nano-Beam", which reduces the cross-section significantly [19]. One of the goals of the experiment is to measure the weak interaction and thus the violation of the CP-symmetry more precisely. The amount of B mesons produced is increased due to higher luminosities, in turn, resulting in better statistics leading more accurate results and highe occurance of the desired rare particle decays.

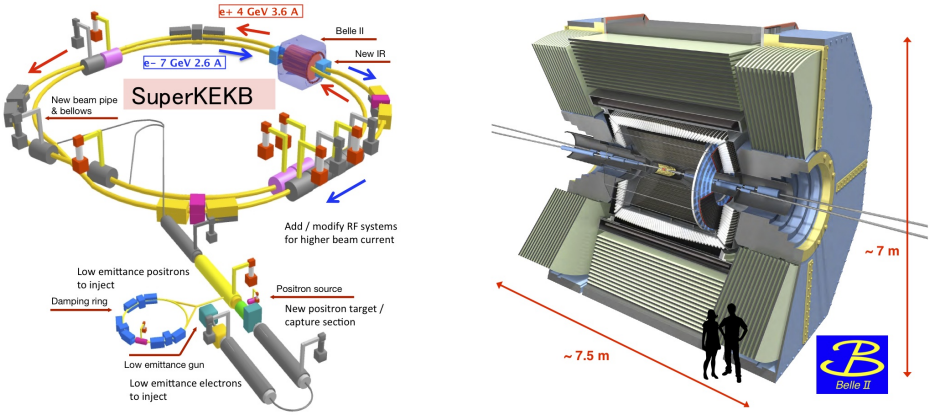


Figure 2.1.: Layout of the SuperKEKB accelerator ring on the left; and a grahpical composition of the Belle II particle detector [15].

An essential part of the observation of collisions is an exact reconstruction of the subsequent decays. The experiment uses hereby three detectors for reconstruction of particle tracks. These are the PXD, Silicon Vertex Detector (SVD) and CDC. The PXD and the SVD are completely new developments for the Belle II experiment, while the CDC is mostly reused from Belle. The PXD forms the innermost detector of the experiment and is placed directly around the interaction point. It provides a much higher precision for spatial measurement, but has a relatively long integration time due to the used sensor technology. The SVD is enclosing the PXD. In contrast, it is a strip detector with less accuracy in spatial resolution but a shorter readout time and higher coverage of its occupied space. These two detectors together form the VXD. They are enclosed by the CDC. Its main task is to estimate a passing particle track's momentum and charge. These three detectors are used to determine decay vertices and espescially find tracks with low momentum.

The tracking detectors are enclosed by particle identification detectors. The Time of Propagation Detector (TOP) is located directly after the CDC at the barrel, while the Aerogel Ring Imaging Cherenkov Detector (ARICH) is installed at the endcaps. These detectors

are enclosed by the Energy and Cluster Detector (ECL), which is used for the detection of photons and the identification of electrons. Finally, the Kaon and Muon Detector (KLM) is used to identify K_L^0 and muons [13].

2.1.2. Sub-Detectors for Track Finding

The three tracking detectors of the experiment are of particular interest, as they essentiell for the methods and systems presented within this thesis.

2.1.2.1. Pixel Detector

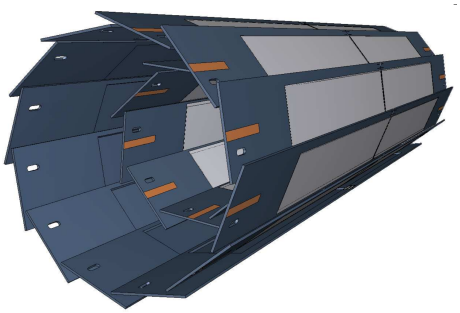


Figure 2.2.: 3D Rendering of the PXD. DEPFET matrices are shown in grey [5].

The primary purpose of the Pixel Detector (PXD) is to capture high-resolution data of particle decays as close as possible to the interaction point in order to determine an exact reconstruction of the origin of a particle track. The detector consists of two layers of silicon sensors. Here, DEPFET technology [48] was selected for the sensors. The decisive advantage is its ability to precisely record the charge left behind by particles. When a particle passes through the sensor, a defined amount of electron-hole pairs is generated and the particle is loosing momentum according to the Bethe-Bloch equation [79] also shown in figure 2.3. The deposited charge can then be determined by using the current at the drain connection of the sensor. This capability is important for the experiment since the momentum of the particle can be determined from the measured charge.

The response of these sensors is of particular interest for the identification of an observed particle based on pixel data. Depending on the charge that was read out and the resulting momentum, it is possible to determine which kind of particle has passed through the sensor. This property is used in chapter 8 to identify slow pions by only using PXD data.

The two layers of the PXD are arranged with a radius of $14mm$ and $22mm$ to the interaction point. Each layer is then divided into so-called ladders, whereby the individual ladders are arranged in an overlapping alignment to avoid uncovered space. This arrangement is shown in figure 2.2. Sensors are arranged on two half modules for each ladder. Each of these modules contains a sensor matrix consisting of 250*768 DEPFET pixel sensors.

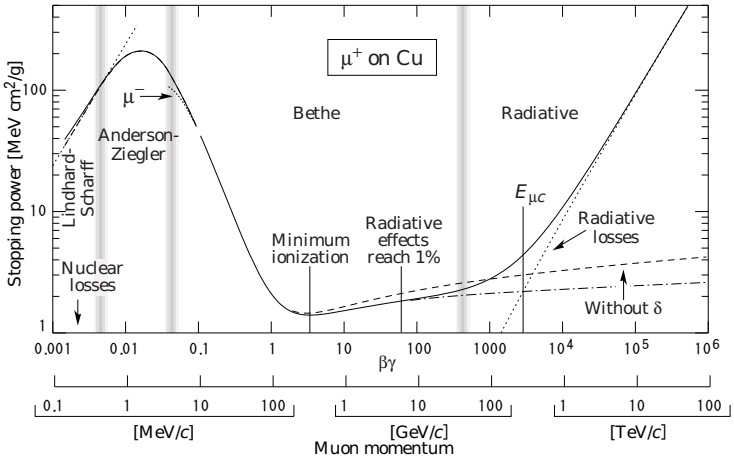


Figure 2.3.: Bethe-Bloch plot that shows the relationship between momentum and energy loss [79].

In addition to this matrix, an Integrated Circuit (IC) is integrated into the module for the control and readout. To reduce costs these ICs are integrated on the same die. The first layer of the detector contains 8 ladders while the second layer contains 12 to cover its higher radius. In total, the PXD has thus 7.68 million pixels. The properties of the ladders are summarized in table 2.1.2.1.

Property	PXD
Technology	DEPFET
Layers	2
Radius	14/22mm
$Pixel_{ladder}$	250*768*2 pixels
$Pixel_{total}$	7680000 pixels

Table 2.1.: Properties of the PXD.

The maximum data rate generated by the PXD is of particular interest for this thesis. It is determined by the maximum number of active pixels at a given readout window. For this the maximum occupancy in addition to the total number of present pixels is decisive. The occupancy describes the ratio between all existing and all of the active pixels for an event. The data rate can then be calculated by using these characteristics. Subsequently, the required compression can be determined by comparing the worst case data rate with the available data rate budget.

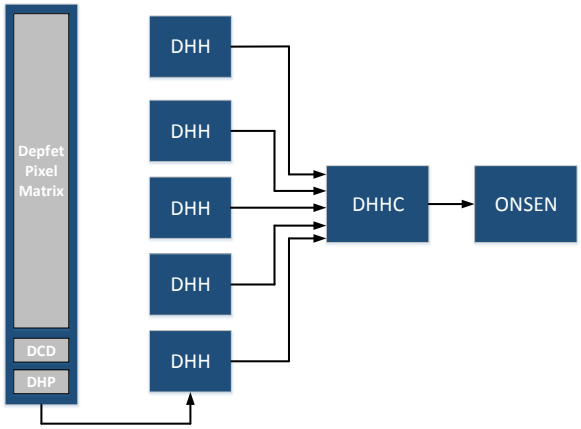


Figure 2.4.: System architecture of the readout system of the PXD.

The data readout chain of the PXD is shown in figure 2.4. The DEPFET matrix of a half module is handled by three components [18]. A pixel matrix is controlled by a switcher ICs. Sensor signals are meanwhile read out by the Drain Current Digitizer (DCD) which also digitizes the values. Digitized values are afterwards processed by the Data Handling Processor (DHP). The DHP has several tasks e.g. so-called pedestals, constant offsets inherent to a pixel, are subtracted from the digitized value. Additionally, in order to determine the deposited charge, a common mode, another constant offset, is subtracted. Before data is finally sent to the next stage a suppression of zero charge values is carried out. Here, the charge of individual pixels is only used when it is above a certain defined threshold. The data is then forwarded to the Data Handling Hybrid (DHH). It handles all the data of one ladder and is based on FPGAs. Its main tasks are the conversion of the Low Voltage Differential Signalling (LVDS) data from the DHP to transmission on optical fibres and load balancing. At the same time, clusters are formed based on the pixel data. Data sent by five DHHs is concentrated at the Data Handling Controller (DHHC). Concentration of data across all ladders as well as the final data reduction mechanisms are hosted at the ONline SElector Node (ONSEN), which is another FPGA-based system.

2.1.2.2. Silicon Vertex Detector

The Silicon Vertex Detector (SVD) consists of the sensor layers directly following the PXD. In contrast it is based on strip sensors, which can only capture one-dimensional information. They resolve the spatial information of the decays with less accuracy compared to the PXD, but since being further outside of the interaction point a larger space is covered for particle detection. Due to its placement it is experiencing a lower occupancy and is thus producing smaller data rates. The whole detector consists of four layers. Together with the information from the PXD it is used to detect so-called Region of Interest (RoI)s

in order to perform the main data reduction for the VXD at the ONSEN. This mechanism is further explained in section 2.3.1 and represents the complementary procedure to the OCA discussed within this thesis.

2.1.2.3. Central Drift Chamber

The Central Drift Chamber (CDC) [108] of the Belle II experiment is located immediately after the VXD. Its main tasks are to support the reconstruction of particles, identification of particles by calculating the energy loss and most importantly to enable an efficient and reliable track trigger system for the experiment's readout. Part of this trigger system is the NNT, which is the main contribution of this thesis. To understand how trigger signals are generated by using data from the CDC, its structure is explained first.

Similar to the structure of the VXD, the CDC is also constructed according to a layered model. Each layer consists of a number of wires stretched parallel to the z-Axis. Within the CDC, all wires are surrounded by a gas mixture. This gas mixture consists of a low-Z gas (50% of helium and ethane respectively). Interactions of the passing particles with the gas mixture is producing charge carriers at the wires. These can be digitized and read out, and are then used to detect particle passing through the detector. The wires are grouped into layers. In total 56 layers are forming the entire detector. Successive layers are arranged with an increasing distance to the interaction point. Additionally, the number of wires varies across layers. Layers that are placed further away contain more wires to cover the correspondingly larger space to be observed for the detection of particles.

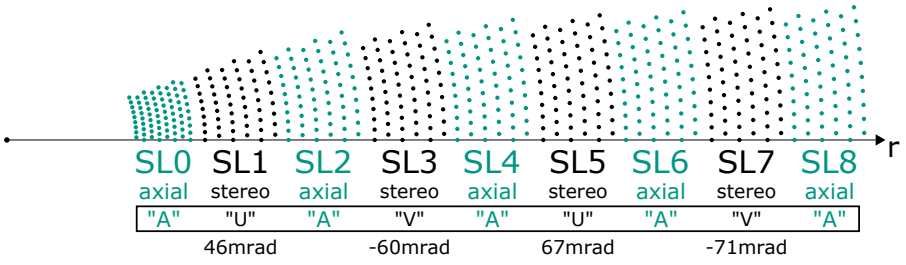


Figure 2.5.: Alignment and configurations of the Super Layers in the CDC [Poe18].

The layers are again grouped together into nine Super Layer (SL)s, that are enumerated from zero to eight. SL0 is closest to the interaction point with a distance of 168mm, while SL8 is the most outer one with a distance of at maximum 1111.4mm. All wires have properties that are defined by their SL. One such property is the orientation, for which they are either aligned parallel to the z-axis, called axial, or they have a stereo orientation in which they are aligned with an angle between 45.4 to 74.0 mrad depending on the SL. This angle is introduced to enable a three-dimensional reconstruction of tracks and is a key component of the approaches described in the sections 5 and 6. They can be idenfied by an index, all SLs having an even index are of axial oerientation. The remaining SLs

with an odd index all have stereo orientation. SLs with different orientation are arranged alternatingly within the CDC. The entire arrangement is shown in figure 2.5. A detailed listing of the properties of each SL can be found in table 2.2.

Super Layer	Layers	Wires	Distance	Angle
0	8	160	168.0 - 238.0 mm	0 mrad
1	6	160	257.0 – 348.0 mm	45.4 – 45.8 mrad
2	6	192	365.2 – 455.7 mm	0 mrad
3	6	224	476.9 – 566.9 mm	-55.3 – -64.3 mrad
4	6	256	584.1 – 674.1 mm	0 mrad
5	6	288	695.3 – 785.3 mm	63.1 – 70.0 mrad
6	6	320	802.5 – 892.5 mm	0 mrad
7	6	352	913.7 – 1003.7 mm	-68.5 – -74.0 mrad
8	6	384	1020.9 – 1111.4 mm	0 mrad

Table 2.2.: Properties of all Super Layers of the CDC.

The CDC is read out by the Frontend Electronics (FEE) [110, 39]. Each wire is then digitized with a 10 bit Analog Digital Converter (ADC). The readout frequency of the FEE is at 31.75 MHz. The resulting data rate is shown in formula 2.1.

$$Wires_{total} \cdot Bitwidth \cdot Rate_{Readout} = 14336 \cdot 10\,bit \cdot 31,75\,MHz = 4,55\,Tb/s \tag{2.1}$$

Generated data is at first reduced at the Readout Electronics for the CDC (RECBE) platform before being forwarded to the DAQ and trigger. Each RECBE is responsible for a total of 48 wires. The digitized data from the wires is used to determine the time at which a wire sensed activity within the CDC. For this a Time to Digital Converter (TDC) is used, which is capable to determine the timing with a resolution of $980ps$. The digitized and processed values are in parallel buffered in a ring buffer for the readout of the detector and sent to the CDCTRG. With the arrival of the trigger signal, data is sent to the next stage of the DAQ. The ring buffer is hereby designed to buffer data for 8 μs. This is also the main reason for the latency requirements at the L1 trigger, as the trigger signal has to arrive before data has to be discarded in order to free up memory for newly arriving data.

2.2. Belle II Trigger System

The trigger system of the Belle II experiment is divided among the individual sub-detectors into sub-triggers. Each sub-trigger delivers different information to the Global Decision Logic (GDL) about an observed event. These are used to determine whether the buffered detector data is to be read out [70]. The architecture of the entire system is shown in figure 2.6. Due to its importance to the approaches described in this thesis the CDCTRG will be described in greater detail, while the other systems are only briefly outlined.

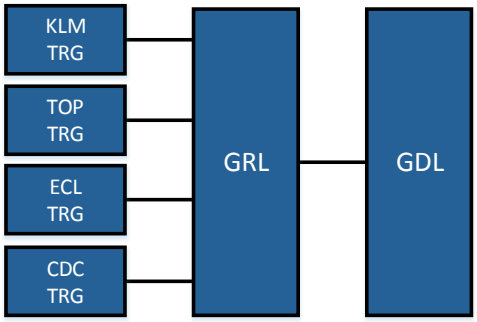


Figure 2.6.: Overview of the entire L1 trigger system used at Belle II.

2.2.1. Trigger System for the Central Drift Chamber

The Trigger system of the Central Drift Chamber (CDCTRG) is the most important sub-trigger of Belle II. It has the longest processing pipeline and therefore requires the highest overall latency to generate the necessary signals. For the entire processing from the front-tend to the GDL including all communication 5 mus is defined as the maximum allowed latency. The NNT developed in this thesis is part of this trigger system. To understand the background and resulting requirements, the CDCTRG is outlined in the following.

Overall the CDCTRG has a multi-level and pipelined processing architecture. Data of individual SLs is continuously read out by the FEE. It is then transferred to the merger units, which concentrate the data sent across all RECBE of one SL. The concentrated data is then sent to the first stage of the trigger logic, the TSF. The TSF combines the individual wires of the CDC into Track Segment (TS)s. Combination of TSs is performed for each SL separately. Additionally, a TSF can be distinguished into processing axials or stereos according to their orientation. Axial TSs are sent to the 2D track finder, which combines them to estimate 2D track parameters. At the same time the Event Time Finder (ETF) receives TS data to calculate the event time. The outputs from these two modules together with the stereo TSs are then sent to the Conventional 3D-Track Finding (3DS) and the NNT. Both of these have the responsibility to send an estimation of the z-Vertex to the GDL. In the following, the tasks of the individual modules are explained in more detail.

2.2.1.1. Combination of CDC Data by Merger Units

The purpose of the merger units within the CDCTRG is to combine the TDC values of several RECBE units and forward them. One merger unit is responsible for four RECBE units, while the number of merger units allocated to a specific SL is depending on the amount of present wires. The data from all merger units that are responsible for one SL is then forwarded to one TSF board at a data rate of up to 6.25 Gb/s. The merger is

implemented on an FPGA platform, in this case an Arria II GX FPGA [109]. The used platform is custom built and specifically developed for the merger use case of Belle II. As it in principle is still a general FPGA platform, it is also available to host for other systems within the trigger system, an investigation of its viability was conducted as part of Ref. [Poe18].

2.2.1.2. Functional Principle for Finding Track Segments

The main task of the Track Segment Finder (TSF) is to group together neighbouring active wires of the CDC. This grouping is performed according to predefined rules. They are defined with the idea of limiting the angle with which tracks are allowed to pass through one SL. Grouping and sending the wire information for the entire CDC would meanwhile gravely exceed the number of available IO channels on any of the available FPGAs with the resulting latency likely exceeding the allocated budget. To mitigate these issues, the grouping is performed separately for each SL on different FPGA platforms. Each TSF instance is hereby only receiving the information from all of the merger units that are connected to the respective SL. The wire information is then combined into these TSs. The used groupings use predefined geometric shapes that enforce the limitation of the opening angle. These geometric shapes describe which wires can be combined together. The shape used for the grouping is meanwhile not uniform across all SLs. To take the different properties of each SL into consideration, two different shapes are defined. Both are shown in figure 2.7. Here, wires are not shown explicitly but rather as boxes that are representing the space they are covering within the CDC. A pyramid-like shape is used for SL0, while all others are using an hourglass shape[1].

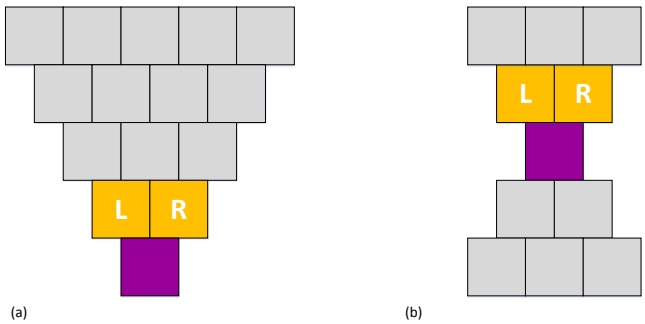


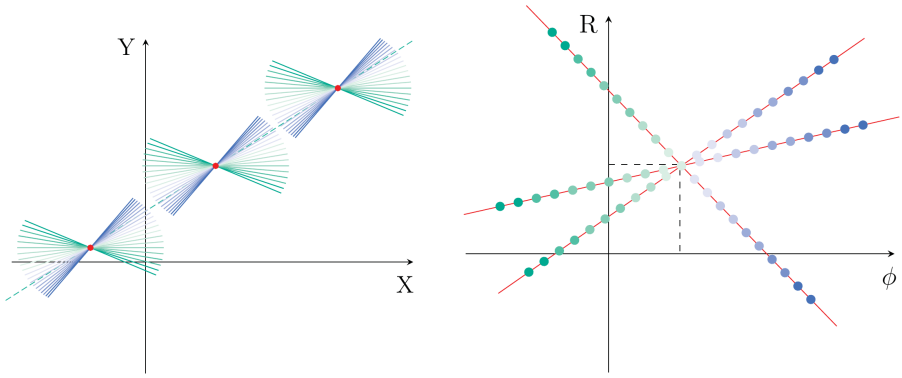
Figure 2.7.: Shapes of TS for both the SL0, pyramid shape (a), and the remaining SLs 1-8, hourglass shape (b).

Special wires are additionally defined within each TS. These wires are shown in figure 2.7 by boxes that are coloured in purple or yellow. Purple coloured boxes represent the so-

[1] While this approach is effective for "traditional" particles, the limitation in size of the shapes is preventing the detection of more exotic events spanning multiple TSs. Simulations have shown that dark matter processes can lead to such events. Current research for future trigger systems is looking into reimagining the TSF.

called primary priority wire. They are assigned a distinct number, which is used to identify a specific TS and its geometric location within a SL. At the same time, the timing value for this wire is used for further processing steps. The other kind of special wires are called secondary priority and are shown in yellow. Among other things, they are used to determine information about the direction of a passing particle track. For this, the two second priority wire cells are used. They are called left or right secondary priority wire. The estimated direction of a passing through particle track is then being determined by a pre-calculated Look Up Table. This is based on a design time analysis of a track's passage based on the simulation of events. Now for a TS to be considered active, wires in four out of five different layers must have been activated within a predefined time window. This means that every active TS will have at least one active wire that has either a primary or secondary priority. Within the scope of this thesis, the TSF is assuming a major role, as all trigger related work is either using its data directly or is even based around redesigning its core logic.

2.2.1.3. Estimation of 2D Track Parameters based on the Hough Transform



(a) (b)

Figure 2.8.: Example for track finding that is using the Hough transform [Hoc18]. A geometric view of the detector's space is shown in (a) while the resulting hough map is shown in (b). The best matching track candidate is found at the intersection point of all tracks in the hough map.

While the TSF is representing the basis for all track finding algorithms within the CDC-TRG, the 2D-Track Finder (2DS) is the basis for all 3D estimation algorithms and provides the basic track trigger signals for the experiment. It is thus of high importance for both the overall trigger system and the systems developed within this thesis. The 2DS used at Belle II is based on finding a track that is matching the observed CDC hits of an event by a transformation from geometric coordinates into a Hough map representing its two-dimensional parameters phi and r. This is shown graphically in figure 2.9. Here, several

active TSs at different SLs are shown in the geometrical X-Y plane. The subsequent Hough transform creates a Hough map that shows all track candidates that geometrically pass through the active TSs. These tracks are represented as single points in the Hough map. The point at which the most TSs are intersecting in the Hough map is assumed to be the best matching track candidate.

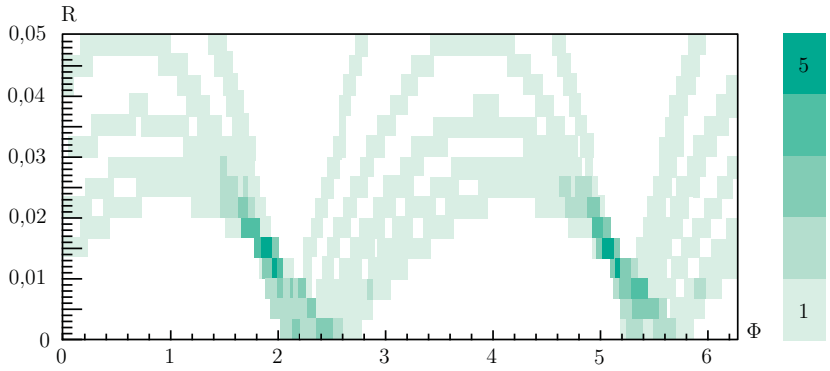


Figure 2.9.: Plot of the Hough map generated by a simulated 2DS [Hoc18]. The two expected tracks are found and represented, dark green, by the two high count intersection points.

For the experiment, this approach was simulated with several physics events. One example is shown in figure 2.9, which shows the resulting Hough map derived from a simulated TSF and 2DS. Two tracks were simulated in this case and both can be found in the shown map as is indicated by the dark green cells. These cells are representing the maximum intersection of all possible track parameters. In this example, the found track parameters are even matching TSs in all five axial SLs. In addition, there is typically not only one single point in the Hough map that represents the maximum intersection, it is rather a cluster of neighbouring track points. This situation motivates the second algorithmic part of the 2DS in which a centre-of-gravity is calculated from the cluster. Instead of only using one cell for estimating the track parameters, several equally important cells are used to estimate the 2D track more accurately. An exemplary distribution of the 2DS parameters that were estimated during operation is shown in figure 2.10. In this, track finding was performed conditionally, that is a track had to match one TS in at least four different SLs in order to be found.

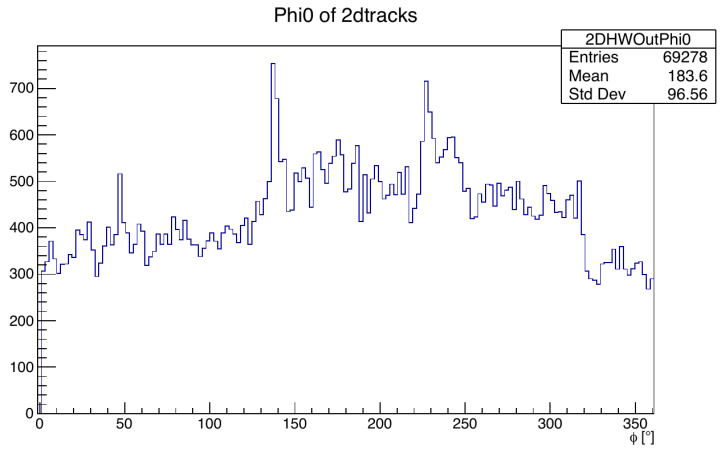


Figure 2.10.: Example of the phi parameters that were estimated for detected 2D tracks during Belle II operation.

2.2.1.4. Estimation of the Event Time

The task of the Event Time Finder is to estimate the point in time at which an event occurred based on CDC data [78]. The event time is approximated by the earliest or the point in time at which the first wire detected activity. However, active wires can also occur due to background events or noise, which would create a wrong event time that is not corresponding to the collision. The core logic of the ETF is to address this issue and is based around distinguishing between wires that are active due to events and those that are active due to background effects in order to find the correct event time. Algorithmically this is solved by considering the number of wires that became active hits within a certain time window. A sudden spike of active wires is hereby considered to be an indication for a background rather than a proper event[2].

2.2.1.5. Global Decision and Reconstruction Logic

The last stage of the L1 trigger system consists of the combination of Global Reconstruction Logic (GRL) and Global Decision Logic (GDL). Both have the overall task of combining all of the individual signals from the different sub-triggers in order to decide whether the buffered detector data is to be read out or not. Hereby, arriving trigger signals are either used to trigger the readout or to suppress it explicitly. Architecturally the GRL is more of a preprocessing stage for the GDL that is locally collecting separate trigger signals. It provides complementing functionality, for example, it prevents repeated trigger

[2] After submission of the first version of the thesis, the ETF underwent a major redesign. That redesign made it more effective, however the functional principles are thus not explained here.

signals based on the same 2D track, which is sometimes sent multiple times with updated TSs. Trigger signals are generating by fullfiling a certain set of conditions, for example an estimation of a track that is sufficiently close to the interaction point[3].

2.2.2. Trigger Systems of Additional Sub-Detectors

The trigger systems described thus far are the most important components in the context of this thesis. They are either directly connected to the developed solutions within this thesis or are generating the data that is required by their hosted algorithms. For the completeness of the consideration of the Belle II trigger system, an overview of the remaining sub-triggers is given in the following.

ECL Trigger System

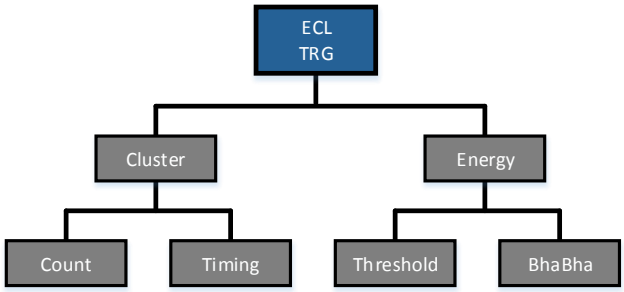


Figure 2.11.: Physics trigger signals generated from the ECL.

Besides the CDC, the data collected by the ECL is of high importance to the L1 trigger system. Complementary to the reconstruction of particle tracks, this detector is used to estimate information about the energy of passing particles. As with the CDC, trigger signals are generated based on information that is derived from this detector. The corresponding sub-trigger consists of two main classes of generated trigger signals. The first class is representing signals based on the energy, while the second class is based on searching for and evaluating isolated clusters of active cells in the detector [21, 66]. Both complement each other by capturing different kinds of important events. The triggers based on the energy are used to detect high electromagnetic energy deposits. Trigger signals based on clusters, on the other hand, focus on low-energy events with multiple hadrons. In addition, the detector is used to provide information about Bhabha events. An overview of the signals that are calculated by the ECL trigger is shown in figure 2.11. As with most of

[3]During the writing of this thesis, the conditions for the developed NNT were defined, explored and tested. The most basic is a cut on the estimated z-Vertex, but more advanced ones are currently explored. For example, a combination of z-Vertex and impulse are used for the so called single track trigger signal.

the other trigger systems, the main difference to the previous ECL trigger system of Belle is the capability to support operation with increased luminosity and event rate.

Programmable hardware such as FPGAs and flash-based ADCs are forming the basis for realizing this system. The readout chain used for this is shown in figure 2.12. The core elements here are clusters consisting of 4x4 crystals within the detector. These clusters are then combined to form trigger cells, which represent the basis for trigger decisions. In total 576 cells are formed from the 8736 available crystals. Energy and timing are then determined for each cell by using a chi square fit, which is implemented in LUTs. As soon as the amplitude of the observed energy is greater than a fixed value and the determined timing is less than a predefined threshold of 88 ns, both energy and timing are passed on for further processing. In total this calculation takes place within 125 ns. Meanwhile, Kintex-7 FPGAs are used for implementation throughout the entire system [54], which are integrated on a custom-manufactured carrier board.

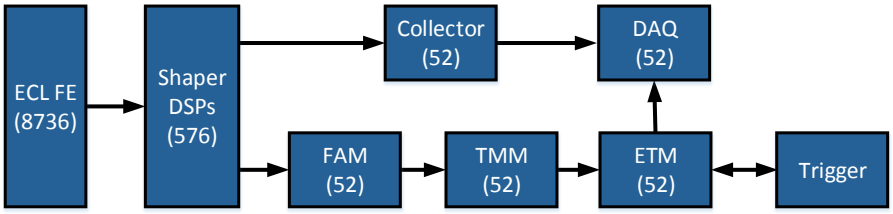


Figure 2.12.: Schematic of the ECL trigger system's architecture [66].

Based on the determined energy and timing, the trigger decision is finally generated on a UT3. The trigger reacts to the total energy or adjacent trigger cells that have reached a pre-defined minimum size. To detect and suppress Bhabha radiation, a new 3D method is used, mainly to correctly estimate low multiplicity events. The total latency of the ECL is currently at about 4 μs, which is well within the requirements of the entire Level (L)1 trigger system. For later configurations of the GDL, the ECL signals were combined with NNT signals to further decrease data rates.

KLM Trigger System

The KLM trigger system was already important during the operation of Belle for the calibration of the detector. For this purpose muon pairs were detected and processed [112]. This trigger system also provided measurements of the efficiency independent of other sub-triggers. The overall system is hereby divided into two parts, representing their geometric location either at the endcap or the barrel of Belle II. As with all of the trigger systems the used hardware platforms are based on FPGAs. In the first processing step, hits within the detector are found and combined across different layers of the respective detector. The parameters of the track are then determined with a chi square fit. Once the found track parameters are close enough to the interaction point, a trigger signal is

sent [71]. An illustration of the generated signals is shown in figure 2.13.

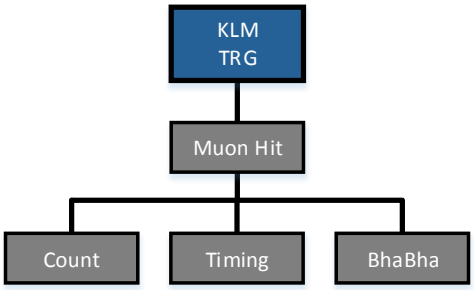


Figure 2.13.: Physics trigger signals generated from the KLM.

TOP Trigger System

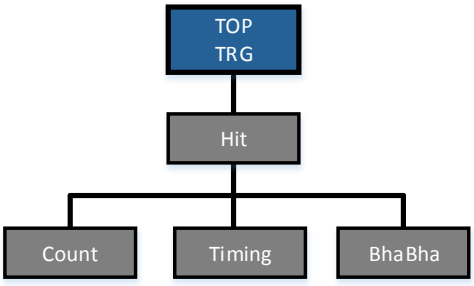


Figure 2.14.: Trigger signals generated by the TOP trigger.

The trigger system of the TOP detector [40] is used to identify particles at the barrel [85]. For this, it is equipped with the ability of recording activity from particles with a time resolution within the nanosecond range. This ability is used to determine the most precise event time across all of the detectors in order to suppress active sensors outside of the time window of the actual event [72]. The recorded data of both the timing and position of observed particles is read out in parallel via optical links from the 16 sectors of the detector and processed at frontend trigger boards [86]. Data streams generated by these boards are then merged at combiner units. In the final processing stage, the global Barrel Particle Identification Detector (BPID), all information is merged to determine the trigger signals. The event time is estimated by first sorting all hits and afterwards calculating their time

deviation relative to a specified reference pixel. Then the time estimation is generated by processing the sorted calculated differences with a set of simulative generated Probability Density Function (PDF). The PDF value is then used to select the best estimate for both time and position. The processing chain is shown in figure 2.15.

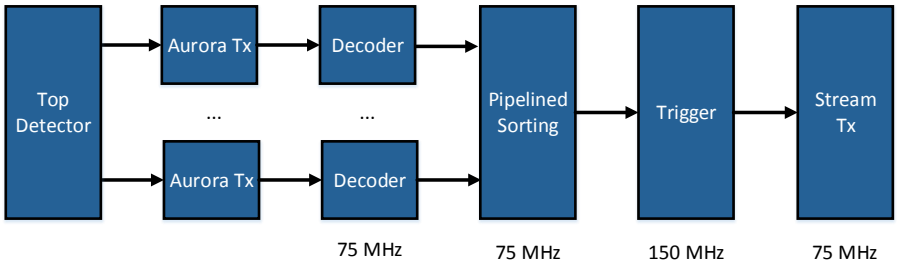


Figure 2.15.: System architecture of the TOP trigger system.

The High Level Trigger System

In contrast to the L1 trigger of the experiment, in which all processing is performed in hardware, the High Level Trigger (HLT) is completely based on software realizations. At this stage of processing, data that was previously selected by the L1 trigger is processed and analysed further using a much higher amount of computing resources and more available latency. Here the HLT is not only responsible for selecting suitable events but also for forwarding the data [67] that matches a reconstructed event. Considering the requirements of the L1 trigger, events are arriving at a rate of 30 kHz at the HLT. Here, each event will have a memory footprint of about 100 kB after the first data reduction. The aim of the HLT is to reduce the outgoing data rate to less than 1 GB/s. This requires a reduction of about 20 to 33%.

The system is deployed into the so-called HLT farm, which is a distributed computing system consisting of multiple processing units. Each unit is in turn divided into work nodes at which the analyses of the experiment's data are carried out. In total, 2000 processor cores are used throughout the HLT farm. The software basis for the processing on work nodes is provided by the Belle Analysis Software Framework 2, which is a unified environment for processing events. It is also used at the offline processing for the analysis of the data and validation of, for example, the trigger components. For efficient and fast processing within the HLT farm, the framework was optimized for parallel execution.

For trigger systems the HLT is espescially interesting for offline comparisons. I can be used as a reference to compare efficiencies and resolutions, since they are much more accurate than their online counterparts. On the other hand they make life more difficult for L1 trigger systems since they filter out even more data. That means that many of the events processed by the NNT for example are filtered here and not available for additional analysis. This can be ciurcumvented by requesting special runs with less effective HLT filtering.

2.3. Belle II Data Acquisition System

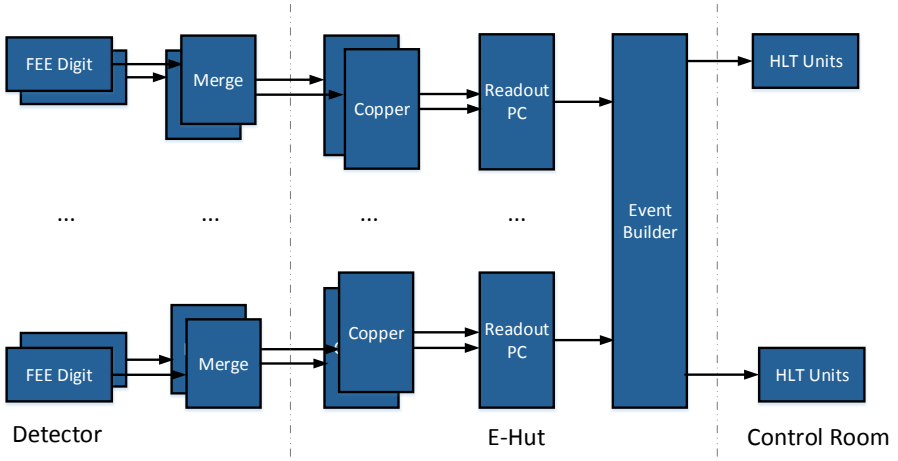


Figure 2.16.: System architecture of the Belle II DAQ.

The main task of the DAQ system developed for Belle II is to transfer the experiment's data from the FEE to offline storage. The system has hereby to support the different implementations of each sub-detector's readout-chain, which have their very own requirements due to their heterogeneous design. At the same time, the DAQ is controlled by the L1 trigger, which decides over the data that is to be sent to offline storage. An overview of the DAQ's system architecture is shown in figure 2.16. Overall it can be divided into three separate sections that are defined by their location relative to the detector. The first section encompasses all of the readout and signal digitization electronics located directly at the detector. Online data processing consisting of data merging, formatting and reduction is located at the Electronics Hut (E-Hut) section. The final processing is performed at the HLT farm in which software-based events are selected. It is part of the control room section. The interface between the detector section and the E-Hut is hereby established via custom hardware, the COPPER boards. These are additionally tasked with locally summarizing event data. This data is then sent to the Readout Personal Computer (PC)s, which perform a more comprehensive data reduction before sending the data to the Event Builder and the HLT.

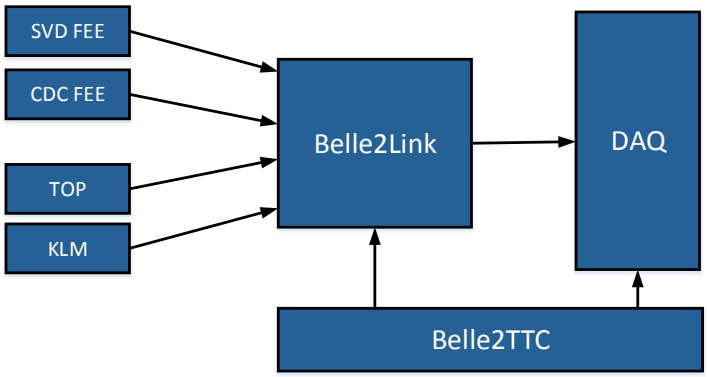


Figure 2.17.: The architecture of the detector's readout scheme that is based on the unified interface Belle 2 Link.

The Belle 2 Link (B2L) [81, 106] represents the basic means of communication between the FEE and E-Hut. In general, it defines a unified interface for the transportation of data and the integration of trigger signals into the overall data flow. Its main challenges and requirements are to provide support for all the different kinds of FEE that are used across all of the several detectors and to establish reliable data transfer. To ensure reliable transfer, the B2L is designed to provide soft error mitigation services [33]. Flexibility is meanwhile achieved by a configurable design that allows the tailoring of communication to the requirements of each type of FEE. In addition to that, it is designed to achieve high data transmission rates to support the anticipated massive data rates. Communication from the FEE is realized using RocketIO Gigabit Transceivers (GT)P [117] based optical data transmission. In addition to the data links, the DAQ is responsible for the distribution of the clocking and Slow Control signals to all sub-systems. This is especially necessary for components of the CDC, BPID and EPID since those are not easily accessible as maintenance cycles are far apart. For the data transfer to the E-Hut a First-In First-Out Memory (FIFO)-based scheme is used within the COPPER modules, additionally, it is also coupled to the timing system of the trigger system [80]. The role of the B2L as a unified communication infrastructure within the DAQ is illustrated in figure 2.17.

Detector	PXD	SVD	CDC	TOP	ECL	ARICH	KLM
Channels	800.000	223.744	14.336	8.192	8.736	65.664	35.808
Event size [kB]	800	14,9	24	9,2	29,6	15,5	7,5
Data rate [MB/s]	2.604	447	720	276	888	465	225

Table 2.3.: Overview of the data readout properties of all sub-detectors in Belle II.

The hardware architecture of communication based on B2L is shown in figure 2.18. It consists of a transmitter and receiver pair, whereby both are requiring the corresponding modules for controlling the GTP transceivers and FIFOs to bridge the interface to the data processing. The trigger timing is meanwhile controlled via the Trigger and Timing Control (TTC) module. Required Intellectual Property (IP) cores are developed within the groups of the Belle II collaboration and provided as Very High Speed Integrated Circuit Hardware Description Language (VHDL) source code. These cores form a library that enables B2L communication for supported FPGAs such as the UT3.

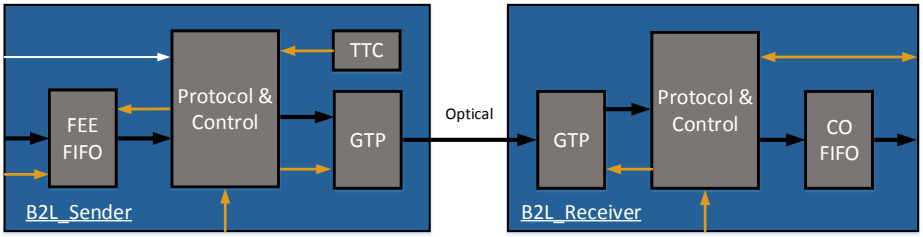


Figure 2.18.: Architecture of the Belle 2 Link sender and receiver pairs. Coloured arrows indicate data (black), timing (gold) or status (white) information.

COPPER platforms, used as part of the B2L, were developed at KEK [38] and were already used at the Belle experiment. Here, they were used to enable a transfer between the frontend and the event builder. In Belle II, the detector readout is realized by using additional extension boards called FINESSE boards. These boards are mainly sampling and digitizing the detector signals, which are then stored in a buffer memory. As soon as the trigger signal arrives, the data is sent to the FIFOs that connect the FINESSE boards to the COPPER boards. Up to four FINESSE boards can be mounted simultaneously on one COPPER board. Each COPPER board is additionally equipped with a CPU, which is primarily used for data merging, reduction, and formatting. The individual data streams generated across the entire DAQ, using the B2L, can be addressed via an identification tag assigned to each COPPER and FINESSE board. With this addressing, each outgoing data streams can be traced back to their source, which is necessary for the subsequent analysis. This addressing scheme is also used to identify the B2L data streams generated by individual trigger components. The trigger systems developed throughout this thesis use this path to send status information for Data Quality Monitoring that is used for ensuring the correct operation of the hardware. This aspect is described in more detail in section 5.3.4, for example, for the NNT.

2.3.1. Data Reduction for the Vertex Detector

The PXD is strongly influenced by the increased luminosity as it is positioned the closest of all detectors to the interaction point. The attached DAQ is designed to be capable to transmit about 10% of the PXD's data when assuming the maximum occupancy for its sensors. Bridging the resulting gap between the incoming and outgoing data rates is the task of the employed online data reduction mechanisms. From the physics point of view,

a mechanism based on lossless compression is desired. During preliminary studies, no algorithm was found that could achieve the required reduction without any loss of data. The most promising and subsequently chosen approach is to limit the number of pixels to be read out and stored. This is supported by the fraction of the expected occupation that is caused by background events rather than collisions. Using simulated events, it was shown that most of the active pixels are caused by such background [96] events. Thus, preventing these pixels from being read out will enable a substantial reduction of the data rate without losing the fraction of data that is actually relevant for the experiment.

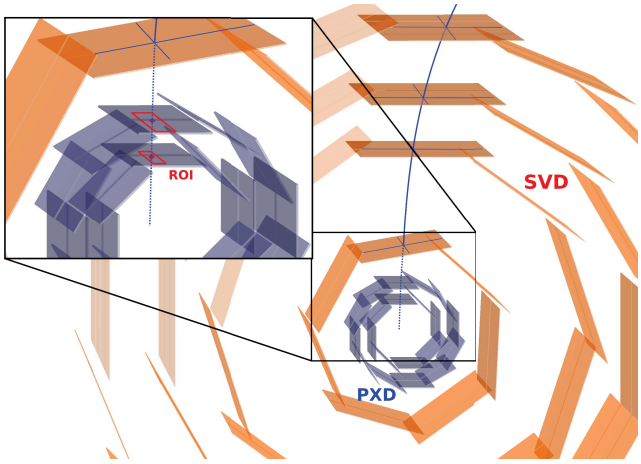


Figure 2.19.: Graphical representation of the Region of Interest approach used for data reduction at the PXD. At first, particle tracks are estimated solely by using data from the SVD. These tracks are then extrapolated to the PXD. Regions are formed around the intersection of the PXD and the extrapolated track as shown by the red areas. Only the pixel data within these regions is kept for subsequent processing [96].

The mechanism that is used for addressing this problem is based on correctly detecting pixel clusters within the PXD that are belonging to physics events and suppressing the remaining clusters. Rather than processing solely pixel data, the approach is relying on the usage of the SVD. As this detector is located further outside of the interaction point, it is less influenced by the increase of the luminosity compared to the PXD and thus generates lower data rates to be processed. At the same time, the sensors used within this detector are supporting much faster readout compared to the rather slow readout of the PXD's DEPFET sensors.

The identification of pixel clusters that are to be reduced is performed by estimating a particle's trajectory using the SVD's data. Algorithmically this is achieved by using a tack finding approach based on the Hough transform that is tailored to the specifics of the detector. The estimated track is then extrapolated into the PXD. Using this extrapolation, a Region of Interest (RoI) is defined at each intersection between the estimated track and

the layers of the PXD. This RoI represents are predefined area consisting of pixels around the intersection point. Here, the assumption is that all active pixels within this area are belonging to the track that was observed in the SVD. Since most of the particles that are produced by background events are not possessing enough momentum to escape the first few layers of the detector and they will not have the a minimum momentum in order to reach of SVD layers. Due to this, this method is providing an effective mechanism to distinguish background from signal within the PXD. The RoI approach itself is not unique to Belle II but is a rather well-known data suppression technique. It is hereby custom-tailored to the requirements of Belle II. From an integration perspective, the data reduction based on RoIs is implemented on FPGAs in order to achieve the required latency. A graphical representation of the mechanism is shown in figure 2.19.

While being an effective mechanism to achieve data reduction, it is introducing a major drawback in requiring particles to reach a predefined number of layers in order to be considered for the definition of RoIs. A complementing solution that can improve the signal efficiency for particles not fulfiling this requirement is the focus of chapter 8 of this thesis. As this approach is going to be integrated into the already established system architecture, it is shown in figure 2.20.

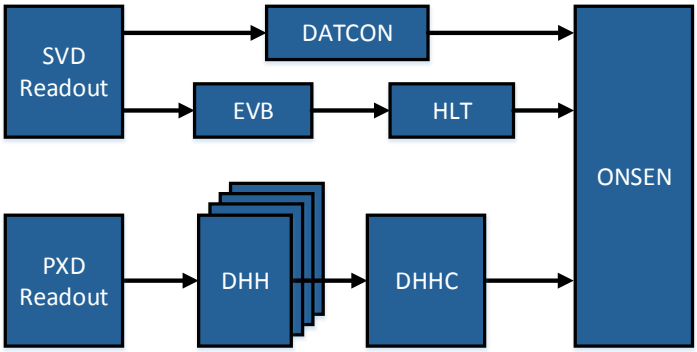


Figure 2.20.: Architecture of the RoI-based data reduction system for the VXD.

2.3.2. Signal and Background Events

In order to carry out an efficient suppression of background events, it is at first necessary to investigate and understand which particles are actually responsible for their occurrence. A detailed analysis of the occurring types of background events and their effect on the experiment was conducted in preliminary studies, for this thesis the studies performed in Ref. [93] and Ref. [92] are the most relevant and the basis for this section.

The types of background can be partitioned into four classes, which are Touschek effect, two-photon Quantum Electrodynamics (QED), Beam-gas Coulomb scattering and Radia-

tive Bhabha. Background related to the Touschek effect describes particles generated due to the interactions between separate particles that are accelerated and directed towards collision together in a particle cloud as part of the same beam. The results of these interactions are background particles with a momentum significantly different from the particles within the bunches and signal events. The occurrence of this effect is meanwhile inversely proportional to the energy of the respective beam. For Belle II this effect is thus more pronounced in the lower energy electron beam. The resulting events that are visible within the detector, are anticipated to be mostly produced by particles leaving the bunch and interacting with the beam pipe. The experiment is impacted by that in the form of an increased amount of particle tracks that have an origin outside of the interaction point. Track trigger mechanisms such as presented in chapter 5, are tasked with suppressing this kind of background. Another major source of background events is represented by the two-photon QED effect. This describes particles that are generated by the interaction of an electron and a positron together with two photons. Opposite to the Touschek effect, this background will have its geometrical origin around the interaction point, which will make it difficult to identify and suppress by tracking alone. However, these particles are predicted to reach only low energie, which can be used for separation from signal events that have higher energies. While much effort is put into establishing a vacuum within the beam pipe there are still some impurities left that can affect the experiment. Accelerated particles may interact with present residue resulting in new particles that can be observed within the detector. Currently, there is no estimation of their contribution to the experiment. Radiative Bhabha, meanwhile, describes an effect that is often present in asymmetric electron-positron colliders such as Belle II. It manifests itself in the form of newly produced electron-positron pairs or generates a photon that interacts with the detector. However, only a few will reach even as far as the PXD, thus they are not expected to be of major concern for operation.

2.3.3. Slow Control and Data Quality Monitoring

The control systems Slow Control (SC) and Data Quality Monitoring (DQM) are two of the most important pillars of operation for the experiment. Both have the task of monitoring and controlling the data readout and overall processing. The SC describes the concept with which distributed control systems within the entire detector's operation are controlled and monitored. For each sub-system of the experiment, information about their current state is recorded during operation. This is not only restricted to the detector and its electronics but also performed for the accelerator itself. Here the information is recorded in the range of seconds. Compared to the actual readout of the detector, this is rather slow, which inspired the naming for these control systems.

Examples for monitored status variables of an experiment are temperatures of a detector's components, voltage supplies but also about the readout electronics [29] and observed background of the accelerator. Especially the latter shows how the information from the SC can be used across different parts of the experiment. The loaded configuration of the trigger system, for example, the neural network loaded for the NNT from section 5 can be based on the recorded background and behaviour of the accelerator. As a consequence, the SC is also in some part responsible for loading suitable configurations of the sub-

systems depending on the state of the experiment.

Slow Control at Belle II

The Slow Control (SC) is performed throughout the different heterogeneous components of the experiment and must thus be easily adaptable. Although some of the basic concepts behind the readout are uniform within Belle II, for example, the readout via B2L, most decisions regarding the choice of hardware and software are typically left to the sub-system's developer. Since the control data is merged, for example, to make decisions about the overall configuration of the experiment, it must be ensured that the communication between all systems is synchronized. This is one of the most important requirements for the implementation of such systems.

One of the main functionalities to be provided by the SC in Belle II is to allow the determination of the current state of the experiment across all sub-systems. The experiment's state is hereby divided into two parts. That is the readiness for operation of the accelerator and the data readout electronics. At first, the operational readiness of the accelerator is determined and monitored. If it is ready for operation, for example, by reaching a desired level of luminosity, the targeted configuration for the readout electronics is selected and loaded on the basis of multiple present state variables. When both parts, accelerators, and readout are ready, the experiment is started. The entire process, including all state transitions, is called run control. A master control process is performed in parallel, which is responsible for the configuration of the individual sub-systems, for example, the sequence in which individual steps are carried out.

The systems described and developed as part of this thesis must also provide the appropriate interfaces and mechanisms for the monitored operation using SC. In the Belle II experiment, different implementations of the SC are used across the different sub-systems. The reason for this heterogeneity is the special treatment of the VXD. For this sub-detector, The Experimental Physics and Industrial Control System (EPICS) [89, 87] was chosen. It is an established solution used in many modern particle physics experiments throughout the world. For the remaining detectors such as the CDC, a solution developed at KEK called Network shared memory (NSM) [75] is used. This solution was already used in the predecessor Belle [82] but was functionally extended for operation at Belle II. The approach used in the trigger system is of particular interest in the context of this thesis since all developed systems within this thesis with a need for SC are part of it.

The SC approach used for the trigger consists of three components [51]. These are the user module, the condition database, and the archiver. The task of the user module is to provide a comfortable interface for the administration and supervision of the status. It has to allow a person who is not an expert of a specific sub-system to control it without getting too involved in the details. The used solution for the user module is hereby uniform for all components of Belle II.

Both the archiver and database are of greater interest as well. The database is logging the current status of all connected components during operation. An example for the NNT is the currently loaded neural network. It is, however, not only logging the state of a component, but also configuring it. In the context of the NNT, new weights of a neural network can be loaded on command. The scheme used within the trigger system is based on an

NSM process that is executed on a local computer with access to the readout electronics, database, and the archiver. This process is then able to record all the relevant parameters, communicate with the database and control the respective components. The structure of this scheme is shown in figure 2.21. Hereby the structure is partitioned into two separate networks, the local trigger network and the DAQ. Access to the hardware only takes place within the local trigger network from a computer specified for this purpose.

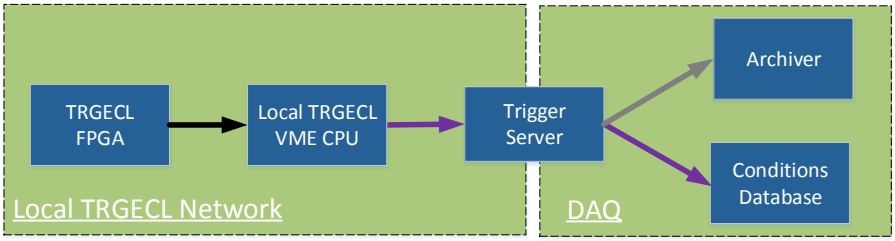


Figure 2.21.: Architecture of the slow control used within the ECL trigger, that represents the reference implementation for all sub-triggers.

The archiver has the task of recording data in real-time, especially results, from the experiment. Like the condition database, it is located outside of the local network. It is hereby interesting that the data exchange is not based on NSM, but on EPICS. In the approach used within the trigger system, the archiver is not used directly with EPICS. Instead, the same NSM process that is used for the database is used to read the required data from the hardware. The data is then transferred to the trigger control server, at which it is converted locally from NSM to EPICS. The converted data is afterwards sent to the server hosting the archiver. Not only trigger data is stored here, but also data from accelerator data such as the history of the recorded beam noise.

The distributed structure of the SC across all related readout crates is shown in figure 2.22 based on Ref. [55].. On the one hand, all of the creates hosting trigger hardware are shown here, on the other, the respective responsible SC process is shown as well. The processes are additionally distinguished by their type of implementation into EPICS and NSM. The used hardware is based on the Versa Module Eurocard bus (VME)-based crates. These crates are equipped with a V7865 Single Board Computer (SBC) [1], which acts as VME master and host of trigger hardware. Each VME crate is equipped with a backplane that is used to mount several trigger boards at once. The master CPU is then used to communicate and connect all boards. Meanwhile, the SC processes are executed for all hosted trigger components in parallel on this master CPU. Individual trigger components are distributed across several different crates, for example, the NNT is hosted on the crate named vmetrg16 in the shown configuration. The Belle II Trigger Server (BTRGSRV) is located at the interface of the global network. Processes executed on this computer are collecting the data across all crates. For the interaction with the archiver, the collected data is converted to EPICS while data for the database is kept natively in NSM.

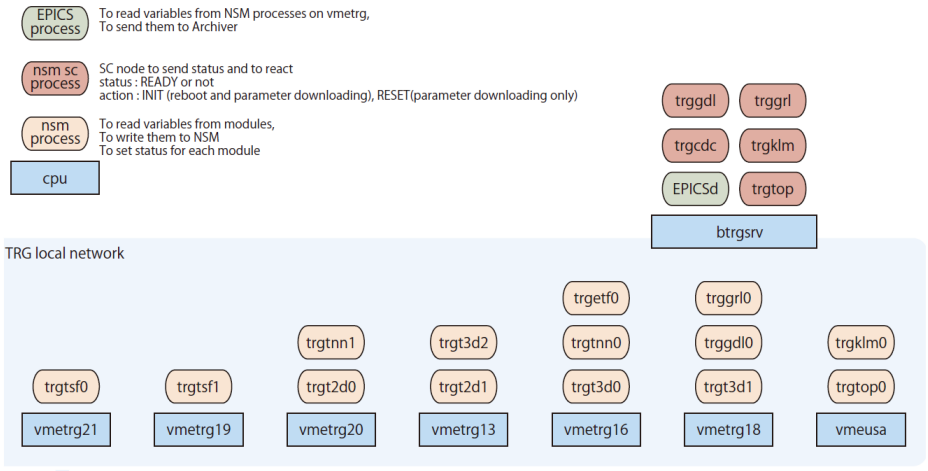


Figure 2.22.: Structural description of the slow control setup across all sub-triggers [55]. It shows the present processes colour-coded by their type together with their hosting computing platforms.

Data Quality Monitoring in Belle II

The task of the DQM of Belle II is to ensure correctness of operation by analysing samples of events online [59]. This is typically done by generating histograms from recorded sample data. These histograms are then updated in ten-second intervals. The method used at Belle II is mainly based on BASF2 [77]. Here, there are two options for the experiment's operation. The first option is to run DQM within the HLT farm. At this stage, PXD data is not available, so only part of the total data of an event can be analysed. The second option, the Express Reconstruction, is another computing farm that can perform online monitoring with data from all detectors as well as the HLT. To use DQM, data must be transferred from local hardware over the B2L. The DQM is important for the systems developed within this thesis, as it is the best method to check the comprehensive correctness of the data processing hosted by the hardware.

2.4. Field Programmable Gate Arrays

The target technology used for all concepts within this thesis are Static Random Memory Access (SRAM)-based FPGAs. These are often used for the implementation of trigger and DAQ systems in modern particle detector experiments. Their capability to configure functionality even after manufacturing is the most important characteristic as these systems can be adjusted flexibly even after integration into the detector's readout chain. Trigger and DAQ hardware are usually difficult to change during operation, especially regarding their connection to the detector. At the same time, the real behaviour of the accelerator is not known during the development of the planned data processing systems. First studies

are hereby relying on simulations, however the methods developed on this basis have to be adapted to the observed behaviour during operation with collisions. Here, the programmability of the FPGAs can be exploited. Furthermore, FPGAs are providing a large number of IOs, which are necessary due to the large amount of data to be transferred. The functional basis of FPGA processing will be discussed in the following.

Logic operations on FPGAs are performed by flexible and generic basic processing elements. These elements mainly are programmed to store the desired function.There are multiple technological and architectural implementations for these elements. Possible technologies that are available on the market are SRAM, Multiplexer, Flash, and Antifuse. Since one of the goals of this thesis is to achieve low latency, the internal structure must be known in order to understand and optimize the internal delays.

The most commonly used FPGAs are based on LUTs with SRAM. They are also the chosen target technology throughout this thesis. Compared to the other technologies, they are offering the largest amount of logic resources for implementation, on the one hand, and the shortest signal propagation delays on the other. Since this kind of FPGA is representing the biggest and most important market for the manufacturers, they are typically also the ones to be refreshed with the newest available semiconductor technologies as soon as possible. For example, the latest product family from the market leader Xilinx, the Ultrascale+ FPGA, is realized with 16 *nm* FinFET [136] [132] transistors. Both signal delays and resources are highly important to the trigger designs discussed within this thesis, as their respective applications require low-latency processing and fixed throughput at the same time. A smaller technology node is significantly impacting the outlook for a successful realization. At the same time, the systems to be developed are also quite resource-hungry, especially in terms of their demand for connectivity. Higher amounts of resources are beneficial to hosting larger and more powerful neural networks. The advantages provided by base elements alternative to SRAM that are available for FPGAs are meanwhile not providing significant advantages for the considered use cases. Characteristics such as high resistance to radiation or persistence of configuration after restarts, are desired features but only of secondary priority. The systems designed here are generally located outside of the areas near the detector that experiences a high dose of radiation. At the same time, the complete electronics are reset in case sub-systems are restarted.

2.4.1. Application Domains

FPGAs generally have a broad application range since they are providing a highly flexible platform that can implement any logic function. At first, their main use case was to serve as a system to bridge communication between two separated systems with heterogeneous and incompatible interfaces. Their application then shifted into domains with demand for adjustment of its functionality even after their integration into the application environment. Coupled with them providing a large array of powerful IO resources, this is the main reason why they are such a popular choice in modern particle accelerator experiments. Nowadays their use is much broader and pronounced in domains that were traditionally relying on general purpose processing. Due to their high efficiency regarding the trade-off between power and performance they are a popular choice in large scale data centres such as Amazon Web Services. At the same time, FPGAs are found in many

embedded domains, such as avionics, or are even used as high-quality ADC for premium headphones.

2.4.2. General Architecture

The processing core within FPGAs are Configurable Logic Block (CLB)s. These are the common structure used to implement an arbitrary boolean function for a fixed number of binary inputs. For the realization of more complex functions, several CLBs are configured and combined by routing signals together. For this purpose, flexible connection structures, the switching matrices, are present on an FPGA. To allow data transfers to external components, a large number of IO blocks are included, these provide drivers and support different standards. These blocks are of particular interest for trigger components as they have diverse communication requirements depending on the respective interface.

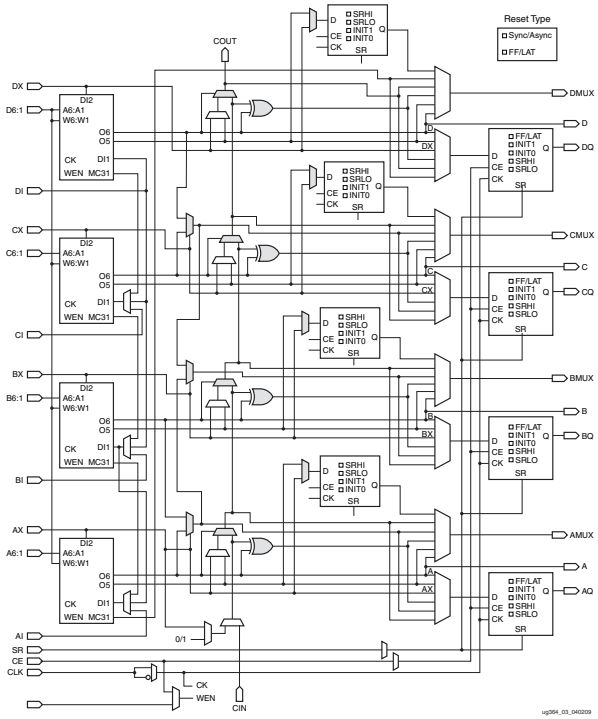


Figure 2.23.: Architecture of SliceM logic resources used at Virtex-6 FPGAs [125].

In purely functional terms, an FPGA is capable of implementing any possible logical function through its CLBs. The actual feasibility, however, depends on the available resources and timing requirements. Since CLBs are realized with very fine granularity on an FPGA,

few bits per block, more complex functions have to be distributed across several CLBs and combined using the routing resources. Distributing logic functionality across several blocks meanwhile introduces additional signal propagation delays. At the same time, the number of routing resources is limited. In case that internal channels are completely occupied, alternative routing paths must be found. These paths can increase the longest runtime thus decreasing the clock frequency. In addition, some operations cannot be implemented immediately and efficiently in LUTs. An example for this are multiplications with variable bit widths, as they are predominately occurring within the processing of general artificial neurons. Since multiplications are required by many applications, modern FPGA architectures have been extended to include additional multiplication processing elements that achieve better characteristics.

Configurable Logic Blocks

CLBs used in Xilinx FPGAs have their own internal architecture, which is in turn consisting of additional smaller processing units. They consist of so-called Slices [125]. There are two possible types present in a CLB, SliceL and SliceM, with the biggest difference being that SliceM can be used as distributed Random Memory Access (RAM) or as shift registers in addition to serving as fine grain logic units. Meanwhile, SliceL does not support any write functionality for the LUTs and is used as constant logic. The difference is especially important when analysing the implementation results. Since the SliceM type is providing additional functionality its architecture is shown in figure 2.23 as a representative for both types. Within this architecture, the first stage is consisting of the LUTs, in this case four LUTs are part of one Slice. The LUTs are followed by a carry chain stage, which allows for efficient transport of intermediate results towards directly adjacent Slices. This is performed without the need for using the external routing resources, in turn reducing the latency. This stage is followed up by the storage stage, which consists of multiple flip flops for optinal data buffering.

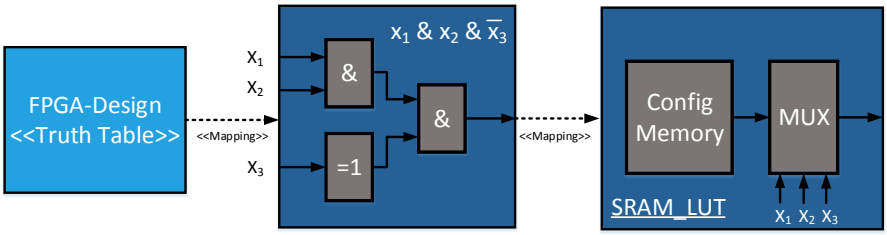


Figure 2.24.: Mapping of a logic function into LUTs of FPGAs.

A logic function is implemented at the LUT stage. Here, the LUTs internally consist of a configuration memory that contains the results of the function and a multiplexer that selects the configured value according to the input value. The contents of the configuration memory are determined at design time. The realization of a boolean function in a LUT is exemplary shown in figure 2.24. The content of the configuration memory can be hereby

at the designer's choice, it only has to comply with the limitations of the fixed input and output ports.

In order to implement larger logic functions, individual Slices are interconnected via dedicated routing resources. Since the interaction of the Slices within an application is not known, these resources are implemented to be highly configurable and flexible even after manufacturing. The architectural unit that provides this feature is the switching matrix. In a switching matrix, a number of inputs can be connected to multiple outputs at intersection points. This is implemented by predefined configuration memories within the intersection points. For a four-terminal intersection, every possible connection path is controlled by transistors. Whether a transistor is passing a signal through is encoded in a connected memory cell.

Specialised Processing Resources

The resources described thus far are already sufficient to flexibly realize all possible logic functions. However, modern FPGAs benefit from the presence of dedicated processing resources that can perform certain tasks more efficiently. The commonly used Block Random Accessible Memory (BRAM) and Digital Signal Processor (DSP) elements are hereby of particular interest, which will be further discussed in the following.

BRAMs are dedicated memory blocks on an FPGA. As an example, Virtex-6 FPGAs are providing a dedicated BRAM module for storing 36 Kb of data together with two parallel ports for read and write access [126]. They are a much more efficient realization for large memory storage compared to the alternative implementation in CLBs. The main drawback, however, is their lower capability for parallel access within the memory. Only two memory locations can be queried in parallel at any given clock cycle. A second important drawback is their static placement on the FPGA as this is making routing of signals more difficult. This trade-off will be considered throughout the realizations presented within this thesis since the algorithms that are considered here strongly rely on parallel access for retrieving constants. The importance of BRAMs for FPGA applications can also be demonstrated by the number of resources provided in modern architectures. Table 2.4 shows the development across several technology generations taken from the product tables [127, 130, 128, 133] of Xilinx. Especially today's big data applications are driving the increased focused on the integration of such memory as they require large amounts of quickly accessible storage.

Multiplications are meanwhile one of the most common operations in digital signal processing and neural network applications. Although LUTs can realize these very efficiently in terms of resources and latency when at least one input is a constant of fewer than 10 bits, this implementation does not scale well otherwise. In order to allow optimization for variable input bit widths, modern FPGAs are equipped with additional units to provide efficient multiplications. In modern Xilinx FPGAs, these units are called DSP slices, whereby they can also be used for divisions or Single Input Multiple Data (SIMD) operations. The processing core of this DSP is a multiplier with fixed maximum bit widths for the input ports. For example, a Virtex-6 FPGA is using two input ports that support 25 and 18 bits respectively. To achieve high clock frequencies, DSPs are additionally equipped multiple register stages to facilitate pipelining and cascading of operations. Here, cascading takes advantage of the fact that DSP slices are placed on an FPGA in columns that allow them

to be directly adjacent to each other. Instead of communicating with each other via the common method of using external switching matrices, a special cascading infrastructure is used. This allows multiple DSPs to be connected in series to perform their processing with a high clock frequency, which is especially efficient for Multiply and Accumulate (MAC) operations. Due to this, these units are of particular interest in the implementation of neural networks. An investigation for this is carried out as part of section 4.4.1.

IO Resources

Another highly beneficial feature of FPGAs is the broad and diverse availability of provided IO resources. For this purpose, modern FPGAs are, for example, equipped with a large number of fast serial GTs. The most important ones used in this thesis are GTX [123], GTH [122] and GTY [129] which differ in their maximum data rates.

Resource	XC6VHX565T	XC7VX1140T	XCVU190	VU13P
Slices	566,784	1,139,200	1,074,240	1,728,000
DSPs	864	3,360	1,800	12,288
BRAM	32.832 Mb	67.680 Mb	132.9 Mb	454.5 Mb
GTX	48	-	-	-
GTH	24	96	60	-
GTY	-	-	60	128

Table 2.4.: Development of resource availability in FPGAs across the latest generations.

Clocking Resources

FPGAs are additionally equipped with adjustable clocking resources. They enable the usage of multiple different clock domains across the chip. This is important to generate a targeted frequency for data processing but is also vital to operating IO resources. At those interfaces, a corresponding synchronization must take place. For the generation, distribution and adaptation of the clock an FPGA is equipped with PLL modules, drivers and dedicated routing resources.

2.4.3. Design Flow for FPGAs

Realizations of firmware for FPGAs can be described in high-level languages such as C, C++, Python, Matlab and more by using High-Level Synthesis (HLS) tools that are translating a description into a synthesizable format.

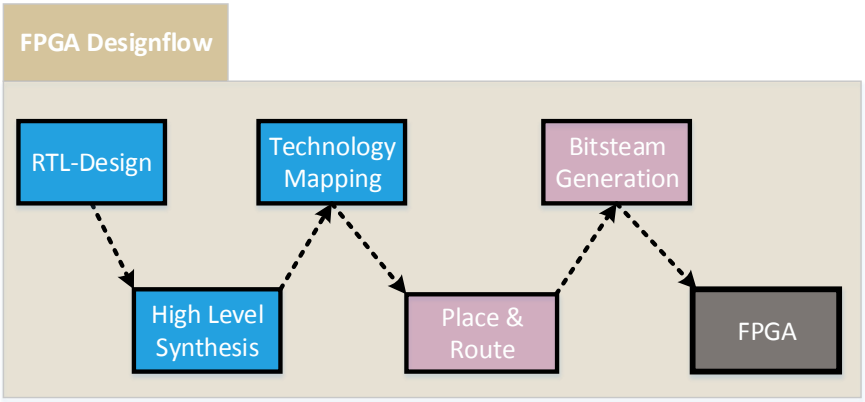


Figure 2.25.: Typical design flow for the development of FPGA designs.

The more traditional way is to describe the realization in an HDL such as VHDL, Verilog or System Verilog. These languages were specifically developed to describe digital circuits. At first, they were used to model and simulate already realized circuits for easier verification, but were over the years picked up by Electronic Design Automation (EDA) tool vendors to automatically translate abstract descriptions into logic-level descriptions of the circuits to be realized. In contrast to the high-level languages, an HDL-based description is typically performed at the Register Transfer Level (RTL), although more abstract descriptions are also supported by tools nowadays. However, in order to reduce the degrees of freedom for the tools and thus to optimize the design manually, RTL is often still representing the best choice.

A description of functionality is typically first checked for correctness. For this, a simulation tool like Modelsim [73] or Xilinx ISIM [124] can be used. With these tools, circuits are examined time-efficiently and in detail before they are implemented. In addition, most simulation tools are offering interfaces for coupling hardware-focused simulation with applications written in C++ or Matlab. With this option, it is possible to use an already available software reference implementation for validation. This is of particular interest in the context of physics experiments since reference implementations of algorithms are typically implemented on a software-level before transitioning to hardware.

The FPGA's functionality is then brought to the device in the form of so-called bitstreams. These bitstreams are packing together the contents for the entire configuration memory within the LUTs, switching matrices and remaining resources. To generate this, first a synthesis is performed. Here the described logic on RTL is converted into a functionally equivalent description on the level of basic logic gates. Using these gates, it is already possible to roughly estimate vital characteristics such as signal propagation times, energy consumption and more. With the help of these characteristics, an FPGA designer can already estimate the feasibility of his design.

The synthesis is followed up by the technology mapping step. In this step, the logic description of the circuit consisting of the respective gates will be mapped onto the basic elements provided by the target technology of the device. In devices by Xilinx, for example, those elements are predefined primitives and macro-cells. These are in turn representations of the previously introduced processing elements on an FPGA such as DSP slices, individual logic Slices or BRAMs. In this phase, the feasibility of the implementation is also examined on a rule-based basis. An example of this is a check for combinatorial loops, for example within a DSP at which a selected operation mode might require the usage of certain dedicated registers. The success of this check can also depend on the selected optimization strategy. Retiming, for example, can result in registers being rearranged in such a way that these constraints are violated.

The physical design is then performed by using the representation of the circuit as basic elements generated from the technology mapping. The core of this step is the placement and routing within the structures available on the chip. The selected elements are placed or mapped onto concrete elements located on the FPGA and connected to the switching matrices. The last step of the conversion is then the generation of the bitstream with which the FPGA can be configured.

The complete design flow for the implementation for FPGAs is additionally shown in figure 2.25. The steps that are typically associated with high-level synthesis are shown in blue while those associated with low-level synthesis are shown in pink.

2.4.4. Design Space Exploration

An RTL description can be physically realized on an FPGA in many different ways, for example, with different placements of modules or routing paths on the device. Depending on these, different timing and resource characteristics are achieved for a design. Meanwhile, the FPGA designs developed within this thesis have to fulfil real-time requirements in terms of latency and throughput. These can often only be met by exploring many different designs. An overview of the optimization options provided by the used EDA tools is given in the following.

Design space exploration can already be carried out at the synthesis stage, for example, by including more details of the physical design. Additionally, different tools are available, especially synthesis tools by Synopsys [107] among others can be used as an alternative to the tools of the FPGA vendor. For the developments of this thesis, the tools by Synopsys were consistently able to create netlists that required fewer resources and simplified the timing closure process.

For the placement and routing as well as the technology mapping, tool manufacturers are providing the most implementation choices to be explorred. For example, Xilinx is offering a set of strategies that are divided into three groups optimizing either timing, power or routing congestion. Power is less important for the considered trigger systems, though they are gaining in importance for future applications. The other two strategies are very important and have been explored throughout, as they severely impact the feasibility.

One of the possible reasons for not reaching the desired frequency goal are regions on the FPGA in which a high fraction of the available pins and channels are locally occupied by

the transportation of a multitude of signals. With proper selection of suitable strategies in addition to floorplanning, the tools can be directed towards avoiding these problems at both the placement and routing stage. The placement strategy hereby tries to find a compromise between timing, wire length and channel utilization. The router then tries to route networks through the less frequently used channels as much as possible, even if they result in worse delay for single paths. Design problems caused by congestion hereby often do not benefit from introducing additional register levels. This rather makes the problem even worse by introducing additional signals to be routed. Instead, optimizations of the resource utilization can lead to a significant improvement in timing by mitigating the congestion problems introducing fewer signals to be routed.

The options that proved to be the most influential for the developments within this thesis are listed in table 2.5. However, FPGA designers do not have to select these options themselves during development. Xilinx already provides an assisting tool for this, the SmartXplorer [104]. This tool selects suitable strategies for one of the three optimization goals independently and starts implementation runs for all. Based on the results achieved by the initial runs, the tools are then independently selecting the best options and vary the cost tables to create better implementation results.

Name	Description	Effect
Packing	Degree of combining logic into one CLB	Less resources, worse timing
LUT Combining	Usage of dual-port options for merging of functionality	Less resources
Logic Opt	Resynthesis and restructuring	Better timing
Register Duplication	Duplication of equivalent registers	Better timing
Retiming	Movement of registers across logic	Better timing
Cost tables	Predefined cost table guiding the tools	Better resources and timing
Timing-Driven	Placement is performed at an early stage	Better timing

Table 2.5.: Most influential tool options for optimizing the physical implementation.

2.4.5. High-Level Synthesis

The majority of development for FPGAs is currently still performed at the RTL. The approach to abstract from this and use higher abstraction levels is a long-time research topic and is supported by High-Level Synthesis (HLS). The advantage of this is that the tasks of scheduling, binding, and allocation are all performed automated by tools. This automation is increasing productivity by freeing up time for the developer. Most FPGA

manufacturers are currently providing appropriate tools to support this approach. Since this thesis is strongly based on the usage of FPGAs by Xilinx, more emphasis is put on their tools, in particular, Vivado HLS.

Vivado HLS is mainly based on the usage of an algorithmic description using a subset of C++ instead of, for example, VHDL. From this description, the tool generates an RTL architecture. Opposite to modern design guidelines, hardware modules created this way are not intended for further development or manual analysis, they are rather to be used as black boxes. Internal structures are described without regard to human developers and are thus difficult to understand or modify. As manual analysis is almost impossible, these modules are typically validated by using HW/SW Co-simulation. Here, both the high-level C++ and the generated RTL description are simulated together and the results are compared for correctness. As the high-level description is basically standard software, any kind of test infrastructure can be used and interfaced with the hardware implementation, allowing the usage of mature concepts such as unit tests. With regard to this thesis's main application, physics experiments, it is possible to interface frameworks developed for detector simulation with hardware modules, thus bridging the domains of physics analysis and hardware implementation.

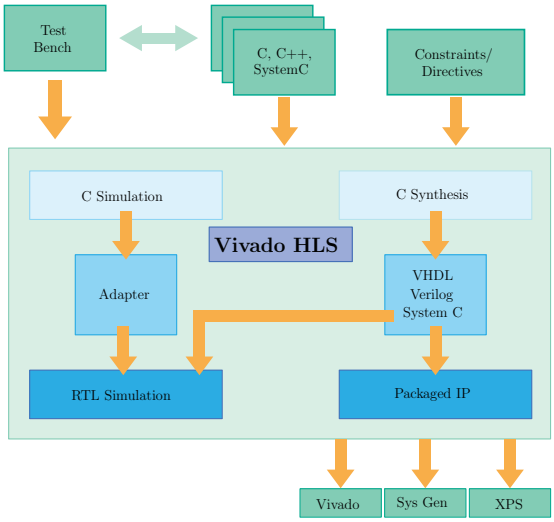


Figure 2.26.: Design flow for FPGA implementation using the Vivado HLS tools [Hoc18].

HLS does not have to be used exclusively for the generation of RTL modules, but can also be used to describe the configuration and parameters of a predefined architecture in the form of high-level description languages. The separate modules of the architecture are meanwhile already provided within a library of parametrizable IP cores and merely configured by the HLS tools. In the following, the design flow of the Vivado HLS is described. For this, the graphical representation of the flow is presented in figure 2.26. The basis of the design flow is formed by the high-level description accompanied by a test

bench for validation and constraints coupled with directives. The later ones are similar to constraints in traditional hardware design, however, they also include more abstract metrics such as a frequency target and implementation strategy to be used. An important strategy is hereby the array partitioning, which directs the tools to implement access to memory structures in parallel instead of sequential access to BRAMs. Using the high-level description coupled with the constraints, C-based synthesis is performed, generating RTL modules in a predefined HDL. These can then be packaged to be used in the overall architecture and used for HW/SW Co-Simulation.

2.5. Machine Learning and Classification

This chapter is introducing the fundamentals of machine learning and classification, which form the functional basis of the algorithms throughout the presented systems. The task of all developed systems is the classification of event data into their relevancy for the experiment. As such, generally used metrics are discussed, which are used later on to evaluate the performance.

2.5.1. Classification

As the methods used within this thesis focus on the classification of data, the basic principles of classification are introduced in this section. The problem of statistical classification can be seen as testing a defined hypothesis. The test is centred around a data set matching one hypothesis against the same data set matching another hypothesis. Subsequently assigning the data set to one of the hypotheses can be graphically interpreted as introducing a separating cut into the data space. Without the precise knowledge of how to assign data to the correct hypothesis, this will always introduce a probability for misclassification. Two terms are then typically associated that are false positive and false negative. False positive represents data that is being assigned to a hypothesis H_0 which in reality, belongs to an alternative hypothesis H_1. On the other hand, false negatives represent data that is in reality matching the hypothesis H_0 but was wrongly classified to be assigned to H_1 due to the dividing cut.

$$Efficiency_A = \frac{Correct\,Classifications_A}{Sample_A} \tag{2.2}$$

$$Purity_A = \frac{Correct\,Classifications_A}{Correct\,Classifications_A + Wrong\,Classifications_A} \tag{2.3}$$

From these considerations, two metrics for measuring performance can be defined. These are efficiency and purity, both can be calculated by using equation 2.2 and 2.3, for a given arbitrary data set A. Efficiency is the ratio of correctly classified data and data belonging to the class for the provided sample, irrespective of the falsely classified data. On the other hand purity is the ratio between the correctly classified data and all data assigned to this class. Both metrics have a relationship to each other that becomes apparent when creating

a purity-efficiency graph. Here an efficiency of 100% and a purity of 100% represent the optimal result for the respective metrics. A typical structure as a result of classification is shown in figure 2.27, in which efficiency is increasing at the cost of purity and vice versa.

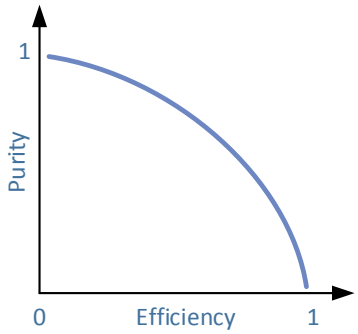


Figure 2.27.: General plot showing the relationship between purity and efficiency.

2.5.2. Approaches based on Supervised Learning

Methods from machine learning can be divided into the two classes of supervised and unsupervised learning. The difference lies within the a priori knowledge to which class a data sample is actually belonging to, which is present in the case for supervised learning. The machine learning algorithm is then learning the classification by determining the errors between the generated and real results by using the provided sample data and adjusting the internal topology in order to minimize this error. In unsupervised learning, the correct assignment is meanwhile not known in advance. The possible classes are then learned by the algorithm itself. The machine learning methods used in this thesis are all based on supervised learning. They are either trained by using simulated physics events and their interaction with the respective particle detector or by using real data recorded from the sensors during operation from either collisions or cosmic rays.

2.5.2.1. Artificial Neural Networks

Artificial neural networks are a processing architecture derived from biology, with the goal to emulate the behaviour of nervous systems. Although the basic building blocks called artificial neurons are in their basic form rather simple in their internal processing structure, they can be used to approximate arbitrary functions when using a suitable topology.

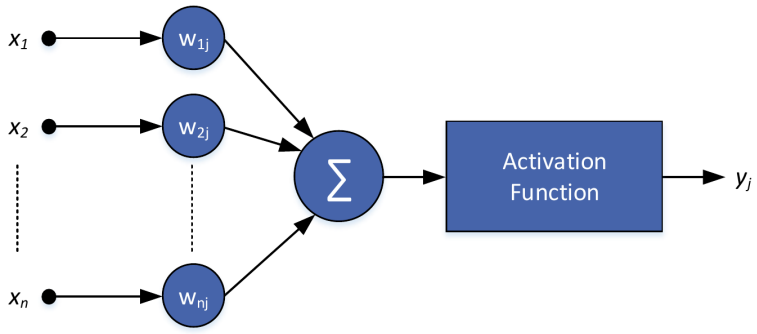


Figure 2.28.: Structure of a single artificial neuron within a MLP.

Of particular interest are artificial neural networks in connection with machine learning. Their internal configuration and thus function is hereby determined by a training process based on provided data samples. This training will adjust internal parameters in such a way that a processing problem can be solved without the knowledge of an exact solution algorithm. The basis for successful training is a sufficient amount of relevant training data.

$$y_j = f(sum_{i=0}^{N} w_{ij} \cdot x_i) \tag{2.4}$$

Artificial neurons can be algorithmically described according to equation 2.4. The output of its function is noted as y_j for a given neuron j. Its weights w_{ij} are determined from the training and represent the importance as well as the impact of either the input data or connected neurons within the network. These weights are multiplied with input values x_i of the neuron, at the end all results from multiplication are summed up. This sum is then fed to the activation function f. Figure 2.28 shows the relationship of equation 2.4 graphically.

Activation Function

Typical neural networks in biology transition between excited and non-excited states. In artificial networks, this is modelled by an activation function. It is typically chosen to be monotonously increasing. The choice of function to be used is a design parameter selected before training of the network, as the training process is highly dependent on this decision. In artificial neural networks, jump functions, piecewise linear or non-linear functions, such as the sigmoid and Hyperbolic Tangent (TANH) functions, are often used as activation functions. The advantage of the sigmoid and TANH functions is their differentiability, which allows training of the nets with backpropagation algorithms.

Backpropagation

One of the most popular approaches to training and thus configuring an artificial neural network are backpropagation algorithms. These are supervised learning methods that are based on the usage of an error function that is predefined, which describes the deviation of a selected neural network's output from the expected desired output. It is then used to determine the influence of the individual weights making use of the differentiability of the activation function. Simple gradient methods such as gradient descent can then minimize the error function. In the context of this thesis, the iRPROP algorithm was used to train the neural network of the NNT that is presented throughout chapter 5 [92]. Compared to conventional backpropagation algorithms, the iRPROP algorithm provides the same cost minima with a shorter runtime.

Multi-Layer Perceptron

The basic form of artificial neural networks is represented by MLPs, which consist of a series of interconnected layers of artificial neurons. A network that does not include any feedback to a previous layer, is then called feed-forward MLP. These have the advantage that both the latency and output of the network are deterministic. It was additionally shown that such a network can be used as a universal approximation for any bounded, continuous function when it is appropriately trained and configured.

$$z_k = f\left(sum_{j=0}^{M} w_{jk} \cdot f(sum_{i=0}^{N} w_{ij} \cdot x_i)\right) \tag{2.5}$$

2.5.2.2. Deep Neural Networks

The previously presented MLP represented the classic variant of a neural network. Today, however, most of modern neural network based algorithms can be assigned to the class of Deep Neural Networks (DNN). These variants are distinguishing themselves by the fact that they often consist of a many successive layers, sometimes dedicated to fulfilling a dedicated task with their parameters determined by a learning method. Two of the most famous members of this class are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [65]. Both are briefly discussed in the following.

Convolutional Neural Networks

CNNs are probably today's most widely used derivative of machine learning-based methods. They are most famous for being used in the recognition of objects within images, which matches the strengths of these networks. They are particularly well suited in the processing of data, which can be represented as multidimensional data arrays and that are in need of finding local patterns. An example case to be classified are images recorded by surveillance cameras with the goal to identify persons. Additionally, this represents one of the use cases in which machine learning is combined with deployment on FPGAs as these systems are often integrated close to a recording camera in order to allow real-time classification.

In contrast to classical approaches, CNNs are specifically designed to have several processing layers. The general architecture is similar to that of MLPs, but often heterogeneous

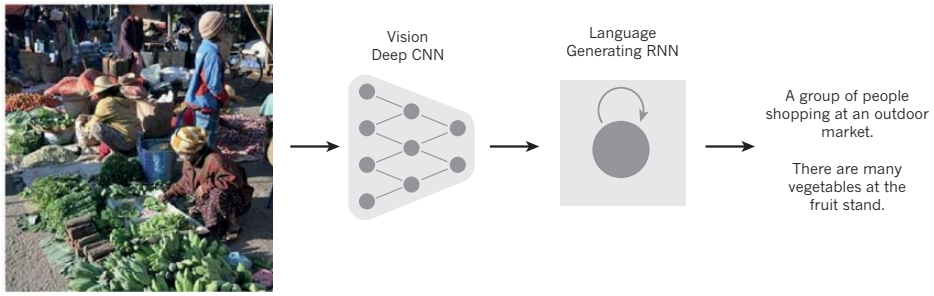


Figure 2.29.: Representation of the data flow in an image recognition use case that is a typical application case for CNNs [65].

processing topologies are used across all of the Hidden Layer (HL). Typically the first layers directly after the input reception are designed to use fewer connections between the neurons, while later layers are moving towards being fully connected. Using layers of fully connected neurons is retaining the original properties of MLPs that are intended to be universal approximators of functions. In addition to the difference in connectivity, different algorithms are often used across successive layers. The core processing techniques of CNN's are hereby convolutional and pooling layers.

The name convolutional is representing the typical arithmetic operations within a layer. A defined part of the original data, for example, a 3x3 pixel-section of an image is processed by using a filter bank. The calculation mathematically corresponds to a convolution, which gave the name. The filters used can differ from layer to layer. Meanwhile, the result of a convolution is passed to a non-linear function, similar to the activation function of an MLP. In deep networks with many layers, the Rectifier Function (ReLU) function is typically used for the activation. The advantage of this function is that it typically allows faster learning and is more efficient to implement. Especially in very deep networks, it is enjoying high popularity with its advantages in avoiding the vanishing gradient problem. On a higher abstraction level, the principal task of a convolutional layer is to detect predefined features or patterns within a subset of the original data. For layers that are deeper within a CNN, the result of the pattern matching is used as the input to the successive layers.

Pooling layers are used to combine features that were found while processing a layer. A typical algorithm used in pooling layers is the determination of the maximum within a list of features found by previous layers. Going back to the application case of image processing, pooling layers with the maximum approach can be used to make the algorithm more robust against different positioning of the desired features within the image.

Deeper layers of a CNN are often switching to another topology. In these, typically more convolutional layers are executed successively without any pooling layers in-between. The connectivity is additionally increased in these sequences across the respective layers. These kinds of topologies led to the tipping point for the success of modern CNNs as they severely increased object recognition efficiency which made them the algorithm of choice

for these types of applications. Topologies following this template assume enormous proportions with over 20 layers and millions of weights. In comparison, the networks used within this thesis have much simpler topologies following the traditional approach to neural networks with a maximum of weights at around 2000 to 4000.

Recurrent Neural Networks

Another modern and popular style of algorithms based on Machine Learning are Recurrent Neural Networks (RNN). While CNNs are especially well suited for processing of image-based data, RNNs are designed to be particularly suitable for sequential input data streams in which successively arriving data is interdependent. The singular special feature of such networks, which makes them very suitable for such applications, is the ability to store an internal state that is reflecting previous processing steps. These states contain not only information derived from the current input, but also the previous ones. Neurons are hereby represented by so-called hidden units together with their current state at a certain point in time. The reason to keep the artificial neuron representation is to use and adapt learning algorithms such as classical backpropagation for these kinds of networks.

The processing principle of such a neuron is shown graphically in figure 2.30. Each neuron contains a triple of weights (U, W, V) that is applied on the input x, the state of the neuron s, and the output v. The input x is hereby a vector over time t. At any point in time, a neuron calculates its output not only depending on the input but also on the state from the last time step t-1.

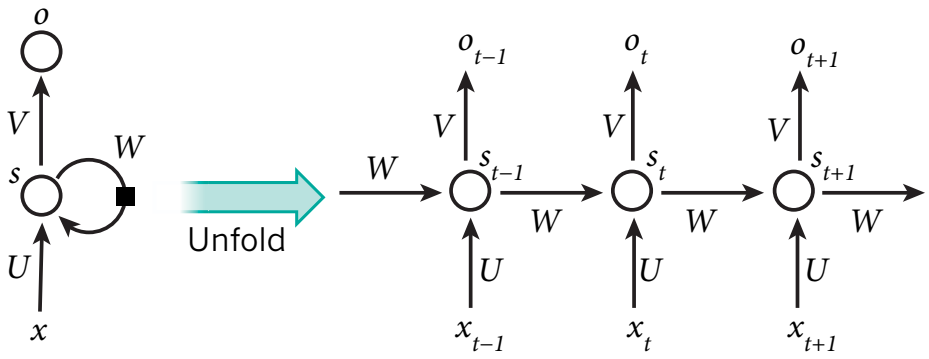


Figure 2.30.: Illustration of the processing principle for neurons within a RNN [65].

In the past, it was rather difficult to train such networks efficiently. As a result, they were not widely used for a long time. Current iterations changed this however and RNNs are representing one of the most popular algorithms to process texts effectively, for example, in machine translation. An interesting approach to DNNs is to use a combination of both CNN and RNN in order to profit from the advantages of both approaches. One such example is the processing of video feeds. Approaches were developed that use CNNs to automatically label single frames within the feed, while a RNN is used to generate the text to be inserted for the labels.

2.5.2.3. NeuroBayes Algorithm

The NeuroBayes algorithm is a modified neural network currently provided and maintained by Blue Yonder and the JDA Software Group [28, 44]. Its original target applications were in the field of high energy physics as it was used to provide an optimized reconstruction of energies for the DELPHI experiment [6]. After its initial success, its employment was extended towards further experiments such as Belle. Nowadays the algorithm is used as a commercial product for data analysis. Usage of the algorithm is provided through the NeuroBayes framework, which maintains several variations of the algorithm, for example, an MLP or the so-called zero-iteration algorithm, which is a simplified implementation of a general neural network. The most important differences to conventional algorithms based on neural networks are the inclusion of a sophisticated preprocessing which allows for high-efficiency operation and large-scale pruning of the network. In addition to this, it is based on the usage of the theorem of Bayes for improving the algorithm's efficiency. In the original design, the theorem was used to exclude behaviour that appeared to not be based on physics due to imprecisions that are present within the detector used for data taking. The zero-iteration algorithm proved to be the most suitable solution for the OCA that was developed as part of this thesis and is presented in chapter 8.

3. State of the Art

This thesis's presentation of the state-of-the-art is primarily focusing on similar trigger approaches used in other modern particle physics experiments. Less focus is put on the investigation of the design of architectures for machine learning approaches, especially neural networks, on FPGAs. Especially here, there is a plethora of solutions available due to the current trends towards using machine learning. Besides the currently available approaches, an outlook into future operation of neural triggers is provided by discussing potential future platforms that might host such approaches.

3.1. Related Trigger Systems based on Track Finding

3.1.1. Approaches based on Neural Networks

The neural L2 Trigger used at the Hera Experiment

The advantages of a track trigger based on neural networks for the particle detector readout garnered much interest during the development of DESY's HERA particle accelerator experiment. It was then designed and constructed to include a neural trigger system [56, 25]. Similar to Belle II, it was expected that full luminosity would result in background formations for which conventional methods would not be able to achieve the required efficiency at track finding. Physics analyses showed that an estimation of the coordinates with the help of neural networks would be able to achieve the required reduction rates. These networks were then implemented on dedicated hardware as part of the L2 trigger system. Similar to the NNT discussed within this thesis, this trigger system had to be placed close to the detector readout. Close location to the sensors was vital for this system to stay within the maximum total latency. The architecture of the used overall trigger system of the experiment is shown in figure 3.1.

The most relevant differences between the system developed back then and the one requested for Belle II are the requirements set by the detector's readout and trigger system. At the HERA experiment, a latency budget of 20 μs was allocated for the neural network trigger system. In comparison, at Belle II CDCTRG only a quarter of that time is available for the entire trigger, not just one sub-system, as it needs to generate trigger signals within 5 μs. The NNT presented here, represents only a part of this system, which has about 300 ns available for its latency budget for data processing. In addition, the L2 trigger was not pipelined, thus was not dead time free, while the NNT must fulfill both.

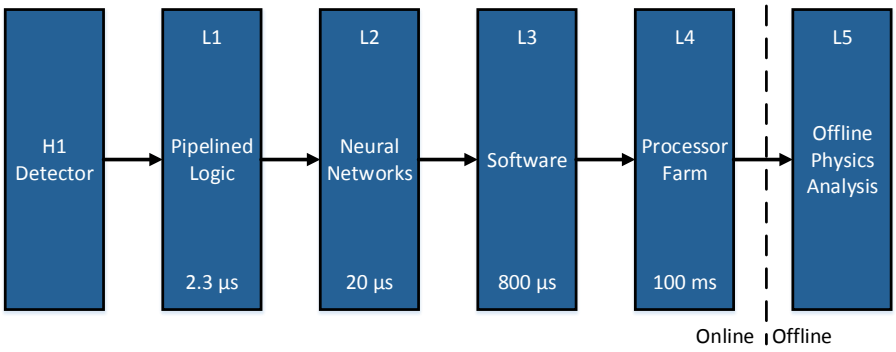


Figure 3.1.: System architecture of the trigger system used at the HERA experiment.

The hardware basis of this neural trigger was a dedicated ASIC called Connected Network of Adapted Processors (CNAPS), which was developed by Adaptive Solution [8][9]. This chip implemented a general feed forward network, which operated internally with fixed-point arithmetic supporting a 8 bit representation for the input and 16 bit for the weights.

Investigation of Neural Trigger Systems on FPGAs

First investigations for the usage of a low-latency neural trigger based on FPGAs were conducted well before Belle II [114]. The main application case discussed within this work is quite similar to the tasks investigated within this thesis as it was primarily developed for particle classification. Subsequently, data sets representing both signal and background events were generated at first, afterwards they were used to train and determine the topology of an MLP. This topology was then implemented on an FPGA. The individual operations were entirely implemented in the Slices of the FPGA, while the activation function was implemented in BRAM. In order to perform the operations efficiently on the FPGA, both inputs and weights were converted into integers with restricted bit widths. Finally, the resulting resolution was investigated, whereby a good separation of signal and background was achieved. This work already showed the potential of MLPs to be used as the algorithmic foundation for the usage in trigger systems as it efficiently separated classes of particles and could be implemented on an FPGA. The latency achieved in this work was at 200 ns, which was assumed to be a reasonable maximum latency to be feasible for experimental operation. This can be confirmed looking ahead to the NNT as it has a maximum latency budget in the range of 300 to 400 ns for data processing.

However, this work rather represents a principal investigation of the usability of neural networks in the scope of these types of applications. It is not based on any data sets or use cases motivated by physics. As such, it does not consider any detector specific components such as detector specific processing and integration into the readout chain. At the same time, the verification was limited to simulated data and simulation of the FPGA architecture without consideration of any real data. These are the key points in which

that work is different from this work, as it represents a real system that is already at this moment operational within the experiment.

High-Level Synthesis for Trigger Systems based on Machine Learning Methods

The most interesting work within the state-of-the-art so far is the framework High-Level Synthesis for Machine Learning (HLS4ML) [26]. This framework represents the work conducted by multiple research groups together with CERN towards using machine learning for trigger applications as the framework aims to instantiate neural network architectures for FPGAs. The core applications that are considered for evaluation are hereby trigger systems on the basis of jets for usage at the Compact Muon Solenoid (CMS) experiment. This work is, therefore, the most comparable to the one presented here.

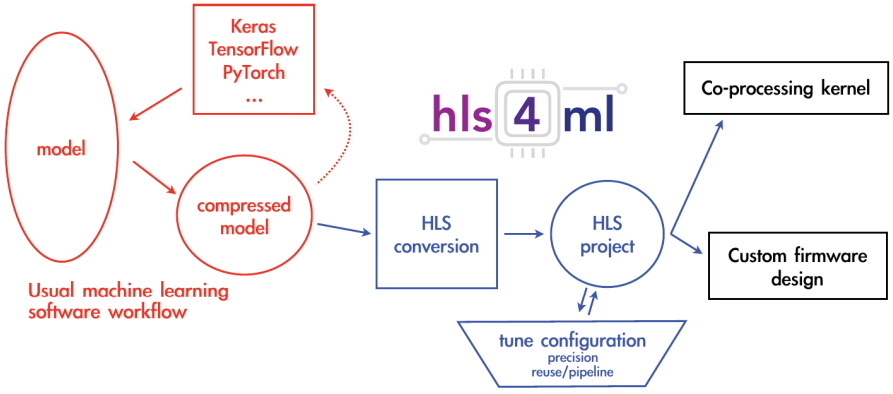


Figure 3.2.: Design flow of the HLS4ML Framework for inference of neural networks on FPGAs [26].

The core idea of this framework can best be described by examining its design flow, as shown in figure 3.2. The starting point is the creation of a neural network by using state-of-the-art tools available for machine learning methods such as Tensorflow, PyTorch and Keras [2]. After creating a suitable model, it is compressed in order to reduce the number of operations as much as possible, which will result in resource-efficient and fast implementation for FPGAs. The compressed model is then converted for instantiation supported by the HLS tools of Xilinx. In addition, implementation targets are defined in terms of the throughput and latency to be achieved. The desired topology is then implemented with the help of the HLS tools, taking these key figures into account.

The design flow is demonstrated and evaluated by using physics use cases derived from data from the CMS experiment. Different pipelining possibilities and bit widths are evaluated within this use case. Meanwhile, the implementation of neurons is based on the usage of DSPs in order to achieve high operating frequencies. The evaluation concludes with the definition of a suitable topology and architecture configuration that is capable of

achieving the defined goals. Especially of interest is the compression of the model as it achieves a significant optimization of the architecture.

The framework developed within HLS4ML offers a mature software infrastructure for inferring machine learning models as it is supporting most of the state-of-the-art tools from this area. This advantage is coupled with highly effective compression methods for the generated models. With these two key features, it sets itself apart from the systems developed and presented within this thesis. Rather than developing a general software framework based on the state-of-the-art, the solutions presented within this thesis are tailored for the use cases of Belle II. As such, it is built around supporting the tools that were used for inferring the neural networks during early and late studies of Belle II. However, especially the compression of HLS4ML is interesting for the use at the NNT. An analysis of compression with alternative methods was carried out for the NNT [Reu18] but did not provide a sufficiently significant improvement in order to be used. The reason for this is the targeted application case. Since several networks are implemented and operated in parallel on the FPGA, compression has to be performed across all networks in order to have significant optimization of the FPGA architecture. Another distinguishing feature of this work is that it represents a rather theoretical study of the presented concepts. The FPGA implementation examined here is not in use, nor has it been integrated. Furthermore, it is limited to the inference of a neural network without any consideration preprocessing.

3.2. Alternative Track Trigger Approaches

Conventional 3D Tracker for the CDC

Parallel to the NNT for the CDC, a conventional 3D estimation [115] is under development. It will be operated in parallel to the NNT, which is showing the importance of an operational estimation of the z-Vertex as two systems are tasked with this responsibility. The idea here is two have two systems having complementary strenghts for different classes of tracks. This approach is mainly developed by researchers from the Korea University. Algorithmically it is based on geometric transformations and uses linear regression to determine the 3D track parameters. The name conventional stems from the method being used to estimate track parameters in previous experiments. Just like the NNT, its main task is to send an estimation of track parameters to the GDL. The goal is meanwhile the same however it is projected to be more effective for certain types of tracks. Preliminary simulations of this trigger showed that reasonable estimations of the z-Vertex could be achieved with a fitted sigma of 1.35 cm for the resolution of the estimated z-Vertex, currently the resolution is worse than that in operation but progress is being made steadily.

The algorithm was implemented on the basis of the UT3. The first implementation was achieving an estimation of the track parameters within 264 ns using this platform. Thus, the implementation of the 3DS will generate its trigger signals within the latency budget of the CDCTRG. With regard to the throughput, it is capable of processing up to 4 tracks in parallel and has no limitation in the amount of supported stereo TSs. In contrast, the NNT developed in this thesis can only support a limited number of stereo TSs at any

time. Additionally, it can process a lower number of tracks in parallel, both did not affect its performance thus far, however.

At this point, it might seem that the NNT has only disadvantages compared to the 3DS, mostly due to its lower throughput. The advantages, however, are present in the quality of its estimations of the z-Vertex, together with its current readiness for being used during operation with collisions. As the NNT is built around using ML methods, it can be adjusted to the situation currently present at the experiment. In contrast, the 3DS is based on less flexible algorithms, which might impact estimations, possibly deteriorating the precision, when the experiment's behaviour is changing. At this stage of development, it is the NNT presented within this work that achieves the best results, with the results being presented and discussed in section 5.4.2.2.

Hough-based Track Trigger for the CMS Upgrade

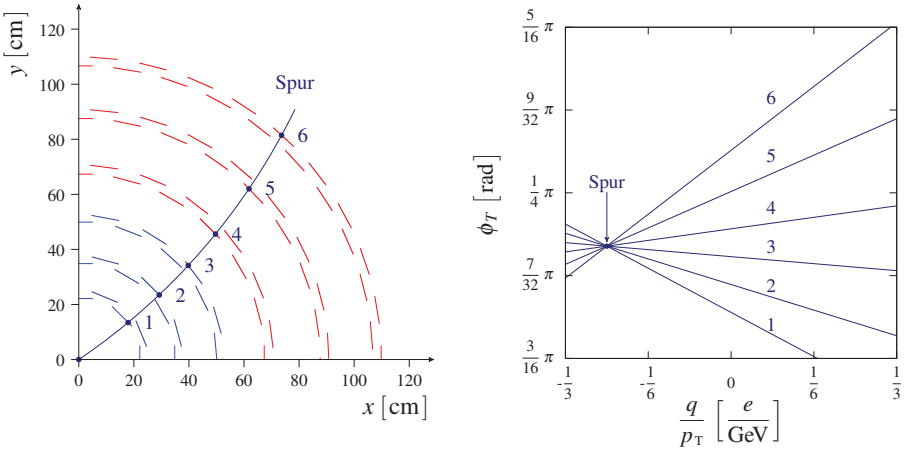


Figure 3.3.: Illustration of the transformation of particle tracks into a related representation in the r-phi plane [97].

One of the main detectors at the LHC of CERN is the CMS. This detector is used to investigate a wide field of physics, for example, Higgs, super-symmetry, and dark matter. During operation, between 20 to 40 collisions are occurring simultaneously at a frequency of 40 MHz. The L1 trigger reduces the resulting data rate within a latency budget of 3-4 μs, while data is reduced with a factor of 400. Currently, an upgrade of the detector is under development in order to be ready for the upgrade of its accelerator the High Luminosity LHC, which is planned to begin operation in the year 2026. The trigger system will be revised to handle the increased luminosity and data rates. It will centre around a novel tracking detector, that is based on silicon strip sensors, similar to the SVD of Belle II but with much larger dimensions. This detector currently represents the largest detector of its kind. Of course these developments already led to revised algorithmic approaches

to track triggering. One particularly interesting solution that was proposed, is the use of Hough-based track reconstruction. For this, FPGAs were selected as the target technology in order to meet the requirements for data transmission and latency.

In the proposed procedure, the detector's data is transformed into a r-phi plane representation according to the detector's geometry [97]. Possible track candidates are represented as straight lines in this plane. A track is then regarded as a possible candidate in case a sufficient number of straight lines are intersecting. This procedure is shown in figure 3.3. The track parameters found this way are then optimized again by using a Kalman filter.

The proposed solution was prototypically implemented on MP7 boards. These are custom processing systems based on FPGAs of the Virtex-7 series. The suitability of the prototype was then shown within a demonstrator setup. Overall, a high efficiency of 94.4% is achieved for track finding. The measured latencies of the demonstrator contained all key parameters, including the transmission latencies. The setup is then able to generate an estimation within 3.7 μs, which is around the targeted maximum latency [43].

Especially the implementation of the Hough transform for FPGAs is of interest for this thesis, as an FPGA-based estimation of three-dimensional track parameters based on a similar approach is the basis for the system presented in chapter 6. Even though they are both based on the Hough transform, both systems are quite different as the variant presented in this thesis is optimized for the particular geometry of the CDC. Algorithmically the approach used in this thesis is based around a weighting scheme for track parameters that is using the theorem of Bayes. In addition, the finding of a suitable track candidate is performed in a three-dimensional Hough space.

3.3. Hardware Acceleration for Machine Learning Algorithms

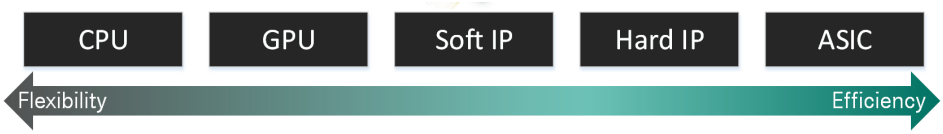


Figure 3.4.: Portfolio of possible technologies for the realization of algorithms based on machine learning. They differ in their trade-off between flexibility and efficiency.

The rediscovery of neural networks in industry and science has opened up many new areas of application. One of the remaining problems for such approaches is the discrepancy between the required and available computing power. Even today, the demand for computing power is still much higher than its availability. In the past, the performance was mostly optimized towards high achievable throughput and accuracy, but with application cases arising from the embedded domain, also energy, space, and latency are in demand. While the throughput was mostly addressed by using Graphics Processing Unit (GPU)s, they are not designed to address the requirements inherent to embedded systems. In order to close the gap, research on the implementation of efficient and dedicated neural

hardware accelerators was intensified. An overview of the classification of neural computing platforms is provided in figure 3.4. This overview is showing multiple possible computing platforms that are categorized in terms of the two dimensions flexibility and efficiency. These dimensions are complementary to each other as a flexible solutions that are supporting multiple applications are not optimized to achieve the highest possible efficiency for one single application.

The highest flexibility is achieved by classical CPUs. Due to their programmability, algorithms can be quickly implemented. As they are built to provide reasonable performance across a wide range of applications, especially control flow heavy applications, they are equipped with several mechanisms optimizing such tasks. This makes them a less efficient solution for data flow dominant applications such as neural networks. Improvement can be achieved by using GPUs. These were initially specialized in performing graphics applications. Their architecture is optimized for highly data flow-oriented processing, as a result GPUs are built to provide a high degree of parallel processing. This, in turn, leads to improvements in the performance for other application domains, as they are particularly popular for the calculation of highly parallel applications, for example, in scientific computing, but also in the domain of machine learning. Today GPUs are the dominant computing platform for fast training of neural networks, but also for inference in data centres. While they are capable of achieving high throughput, they are not particularly well optimized for power, low-latency, and deterministic processing. Considering the highest-grade classes of GPUs in terms of achievable performance, a fixed deterministic latency typically cannot be guaranteed. Not only are they based on communication infrastructures that are not deterministic in terms of transmission times, but even the calculations themselves are not deterministic on GPUs. In these, operations are controlled by hardware-based schedulers that have the goal of maximizing parallelization. It has been shown that determinism can be forced upon these schedulers by using synchronization features like semaphores, however, this led to much worse performance. Besides using GPUs, high efficiency in terms of throughput and energy can be achieved with the usage of dedicated processing units. A distinction can be made here between three categories. A neural accelerator can, for example, be implemented as Soft-IP. Here, the hardware accelerator is provided as a netlist that can be implemented for example on an FPGA. This module is often somewhat flexible as its mapping to the target platform can be influenced. However, it typically does not achieve the highest efficiency due to the flexibility of the selected target. An accelerator implemented as Hard-IP fares better here. In this case, it is implemented on the chip during production as a Co-Processor and cannot be changed afterwards. It is capable of achieving the highest possible efficiency, as it is basically an ASIC. However, it offers the lowest flexibility as it cannot be modified in any way after production.

3.3.1. Realization of Neural Networks on FPGAs as Soft-IP

This section presents a selection of Soft-IP based implementations of neural networks on FPGAs. They are divided into approaches from academia and industry. Both are interesting because implementations from industry are usually representing more mature integration solutions into the FPGA toolflows, on the other hand, the approaches from

research are meanwhile mostly available as open-source and can, therefore, be adapted to custom needs. The presented implementations include different algorithmic implementations, but they are all part of the neural networks.

3.3.1.1. Academic Approaches

Frameworks for Inference of Neural Networks on FPGAs

Frameworks to assist in the implementation of neural networks on FPGAs are one of the most popular topics in today's research. Most of them focus on providing a flexible interface to support a broad range of applications. As most of the commercial applications are centred around DNNs, much effort is put into facilitating their acceleration. Most of the frameworks follow the same basic principles, such as providing a library of optimized Soft-IPs integrated into the overall tool environment. These are then often used by an architectural description at a high abstraction level. These frameworks are then focussing on the optimization of throughput and latency. In Ref. [137], a throughput-focused framework is presented that can be used for several different algorithms from the class of CNNs. As it is driven by achieving high throughput, it is particularly interesting in the context of future applications for data reduction for a detector's readout, as is investigated in chapter 8. The solution used within this thesis is based on a proprietary algorithm. While it is achieving sufficiently high performance and fulfils the requirements, an open-source solution is going to be required for long-term operation.

Mapping of Neurons onto DSP-based Macro-Cells

One of the core ideas that is discussed in section 4.4 is the usage of DSP blocks that are available on an FPGA to implement the MAC operations of a neuron. In order to be as efficient as possible, the mapping of these operations must incorporate the detailed architectural properties of a DSP. Such an approach is the main focus of Ref. [42], in which the distinguishing feature is that it goes beyond the standard approaches of RTL design by investigating an optimized mapping on the physical level of abstraction. The solution found in this work is optimized for the structures on the level of single primitives available on an FPGA. This is also the distinguishing aspect compared to this thesis, in which focus is put on RTL design. Placement and routing are accordingly not performed by the vendor's tools, they were rather done with the help of the open-source place and route tools VPR [17]. With this approach, it was possible to create FPGA designs that can be operated with very high clock frequencies of up to 300 MHz. However, as they are designed at the physical level, the developed structures are dedicated to the used FPGA architecture.

The interesting part here is the optimization on the physical level, which can also be carried out for the systems developed in the scope of this thesis. All final setups developed in this thesis are able to achieve the minimum set requirements as shown in sections 5.4.2.2, while using a combination of tools from Xilinx and Synopsys. However, especially regarding the throughput, only the minimum is achieved thus far, and further optimizations using physical design could lead to a boost in performance. These optimizations are strongly constrained to the target architecture of the FPGA. This makes it less flex-

ible, however in the trigger context, the choice of platform is already very limited. In addition, a platform must be defined early in the development process. This requirement means that the trigger system will be committed to one platform, and these kinds of optimizations are an option worth consideration. The systems discussed within this thesis were primarily developed with the goal to be operational at the early stages of the experiment. Thus, the focus was put on the fulfilment of the minimum requirements. During the course of the experiment, however, maintenance phases will be performed that can be used for the loading of optimized firmware. Here the concepts presented in Ref. [42] could be adapted.

Binary Neural Networks for FPGAs

One of the key components in the implementation of neural networks is the selected binary representation for the inputs, weights, and activation function. In classical software-based approaches, floating-point representation with high precision, was typically chosen in order to achieve the best possible classification efficiencies. However, these representations are more complex to process on FPGAs and require additional memory resources. To avoid this situation, data types can be restricted to computationally less intensive representations such as fixed-point. This is much more efficient in terms of resources and latency, especially with regard to hardware implementations. One extreme of this approach is the usage of Binary Neural Networks (BNN). In these only binary weights, activation functions and inputs are allowed. With this representation, the memory footprint, the number of accesses, and arithmetic operations can be both reduced and simplified. One of the basic principles within these networks is to reduce the complexity of neurons by avoiding any multiplications. Instead, only logical operators such as XNOR and bit shifts are used. While this is reducing the computational complexity, it typically comes with the cost of reduced accuracy of the algorithm. However, the advantages are significant, and for most applications, reasonable accuracy can still be achieved, thus justifying the usage of such networks [22]. The implications of a hardware-based implementation on FPGAs, CPUs, GPUs, and ASICs have been studied in many papers such as Ref. [88]. On FPGAs, especially, the power demand can be reduced, which is vital for embedded applications. This is mainly due to the reduction of memory usage as accesses to reload weights are significantly reduced due to the smaller footprint. Such networks were investigated for usage at the NNT as part of the master thesis Ref. [Zha18]. However they were achieving low precision in the initial tests with a reduced resolution of about 10 cm compared to the later achieved 5-4 cm. Due to this, the approach was not further intvestigated.

Framework for Fast, Scalable Binarized Neural Network Inference

The Framework for Fast Scalable Binarized Neural Network Inference (FINN) is a mixture of academic and industrial research. Its goal is to provide a framework for FPGA users that combines state-of-the-art ML tools with the creation of an efficient architecture. The original variant is hereby designed to use an optimized library of IP cores for BNN and was investigated for adoption on a Zynq FPGA. In the proposed design flow, a defined topology with its parameters and a throughput target are transformed by the FINN synthesizer into C++ code, which can be interpreted and synthesized by the Vivado HLS tools. This code mainly describes the architecture parameters to be used for processing module that are provided in a library and parameterized accordingly. The overall devel-

oped design flow is shown in figure 3.5. This framework is nowadays often used as a template for design flows that generate neural network architectures for FPGAs. While it is not considered as the framework to be used for the NNT and the OCA, as they are pre-dating FINN, the ideas are incorporated in the design flow for the S3D, which uses HLS tools.

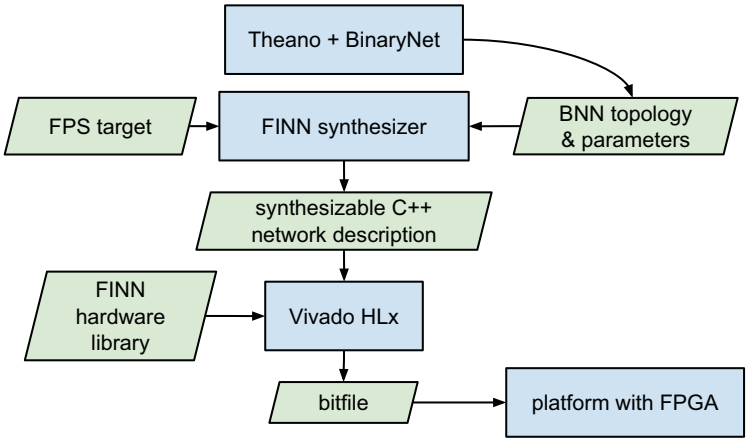


Figure 3.5.: Design flow generating a neural network hardware accelerator based on FINN [111].

Residual Binarized Neural Network

An extension of the idea of a BNN are residual BNNs. The fundamental difference to the original form is that values within the network are no longer just binary. They are instead partially discretized into predefined stages. The idea behind this is to achieve better algorithmic precision while maintaining the basic principles of simplified processing based on pure logic operations in favour of arithmetic operations on the FPGA. As with most approaches, semi-automatic design flows were investigated and available for creating an architecture using this principle. An example of this is Residual Binarized Neural Network (ReBNet) [32], which, similar to FINN, is designed to implement a defined BNN topology based on a library of optimized parametrizable IP cores. These types of networks were investigated for the NNT, however they did not satisfy the accuracy goal as reported in Ref. [Zha18].

3.3.1.2. Approaches from Industry and FPGA-Vendors

The DeePhi Platform

The solutions developed by the company DeePhi are especially interesting since they are already successfully in use in the field of object and person recognition [113]. Their solutions are based on a framework that supports the implementation of a selected topology

on an FPGA [35]. For this, a Soft-IP architecture is provided, which is then optimized either for CNN or RNN [36] processing. Different topologies are mapped onto the provided architectures by using an in-house compiler. This compiler converts a high-level description of the algorithm into the corresponding instructions. The implementations are generally among the most efficient on the market of FPGA-based acceleration. Depending on the selected network and application, they are capable of achieving similar performance as achieved by modern GPUs. They can also be used on smaller ZYNQ platforms, which is especially important for embedded applications or future physics experiments of smaller scale. In the case of object recognition, the developed solution was integrated into camera systems to classify the traffic observed at crossroads.

One of the key technologies is the efficient compression of ML models [37]. It significantly reduces the required memory size and topology, which leads to reduced processing demand. A provided ML model is compressed across three successive steps. First, connections within the network are reduced by removing less significant ones from the network. Weights are then optimized by reducing the bit width and reusing weights across neurons as much as possible. Finally, the weights are Huffman-encoded to achieve further compression. For popular ML applications such as VGG-16 [103] and AlexNet [60], both from the field of image processing, the size of the model was reduced by around 98%. This compression is particularly important because the architecture of the implementation is designed statically, and less FPGA-specific optimization of the weights is performed. This later optimization strategy is the primary technique used for the NNT, but could be improved using such compression techniques.

Xilinx IP Cores

The market-leading vendor for FPGAs Xilinx is pushing forward the development of neural accelerators. One example is the Xilinx Deep Neural Network Processor (xDNN) which is specialized in optimizing DNNs [131]. The architecture is designed to be highly adaptable for the entire range of machine learning applications. For this purpose, all of the typically required resources are provided and programmable. This library will be interesting for future investigations DNNs at trigger systems, as the current solution is using traditional MLPs. As these implementations are provided by Xilinx, they can be expected to be highly efficient.

Machine Learning IP Cores for Intel FPGAs

Being one of the two major manufacturers of FPGAs, Intel has also turned its attention to the application field of machine learning. Targeting the most popular algorithms from CNN and RNN, highly optimized IP core libraries for hardware acceleration, have been developed [14] for their high-end FPGAs. To enable its usage, a broad ecosystem called Deep Learning Accelerator (DLA) [3] is provided. It abstracts from the typical hardware languages and enables the support of domain-specific approaches such as Caffe [46] or Tensorflow [2]. This software framework generates an appropriate scheduling for the used architecture and specific application. The processing within the architecture is then controlled by using a Very Long Instruction Word (VLIW) operation.

The core of this approach is an architecture for neural network acceleration on the FPGA. It consists of a network of systolic computing units, which perform the typical operations in such networks. These units are flexibly designed to support the processing with different bit widths and can thus be configured to the needs of the application. To support different post-processing functions, dedicated functional blocks are placed right after the systolic array. The viability of the developed architecture and flexibility of the software infrastructure were evaluated with the popular use cases for CNN and RNN, for which promising results were achieved.

DLA is a promising solution for easy adoption of machine learning approaches when using Intel FPGAs. One of the possible future platforms for the NNT is based on such FPGAs. Thus it is a solution considered for future application. Usage of this platform was investigated within the scope of this thesis, however already foretelling its conclusion, Intel FPGAs were not chosen for hosting upgraded trigger logic. This, however, might change in future redesigns of the trigger systems, which is only a matter of time.

3.3.2. Realization of Machine Learning Accelerators as Hard-IP or ASIC

In addition to the implementation of dedicated accelerators for inference in ML applications as Soft-IP, the industry is increasingly moving towards the direction of Hard-IP on a SoC for these tasks.

Adaptive Compute Acceleration Platform

The Adaptive Compute Acceleration Platform (ACAP) developed by Xilinx [135] is a very interesting development within the industry, especially with regard to future applications of neural networks in particle accelerator experiments. This platform consists of a heterogeneous architecture that aims at combining all of the separate advantages provided by CPUs, FPGAs, and dedicated neural network accelerators as ASICs. Three basic elements of data processing are hereby integrated on one platform. These are scalar units, vector units, and programmable logic. Here, scalar computing units are designed for use in applications with control flow and software-level programmability, while the vector units are designed for strongly data flow-dominated applications such as found in machine learning.

The first implementation of this concept can be found in the Versal platform. Here, the scalar units are implemented by several ARM Cortex processors. In contrast to conventional FPGAs, the programmable logic is mainly designed to serve as glue logic between the scalar and vector units. A supporting software framework facilitates the usability of this platform. This framework is generating the programming representation for the respective resources. The presence of both programmable logic and highly optimized vector units makes the platform quite suitable for hosting trigger algorithms based on neural networks. The vector units can be used for the neural network, most likely achieving much higher performance than the currently used DSPs coupled with Slices. Meanwhile, the programmable logic can be used to handle the communication tasks of components and realization of the preprocessing algorithms that are traditionally specific to the geometry of the particle detector. In addition, the available processors can be used to handle service

tasks such as configuration management, SC, and DQM. These are typically control flow heavy and only inefficiently implemented in programmable logic. This is supported by the current developments at the KEK. These efforts are focused on providing implementations of the current service portfolio such as B2L for ZYNQ platforms. These tasks are hereby hosted on the ARM processors that are integrated on the ZYNQ. It is quite probable that these implementations can be ported onto the ACAP without too much effort.

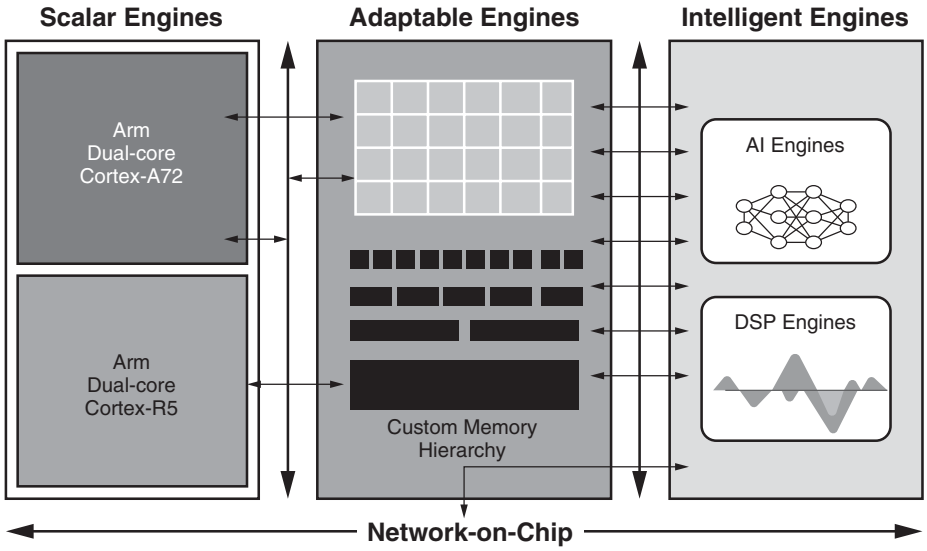


Figure 3.6.: System architecture of the ACAP. Programmability, flexible custom processing and efficient support of ML applications are combined by using heterogeneous resources [135].

DWAVE Computing Platform

While the ACAP is targeting to cover a wide range of applications, other hardware manufacturers are focusing on directly accelerating data flow-oriented applications, as is the case for neural networks without any additional processing. An interesting platform hereby is the DWAVE Computing Platform [84]. It is aiming at accelerating both the inference and training of neural networks as efficiently as possible. Its key selling point is at providing performance on the level of computing centres without their space occupation and power consumption. The core idea of the platform is to move away from traditional computing at the data centre towards a solution that can be placed locally at the workplace. This system is providing low power consumption and is thus without the need for high-end cooling. The most interesting part of these developments is represented by their future edge computing portfolio that aims at the embedded domain. Depending on the available communication resources and carrier platforms, these might be interesting dedicated platforms for machine learning-based trigger systems.

3.4. Summary

Neural networks for a trigger system in particle accelerator experiments have already been used or investigated in previous experiments. Thus far, they were not used at L1, as they were instead deployed at later stages that have larger latency budgets and are typically not integrated into a system-wide pipelined processing architecture. Investigations on their suitability for L1 trigger operation have been carried out to prove effectiveness with suitable frameworks that try to close the gap between algorithm and FPGAs. However, these frameworks are not in use and typically have simplified assumptions such as not considering the integration aspect, and thus, the concrete data flow. The systems presented within this thesis are, on the contrary fully integrated and thus the first operational systems that truly show the advantage of employing such algorithms.

Neural networks are not the only option for track triggers and online data reduction. Already within Belle II they are supplemented by alternative approaches with the same task, that is the 3D track finding and RoI based data reduction. These proved to either achieve worse prediction results when facing background events that were not anticipated or are less capable when tasked with taking care of special cases such as identification of specific particles with low momentum. As there are many approaches, the investigation focused on solutions with a similar experimental background, which is represented by HL-CMS. Here a specialized variant of the Hough transform coupled with a Kalman-Filter has shown to achieve results that are satisfying the strict requirements present for such systems. It shows that significant improvement can be achieved by coupling dedicated traditional algorithms with novel approaches. However, the shown approach is custom-tailored to this specific experiment, which also has a much longer time schedule until the experimental operation is expected to commence compared to Belle II. Within this thesis, the realization of a derivation of the Hough transform coupled with the theorem of Bayes, is presented. The focus is put on the hardware-based design and realization. Due to the present time schedules, the decision was made to use and refine the already present and implemented methods, for example the 2D track finder, instead of adapting effective approaches from similar experiment.

The usage of machine learning at the trigger system is rather novel in itself, as there was no operational system prior to the one presented in this thesis. These methods are currently a focal point of both industry and research. As such, the volume of the state-of-the-art is immense outside of particle physics. The dissemination of this can be partitioned into frameworks for architecture generation, hardware-level optimization approaches, and an investigation of available complete ecosystems of accelerators and frameworks. Most of these are focused on the usage of modern ML approaches based on DNN as well as modern high-end computing platforms. The systems to be developed here have to be designed for a time period of over ten years, in which they have to be compatible with the tools and hardware used at the early stages of the experiment. The best solution was to use more simplified MLPs not relying on high-performance FPGAs, to be operational at an early stage. An intriguing future idea is to investigate the suitability of hardware platforms aiming at providing dedicated acceleration of ML algorithms while inheriting the flexibility of FPGAs, as is the case with the ACAP.

4. General Requirements and Fundamental Design Templates

The aim of this thesis is to develop FPGA-based processing systems for use cases from particle accelerator experiments, which require real-time processing of algorithms from the field of machine learning. Accelerator physics hereby provides special applications with strict requirements. This chapter is at first discussing the requirements present at such systems. Both design and implementation are strongly influenced by these requirements. By investigating the requirements, the systems discussed here are additionally distinguished from other applications in the realm of hardware acceleration for machine learning. Subsequently, a general architecture and implementation strategies for neural networks-focused designs that are used throughout this thesis are introduced and discussed.

4.1. Requirements and Constraints

Pure offline analysis of particle decays based on machine learning methods are focusing on achieving the best possible results for mostly two metrics. The aim is to achieve the highest possible efficiency and suppression. For a real-time implementation of such methods on FPGAs, however, further constraints have to be considered. These must also be considered early while establishing a neural network's topology to ensure that the obtained results are representative. In this section, a selection of metrics is presented to evaluate the performance of a machine learning method implemented on an FPGA. The discussion of these metrics is focused on the primary application cases of data reduction and trigger systems.

4.1.1. Connectivity

The data flow of modern trigger systems consists of deep processing pipelines across multiple stages of readout electronics. In these pipelines, the data from the detectors is read out by a distributed system. Afterwards, they are at first combined into a unified data stream across the boundaries of the distributed electronics and converted into a representation that can be used by subsequent processing stages, for example, tracking.

The FEE of modern detectors has numerous parallel channels through which data is transported. The number of channels is hereby proportional to the required bandwidth. For FPGAs in this environment, this is also proportional to the number of required IO ports and determines the minimum speed grade to be used. Usually, the number of ports re-

quired to establish communication is too large for one single hardware platform to receive all the data in its entirety. Preprocessing stages are typically used in trigger systems for combining data and applying coarse data reduction, such as the TSF, for example, which groups wires and limits the accepted angles of tracks passing through the CDC. In addition to the basic data transfer, trigger components typically have to support service interfaces that transmit status information, receive a common synchronous clock and more.

Three major tasks have to be solved for an FPGA-based realization. Firstly, a platform must be found that provides a sufficient number of IOs to receive and send the necessary data. At KEK, for example, custom-made FPGA boards are usually developed for this purpose, as most of the commercially available platforms are not equipped with the required set of interfaces.

The next task to be solved is that data is sent through multiple technologically heterogeneous channels. These are typically generating their data streams in parallel to each other and, as a result, are having different latencies. As a result, a trigger component must be able to compensate for the difference in latencies to synchronize the internal data processing. This synchronization requires additional data buffering concepts, that can be specific to the behaviour of the respective detector, for example, the CDC is based on drift times, which are not transparent to the receiving system.

The last task to be addressed relates to the geometric view of designing for FPGAs. The received data must be merged together for common processing since it the channels are usually connected to different ports of the FPGA. These ports are geometrically located at different positions on the FPGA itself. As a result, care must be taken to ensure that the signals of the individual data streams can be routed together for processing as efficiently as possible in order to avoid low operating clock frequencies due to high signal propagation times.

4.1.2. Monitoring

Both trigger and data reduction components must be checked for correctness during operation. If a component does not function correctly, either unnecessary data is continuously written out, which worsens the experiment by reaching the bandwidth limit, or even worse wrong estimations are produced, which will result in the loss of valuable physics data. Both situations cannot be tolerated during operation. A control mechanism must be set up that can check whether the trigger decision was correct as soon as possible after it was generated. If the generated estimations are deviating too far from the expected values, the system should be switched off for the current run to avoid problems and the run coordinator should be notified to be aware of possibly problematic data.

As a result, an interface and concept for online monitoring of the system must be established. This concept must explore the possibilities of monitoring internal processing for in-detail investigation of the correctness within a defined latency, typically in the range of seconds. For this purpose, an additional communication interface must be established, which forwards the required data. Here the DAQ of the experiment can be used. Since it is used by several components simultaneously, most of its bandwidth is already used up

so that the maximum data rate for online monitoring is typically very limited. This must be taken into account in the design when selecting which data is to be sent. Additionally, a concept for extraction and validation of the data outside of the trigger system has to be established.

4.1.3. Accuracy

One of the main application domains for machine learning methods is the classification of data and thus the extraction of meaningful conclusions. The ability of a method to perform these tasks correctly is measured in either the efficiency or suppression that is achieved. In an offline analysis, data types for floating-point arithmetic will usually use unrestricted bit widths or be specified for the used computer architecture. Since these analyses are often performed on computing clusters or server machines, they are typically too impractical to be used at embedded devices. For example, the bit widths required for floating-point operation can traditionally only be realized with considerable overhead on an FPGA-based system. Thus a high bit width used for the signals usually leads to an increased demand for hardware resources together with an increase for the latency of the calculation.

When a found network topology is to be mapped onto FPGAs, these aspects have to be considered, and parameters like weights of the neurons or the activation function have to be mapped to a suitable representation. Usually, this mapping leads to deviations of the results generated on the FPGA compared to an ideal realization on an offline system. This deviation can be measured in terms of the accuracy of the prediction as the difference of the classification decision for both variants. Even more important is the influence on the efficiency and suppression rate, which has to be determined together with a physics analysis.

4.1.4. Throughput and Latency

In the application cases considered here for online data reduction or trigger systems, the throughput of the classifications performed per second, and the latency are of great importance. If both characteristics are tightly constrained for the selected application, careful balancing is required. In most cases, latency and throughput are on opposite sides of optimization goals targeted by implementation methods. High throughput is achieved in hardware by applying a high degree of pipelining. Adding additional registers to the data processing meanwhile increases the latency of the entire processing due to the added setup and hold times as well as the clock skew. In the realizations, a compromise must be found here for both goals. The throughput in such applications is typically determined by the detector readout and its data rates. At Belle II all online processing modules have to fulfil hard constraints. The same holds true for the latency, however trigger applications typically have harsher latency requirements due to their task of deciding on data before back-pressure is becoming an issue.

4.1.5. Memory Demand

In a prediction model based on neural networks, storage elements for all of the used weights are necessary. For example, for a neuron within the hidden layer, a separate weight must be stored for each input value together with an optional bias weight. Assuming a 16 Bit fixed-point representation for each weight and 27 input values, this will amount to 27*16+16 Bits = 448 Bits that have to be stored for one single neuron. In addition to weight sets for every single neural network, multiple networks could be used in alternation, which will increase the total demand for storage on the FPGA. The total amount of weights to be stored can easily exceed the limits of the targeted platform. If one considers the relatively small amount of on-chip memory available on FPGAs, the limit is easily reached. FPGA platforms typically provide external memory for large scale storage. However, these typically inhibit high access latencies due to the increased transmission distance. This is particularly problematic in the applications at particle detector experiment, as they have to fulfil tight low-latency requirements.

To use neural networks on FPGAs, a sufficient number of memory resources have to be present, preferably on-Chip memory, to keep the latency low. Design tools for implementation on FPGAs, however, provide some relief since they employ optimization techniques to reduce the overall memory footprint. While this is working quite well for single networks, it is more difficult to achieve a high reduction when using more weights, that share the same computation resources.

Regarding the usage as a trigger system, the most important criteria are the number of specialised networks that can be used in parallel at runtime. Events arriving at the trigger hardware can be classified in different groups according to multiple criteria, for example, the geometrical space of the detector in which they are occurring. Another example is the presence of signals in the detector due to operational anomalies. In such a case, it might be possible that a particle cannot be tracked continuously across all of the detector's layers as one layer might not produce usable data. One approach to achieve good results, even for these cases, is to use neural networks that were specifically trained for those cases. Limited on-Chip memory hereby limits the number of feasible specialised neural networks.

4.1.6. Runtime Adaptivity

The selection of training data determines the classification capability of a neural network. It can be trained to classify data for all possible inputs. Such a training makes the network quite general, however it deteriorates the performance significantly for special cases. However, it is possible to train networks with a pre-selected set of data representing special cases that are occurring during detector operation. The resulting networks are then particularly well suited to classify exactly these types of cases. As a result, instead of a general network, many specialized networks can be used to increase overall classification by a system. This is often the case for the experiments considered here.

For an online classification on an FPGA, this means that several networks have to be stored simultaneously. At runtime, the received detector data is at first processed to determine which network is the most suitable for processing and thus to be loaded. As a

result, the entire architecture must be designed in a way that it can support this network selection and reloading of weights at runtime. Pipelining must be considered here to match all of the internal signal delays. This adaptivity of the network configuration also reduces the capabilities for optimizations, for example, pruning at the hardware level in which network connections are removed. Additionally, while it is possible for a single network to reduce the bit width of single weights, this cannot be done easily anymore in the case that several different weights are to be potentially loaded into the registers.

4.1.7. Design Time Flexibility

The basic processing architecture developed within this thesis must be flexibly adaptable to the changing behaviour of the experiment. In the considered application cases, the exact behaviour of the experiment is not known beforehand, and operation parameters have to be adjusted over time to reflect the current status. For example, the luminosity of the experiment is not achieved immediately but rather increased over time, which will impact the observed data patterns. A previously trained neural network must then be retrained and configured for the prevailing operating conditions. This has to happen quite often and especially over a long period of time, as Belle II will be operated beyond 2021. To address this, the reconfiguration must be preferably performed without too much effort.

Flexibility must also be ensured for transferring the architecture to an alternative updated FPGA platform bound to be used in future operation. Due to the long runtime of the experiment, there will be hardware updates over time to keep up with the technological state-of-the-art. Here, also flexibility in the processing is advantageous, for example, by providing parametrizable pipeline stages or a configurable degree of parallelism within the individual processing stages. At the same time a flexible architecture coupled with flexible algorithms, can be used to fine tune the parameters to achieve the desired functionality and performance. For example, the neural network or preprocessing could be downscaled to upscale or introduce other modules required to keep up the experiment's development.

Considering online data processing systems within a particle detector experiment, it is also important to note that these will be updated over time. New components will be developed or updated over the entire duration of operation. Here a system, as is the case for the trigger system, that is in the middle of the data processing chain is depending on many potentially changing interfaces, components, and services. To be up to date with recent changes, it must be flexible to be adapted quickly and easily. For this purpose, semi-automated tools are particularly important in order to reduce the implementation effort.

4.1.8. Summary

Trigger systems have, in general, hard real-time requirements for both the latency and throughput. Looking especially at the L1 trigger system, the latency budget is very restricted and allows only the usage of simplified algorithms to be executed. The complexity of the algorithm to be used meanwhile depends on the used FPGA, as its resource

budget is defining the reachable level of parallelism and clock frequency. Early versions of systems have to be planned and prototyped long before the first collisions. As a result, the hardware used at the beginning of operation is typically already outdated. However future iterations and upgrades allow an increase in functionality as well as an opportunity for consolidation.

Since trigger systems are combining data from multiple sources, they have to provide high-speed data transmission. From a design point of view, these data sources have to be synchronized for further processing. On a physical level, the high utilization of IO resources influences the place and route phase of the architecture. As resources are spread out and have to be combined at one synchronized location on the FPGA, care has to be taken to achieve timing closure.

Non-essential features of such systems improve its usability and ensure the correctness, such as flexible design and monitoring services. While an operational system can be achieved without these services, it will be impractical without them since validation will be close impossible, which is required for such a powerful system in the experiment.

4.2. Basic Architecture Template

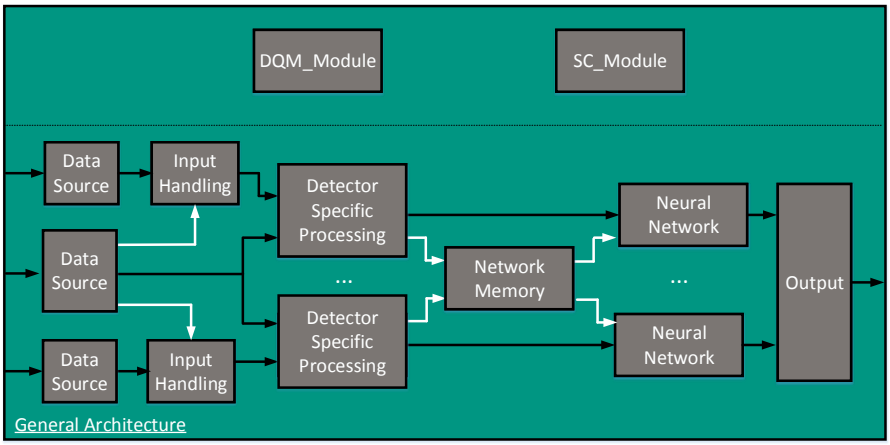


Figure 4.1.: Overlying architecture template for neural network-based trigger and data reduction systems. The control flow is indicated by white arrows.

Before delving deeper into each respective system that is presented within this thesis, a more abstract and general view on the realization of neural network-based trigger and data reduction systems is provided within this section. The basis for an architecture template to be used across all applications. The template used throughout this thesis is shown in figure 4.1. Here, data processing can be functionally divided into three stages, which are executed sequentially in a pipelined way. Pipelining is hereby essential for achieving

the required update frequency for both data reception and transmission since the overall processing latency is much higher than one transmission clock cycle. The responsibility of the first stage is the correct reception of input data, which is typically received from several distributed and heterogeneous processing systems. These systems typically have their own protocols, which need to be supported and can already include behaviour that is reflecting the functional principles of the connected detector. This requires additional processing that takes the detector into account, as well as mechanisms for compensation of the different latencies. Incoming data includes control flow information, which for example starts the internal processing or selects a network.

Received data is then passed to a preprocessing stage, which is, in this case, predominantly consisting of detector specific algorithms. Modern machine learning algorithms such as CNNs can partially avoid usage of preprocessing by compensating it with additional processing layers within the network. However, all the applications that are considered in this thesis are significantly profiting from the usage of a separate preprocessing stage, which typically allowed to keep the size of the network small enough to avoid exceeding the available resources of the used FPGAs. The preprocessing stage hereby receives its data from several possibly independent sources and transforms it into a representation that is optimized for usage by the neural network. Algorithms that are used at this stage are meanwhile strongly dependent on the detector's geometry and are mostly custom-tailored for this specific use case. This means that the possibility of using an already available IP core library is rather limited. In principle, different preprocessing algorithms can be performed in parallel. In most cases, separate input data streams for the later neural networks are generated. Before being used by the neural networks, the separately generated preprocessing data streams have to be synchronized. The results of the preprocessing are typically unveiling a more detailed view of the currently processed events. In many cases, received data is not perfect, especially during operation with collisions. Interference within the electronic environment of the detector might result in situations of high noise, which limits the capability of a detector to detect a particle correctly. A neural network that was trained with ideal operational conditions in mind might not be suited for operation in less than ideal conditions that are present in reality. One typical approach to deal with these situations is to provide special neural networks, which were trained under these conditions. In this case, several weight sets have to be stored and dynamically loaded during operation. The architecture is providing a network memory for this, together with a decision logic that selects the appropriate weights depending on the preprocessing's results.

After the preprocessing stage, the neural network algorithms are performed. Several instances can potentially be processed in parallel, with regard to the Belle II use case, these instances can estimate several possible track candidates that are delivered by the 2DS. The algorithms used here are independent of the actual experiment and its detector. Thus they can be implemented as reusable standard components being part of an IP core library, which is then only configured to the individual requirements. The last stage of the architecture is the postprocessing of the output that was generated by the neural network. Three examples of possible postprocessing algorithms are briefly outlined. First, a cut can be applied to the output that maps a large range of values onto a binary value, that is, for example, used to indicate the presence of a track that is possessing predefined characteristics. In parallel, a voting mechanism can be used, which calculates its decision based on

a combination of all the results generated by the networks. Another example is a mapping of the neural network output onto a probability, for example, the probability of an observed particle being the one that is searched for. Even though the processing stages are the most important part of such a system, it has to be extended by monitoring and validation structures. These are represented as separate DQM, and SC modules, which basically capture certain internal processing data and send it over the defined interfaces to data sinks responsible for online monitoring or validation.

While several design principles were applied for the development of the systems described here, two of them are mentioned separately due to their importance. These are designed for enabling retiming and for design time adaptability.

Design for Retiming

Most of the tasks to be performed throughout all of the processing stages within the architecture are data flow-oriented. At the same time, achieving high throughput coupled with low-latency processing, is targeted. While general optimization of logic and pipelining are already significantly contributing to achieving these goals, additional performance can be gained by facilitating the usage of retiming-enabling design throughout the entire architecture. Retiming itself is based on the realignment of internal register stages across the described logic operations [68]. The goal of this realignment is to balance out the maximum data path length between pairs of registers across all processing paths. Even though being a state-of-the-art technique for a long time in the digital design community, it can only be applied by design tools when the described architectures are fulfilling predefined conditions. All components developed within this thesis are hereby designed to comply with the design guidelines of the tool and FPGA vendors. For example, the FPGAs by Xilinx cannot perform retiming in case that registers with asynchronous resets are used.

Flexibility and Adaptability at Design Time

Scalability of the architecture is necessary for long-term use in an experiment. An initially found configuration is often only efficient for the currently present behaviour of the experiment possibly behaving badly during later stages of the experiment at which, for example, an increased luminosity is present. This is especially true when using neural networks as the algorithmic basis as they can be retrained to reflect the changing conditions. Due to this, all developed HDL modules are designed to be as flexible as possible. Such a design will allow the modules to be continuously adaptable. All constants that can change across iterations are decoupled from the functional implementation inside interchangeable packages. In case of an update, only these packages have to be changed, while the core architecture is preserved. In addition, parameters such as bit widths are defined, degree of parallelism and allocated resources as configurable parameters, as generics in VHDL, for all the modules to be easily adjusted.

4.3. Basic Design Flow Template

Flexibility requirements set upon machine learning-based trigger and data reduction systems require a semi-automated design flow in order to be efficiently fulfilled. Since the systems developed in this thesis are partially depending on the usage of custom algorithms and even different programming languages, design flows specific to the respective use cases were developed. However, all of them have general similarities that can be solved together by defining a design flow template that serves as a reference. The developed template is hereby illustrated in figure 4.2. In this illustration, individual design steps are shown in separate boxes. The entire design flow can be divided into two domains of design. These domains are the development of the algorithms and the FPGA-based design. The general design flow is further described in the following.

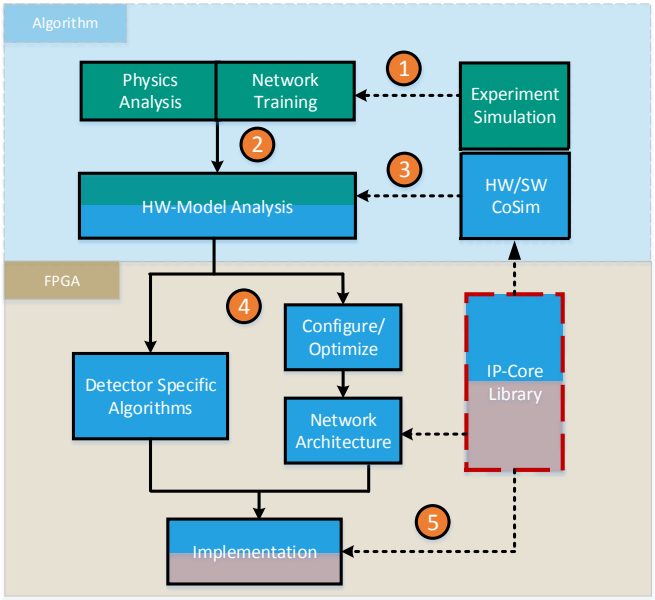


Figure 4.2.: Design flow template that serves as a reference for all derivatives used across all of the developed machine learning based systems of this thesis.

The basis for the development of suitable algorithms in the domain of a particle detector experiment is a simulation framework that models its behaviour. With the help of this framework, it is possible to investigate the capabilities and parameters of an algorithm well in advance without the detector being available for operation. Particle collisions are simulated for this, together with recreated noise and modeling of the expected interaction between the materials, which are used within the individual detectors, and the expected created particles. The simulation will not recreate real operation accurately from the be-

ginning, however, the assumption is that real operation can be recreated by adjustment of parameters to achieve a high accuracy. Under this assumption, the evaluations carried out at the simulation stage are representative of the later experiment. With regard to the use of neural networks, two steps are carried out at the stage of algorithm design. The first step is the definition of the input data to be used by the network. This includes the investigation of suitable representations that can be derived from the targeted detector's data. The algorithmic description for generating the input values using the detector's data is corresponding to the preprocessing performed at the system. At the same time, this definition is performed together with the definition of the neural network's parameters, for example, its internal topology.

The simulation of the experiment and its detectors is meanwhile providing insight into the effectiveness of an algorithm that is under development. To move closer towards the proof of functional correctness, the data readout system of the detector has to be taken into account, for example, the CDCTRG. Representative behaviour of this system requires the inclusion of all its essential sub-systems. Within Belle II, this kind of simulation is performed by Trigger Simulation (TSIM), which represents a software-based simulation that includes all sub-triggers and can be coupled with the experiment's simulation. As a result, the real data flow of the detector readout can be taken into account to add more detail to the development of the algorithm. For example, the behaviour of the FEE, which reads out the detectors according to a defined scheme, is emulated.

Before the hardware-based development starts, it is helpful to investigate key parameters for the implementation on FPGAs beforehand on the basis of the established simulation. These are, for example, bit widths to be used internally and their impact on the overall performance of the algorithm. This can also be investigated later on, on the basis of hardware-level simulation using an HDL, however, this is very time-consuming as it is on lower levels of abstraction. The approach taken here is to implement a model of the FPGA's internal processing to be used for fast exploration of the key parameters. These models are then validated against the HDL implementation using HW/SW Co-simulation in order to ensure accurate recreation.

The transition towards the FPGA development is realized by a split approach for the processing modules of the machine learning algorithm and the preprocessing. Preprocessing algorithms are in general detector-specific and must be custom-designed. Thus they cannot be reused across different applications. Machine learning-based algorithms, on the other hand, are general but have to be configured for the respective use case. They are hereby organized in a library of optimized IP cores. The goal of using this library is to facilitate reuse are across the boundaries of applications. They are designed to be highly flexible and parametrizable in order to be adaptable to changing circumstances. In addition, a validated Software (SW) model is provided for each IP core in order to allow a quick evaluation of the algorithms with high-level details about the hardware implementation. For each IP core, an estimation of its characteristics, such as achievable frequency and estimated resources are generated beforehand. These estimates are based on synthesis results for a particular configuration that can be scaled according to the requested parameters. Alternatively to these scaled estimations, HLS tools can be used to generate a potentially more accurate estimation. However, they require longer processing times to

generate the estimation. The advantage is here to have an early and quick estimation of the characteristics at the stage of algorithm exploration.

4.4. Design of Neural Networks for FPGAs

The two main systems developed within this thesis are using algorithms based on neural networks. The basic processing principle is hereby based on MLPs, for both approaches. As they are shared, a fundamental investigation of the implementation of MLPs on an FPGA is carried out in this part of the thesis. The results found here are reused in the corresponding chapters 5 and 8 that feature the concrete systems.

4.4.1. Realization of Low-Latency and High-Throughput Neurons

The algorithmic core of an MLP is the artificial neuron that consists of a set of MAC operations. A block diagram of such an operation is shown in figure 4.3, together with the possible design parameters for the implementation. The number of operations to be performed by each neuron corresponds to the number of input values that it has to process. Typically, a neuron is including an optional bias. However, this is not requiring additional multiplications since it is not a weighted input variable. It is then more efficient to just treat it as an additional constant to be added at the end. At this stage, there are already many degrees of freedom for the design targeted for FPGAs. The optimal implementation choice depends on the characteristics of the network, such as the bit widths used for all values or the need to load different weights. Especially the reduction of memory accesses for new weights is one of most popular optimization goals, however this is not considered here since all weights are loaded within one clock cycle when using on-chip memory.

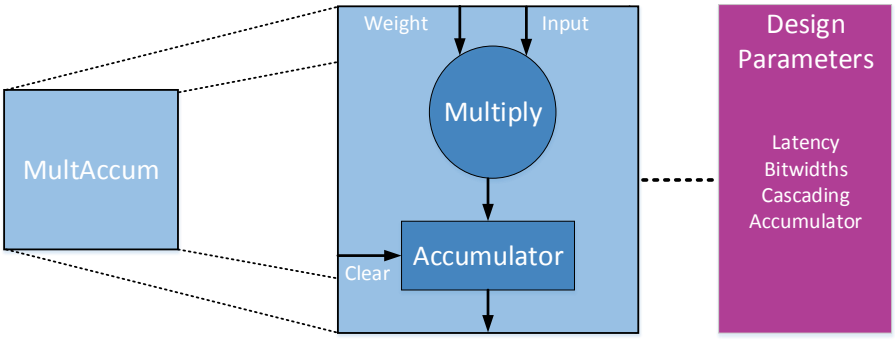


Figure 4.3.: Architecture of a module performing MAC operations as they are present in an artificial neuron.

On modern FPGAs there is the option to implement artificial neurons either based on Slices or DSPs. DSPs are a particularly efficient choice in case that multiple different weight sets are to be loaded during runtime. These units are equipped with an optimized Hard-IP implementation of a general multiplier that is supporting a predefined range of bit widths. In addition to that multiplier, each DSP is already equipped with an accumulator that can be used to sum up the weighted inputs. As a result, the entire MAC operation can be implemented within a DSP unit. The optimized integrated multiplier allows DSPs to achieve the highest throughput and lowest latencies for general operation. Further supporting the usage of DSPs is the fact that multiplications with variable weights can only be realized rather inefficiently when using Slices.

However, for the cases in which weights are constant and not to be reloaded during runtime, a realization based on Slices can represent the preferable implementation choice. Instead of providing resources for general multiplications, they can be implemented for a specified constant. This dedicated implementation is leading to an even more efficient solution than general multiplication based on DSP usage. In addition to this, the implementation based on Slices can be routed easier and allows for better logic optimization due to their large-scale availability and finer granularity.

An implementation of neurons is also strongly dependent on the selected bit widths. A DSP slice provides a fixed bit widths for both of its input ports. On a Virtex-6, these are limited to 18 and 21 bits. As soon as the supported bit widths are exceeded, additional DSPs are required for processing. In addition to the increase in DSP slices, additional resources are required for the interconnection between the additional slices. This interconnection is, in turn, increasing the overall latency due to the added signal delays for combining the partial results of the multiplications.

A general disadvantage of an implementation using DSPs is their limited availability on an FPGA when compared to the more widely available Slices. In addition to this, they are less commonly distributed across the entire FPGA and mostly arranged in fixed columns on the floorplan. While this type of architecture is facilitating cascaded operation, the columns may be far away from their data sources. As a result, routing of the signals might infer high signal propagation delays that are increasing the initial latency. Another source for problematic implementation using DSPs is occurring in case numerous inputs are to be processed by the same unit. This is the case for neuron processing when the same DSP processes several inputs. Routing congestion can occur here, when they are obtained from many multiple sources and optimization of the multiplexing network needs to be performed.

The arrangement in columns, however, has one important advantage to be considered. That is the option for cascaded processing in which data is passed across several neighbouring DSPs using a direct connection. This stands opposite of the traditional communication path using the general routing resources. Due to this optimization, cascaded processing is capable of achieving the highest frequencies, even reaching up to one GHz, which is highly advantageous for high-throughput operation.

Realization of MAC Operations for High-Throughput Processing

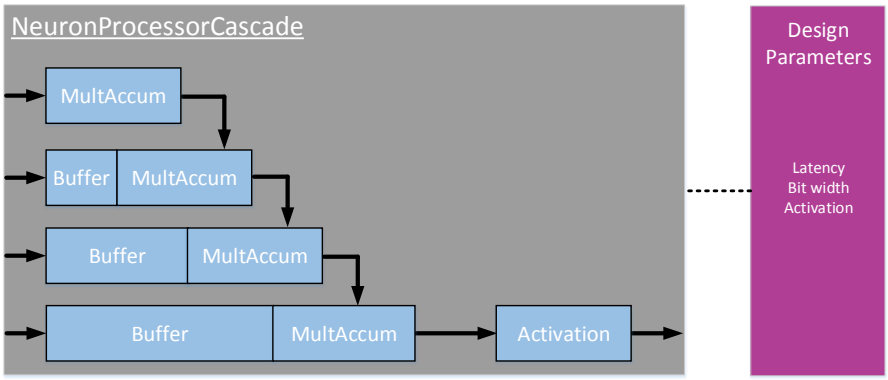


Figure 4.4.: Architecture of an implementation of MAC operations that is optimized for high-throughput operation.

When using cascaded operation, the communication is performed by using the two dedicated ports PCIN and PCOUT that represent a direct connection between DSPs located adjacent to each other on the FPGA. With regard to the processing of a neuron, a weighted input can then be sent to the neighbouring DSP so that it can add the value to its locally calculated weighted input. The PCIN input port is hereby directly connected to the accumulator of the neighbouring DSP, which is providing the lowest signal propagation delays for data exchange. Overall processing can then be implemented in a pipelined way with a total latency of two clock cycles. In this case, the multiplication is carried out in the first clock cycle. The subsequent addition is then performed in the second clock cycle. At this point, however, a data dependency arises that limits the minimum latency. When using cascaded operation, every DSP, besides the first one in the cascaded chain, must wait for its predecessor to finish processing of its partial sum. Overall, the highest possible throughput and the best resource utilization are achieved using this communication path, while the latency is increasing with the number of inputs to be processed. The internal structure of this implementation is shown in figure 4.4. It is indicating the process of partial sums being passed across several MAC units with buffering elements being used for compensation of the present data dependencies.

Realization of MAC Operations for Low-Latency Processing

In case a system is required to be optimized for primarily low-latency operation, an as soon as possible scheduling is generally yielding the best solution irrespective of resource demand. Here, all operations are performed in parallel as soon as their respective data dependencies are resolved. With regard to the processing of artificial neurons, all multiplications are being processed in parallel. The weighted inputs are then summed up afterwards in a separate accumulation unit. This unit is then typically implemented as an adder tree to keep latency as small as possible. In the best case, one DSP slice can

perform one such multiplication within one clock cycle. The accumulation step's latency depends on the number of weighted inputs to be summed up, coupled with their chosen bit widths. Typically, when using a tree structure, at least 32 inputs can be accumulated within one clock cycle while using a clock frequency of 127 MHz. In total, that means that the lowest achievable latency for the entire MAC operation is at two clock cycles.

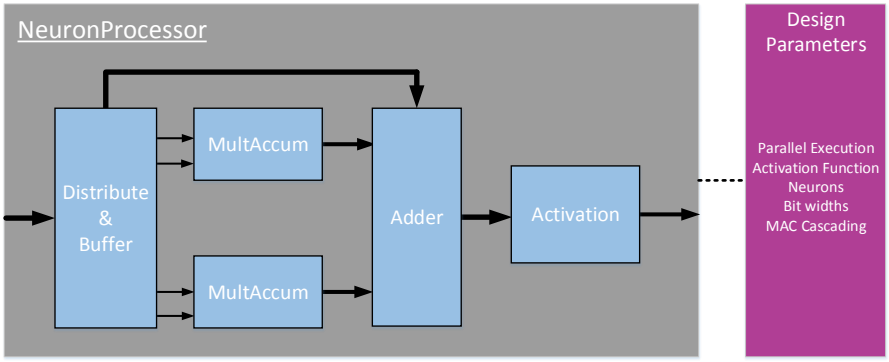


Figure 4.5.: Architecture of a MAC realization for low-latency operation.

Using this approach, however, it is not possible to exploit the advantages of cascaded operation, thus resulting in lower overall clock frequencies. At the same time, there is a higher resource demand due to the required adder tree, which can otherwise be implemented within the accumulators of the DSPs. In addition, the latency of the tree realization depends on the number of inputs. With high parallelism, the number of adder stages required increases as well, further reducing the maximum clock frequency. In order to maintain a defined frequency target, additional register stages and clock cycles may have to be allocated.

Multiplexing of MAC Operations for Increase in Resource-Efficiency

A good compromise between latency, resources, and routability can be achieved by employing time-multiplexing for the execution of MAC operations of a neuron when using a constrained set of DSPs. Instead of allocating exactly one DSP for each input of a neuron, a DSP can be designed and scheduled to process a subset of all inputs across several clock cycles.

Using time-multiplexing will increase the latency for processing an entire neural network compared to full spatial parallelism but offers the opportunity to reduce the number of required resources significantly. In addition to this, it influences the subsequent routing to be performed by reducing the number of signals to be transported in parallel. Meanwhile, it is reducing the subsequent adder trees by reducing the number of partial sums to be processed. At the same time, this approach can make use of the DSP's internal accumulator by summing up weighted inputs across multiple clock cycles without using the dedicated adder resources. The following adder tree then has to add only the partial sums

instead of all weighted inputs, which further reduces resource demand. However, by limiting parallel processing, the overall latency is increased. Additionally, this approach also requires additional registers for storing input values and weights across multiple clock cycles together with multiplexers to switch inputs. Furthermore, a controller for directing the multiplexers across clock cycles is necessary.

In addition to time-multiplexing of MAC operations by reusing the same processing resources, neurons themselves can also be time-multiplexed. In this case, neurons of the same network are processed on the same resources at different clock cycles. The advantages and disadvantages are the same as for the multiplexing of inputs, however, this allows to finish processing of a group of neurons at an earlier stage when compared to starting the processing of all neurons immediately. This characteristic opens up new opportunities for additional pipelining, which will be discussed in the following. Both multiplexing variants are shown combined in the schedule presented in figure 4.6. In this schedule, dotted lines are indicating different clock cycles. Operations are meanwhile multiplexed in time by processing all input values of a single neuron within three consecutive clock cycles, with the next neuron being processed afterwards. At the same time at which the next neuron is processed on the DSP slice, both the adder tree and the activation function are processing the previous neuron that is allowing an overlap in time with the multiplications of the subsequent neuron.

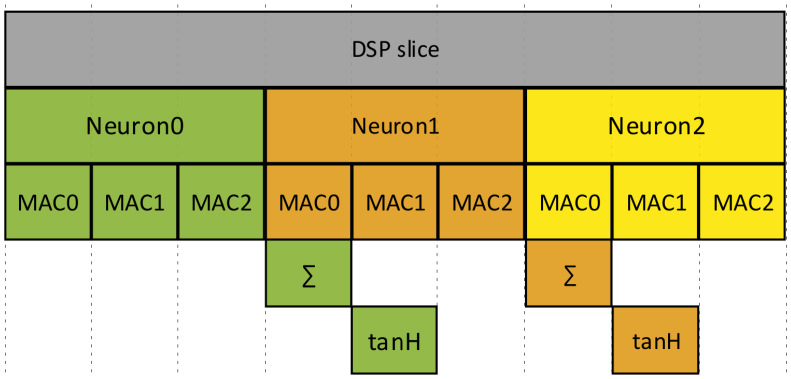


Figure 4.6.: Schedule of a time multiplexed DSP. MAC operations of the different neurons are performed on one DSP unit at different time intervals. Both the adding and activation function can be interleaved with the processing of the next neuron.

Summary

Three alternatives with their distinct trade-offs are available for implementing a neuron. They are summarized together with their latency, throughput, and resources in table 4.1. Within this thesis, only the most aggressive low-latency option without any multiplexing was not used since it easily exceeded the resources present on the available FPGAs. Multiplexed operation is hereby used throughout the versions of the NNT, presented in

section 5.2.5, to find a suitable trade-off between low-latency processing and resource demand. Meanwhile, the high-throughput design is used for the OCA, described in section 8.2.2.2, since throughput is representing the strictest requirement to be fulfilled.

Implementation	Latency in cycles	Throughput	Resources
High-Throughput	Inputs+1	300 - 700 MHz	Inputs*DSPs
Low-Latency	1 + adder tree stages	100-200 MHz	Inputs*DSPs + Adder tree
Multiplexed	Inputs /fixedDSPs	100-300 MHz	fixedDSPs + reduced Adder

Table 4.1.: Listing of the used implementation options for the MLP.

4.4.2. Low-Overhead Realization of the Activation Function

A direct translation of the mathematical function used for activation is typically very compute- and resource-intensive when implementing for the resources available at an FPGA. Two popular variants of these are the TANH and the sigmoid function. In both variants, divisions are necessary for the calculation, which can usually only be carried out inefficiently. Modern networks use simplified functions like ReLU to avoid this problem, however the networks used throughout this thesis are relying on the non-linearity of classic activation functions in order to achieve a high resolution for the z-Vertex [92].

In order to keep both the computational and resource intensity low, the implementation strategy within this thesis is to implement activation functions as LUTs on the FPGA. The effectiveness of this approach depends on the bit widths used for both the address and data. These bit widths are, in turn, depending on the required resolution. For large address spaces, the realization as LUT is not scaling well and quickly becomes very inefficient as the demand for resources increases exponentially. For example, a BRAM on a Xilinx FPGA supports the storage of data up to 36 Kbit. In case that a function to be stored is exceeding the maximum provided storage, additional BRAMs must be allocated to accommodate the required storage. For the targeted FPGAs from Xilinx, unreasonable resource overhead was reached with address spaces larger than 4096 entries. In such cases, an alternative solution is used. The alternative to implementation as a LUT is the usage of partial linear interpolation between defined points. While being more resource-efficient than the storage of the precise result, the accuracy is degrading.

Two activation functions are considered within this thesis, the TANH and sigmoid, both of them being point symmetric. This characteristic can be used for optimization by only storing the positive values, while negative values are calculated afterwards by inverting

the positive value. By employing this approach, additional logic is necessary for the inversion. However, the demand for memory is cut to half. For address spaces above 1024 entries, as is the case in this thesis, it proved to be more efficient to use inversion logic instead of storing negative values.

Further optimization can be achieved by taking into account that the activation function is the same for all neurons within one layer. Meanwhile, BRAM on FPGAs is often architecturally designed to provide a dual-port option. The optimization is then to share single BRAM instances across pairs of neurons. Dual-port operation hereby allows using fully parallel access from two separate sources. Typically this gets problematic in case of parallel write and read access. However, since it is only implemented as a read-only LUT, such situations cannot occur.

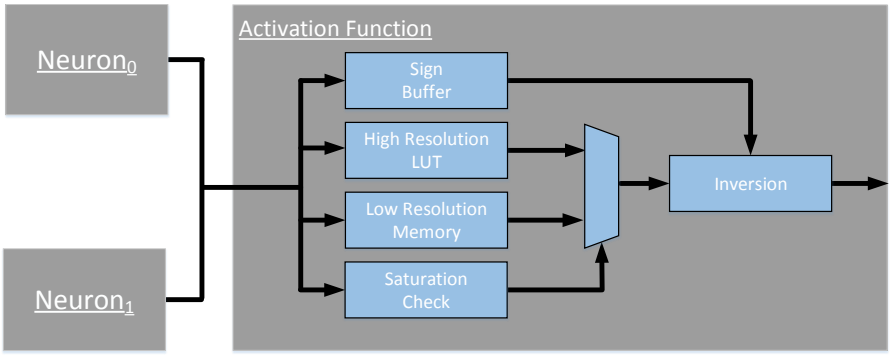


Figure 4.7.: Architecture of a LUT-based implementation with the described optimizations.

Depending on the activation function to be used and the targeted bit width, certain address spaces might only require a low-resolution implementation in order to achieve good or even exact results. An example is the TANH function when using a 10 bit fractional representation, as is the case for the NNT. For this configuration, high values are saturating and are nearly constant across most of the upper address space. Instead of storing the function in the LUT uniformly for all possible input values, a case distinction is implemented that identifies address spaces that are only requiring a low-resolution implementation. In the case that the current input value is above a certain threshold, the output of the function is not determined by using the BRAM. Instead, an alternative path is used, which consists of additional parallel registers that contain the result for predefined address ranges. The idea here is to use just a few registers to cover large ranges of the function instead of allocating excessive memory space. This approach will introduce some computational overhead since logic for case distinctions is necessary. In a function like the TANH, however, only a few cases have to be checked when data is limited to a bit width of 10 bit fractional. The savings in memory demand easily make up for that overhead, as it proved to be the most efficient implementation. The architecture of a BRAM

implementation with all optimizations is shown in figure 4.2. All used implementation and optimization options are additionally summarized in the tables 4.2 and 4.3.

Implementation	Description	Effect	Conditions
Look up table	Pre-calculated activation function	No processing Precise	At most 4096 entries
Linear Interpolation	Interpolation between pre-defined positions	Balanced resources and performance	More than 4096 entries

Table 4.2.: Listing of approaches towards implementing the activation function.

Optimization	Description	Effect	Conditions
Dual-Port Access	Resource sharing across neurons	Memory reduction	Reusable activation function
Inversion	Storing only positive values	Memory reduction Inversion logic	At least 1024 entries
Saturation	Separate low resolution memory	Memory reduction Saturation logic	Low resolution address space

Table 4.3.: Listing of used optimization strategies for the activation function.

Examples for the usage of both implementation options, LUT, and linear interpolation, are presented within this thesis. LUTs are used for the NNT presented in section 5.2.5, while linear interpolation is being used for implementing the sigmoid function that is required for the OCA as described in section 8.2.2.3.

4.4.3. Pipelining Options for Resource-Efficient Neuron Processing

When employing time-multiplexed processing for not only MAC operations but neurons within a layer of the network, additional options for pipelining across different layers are opening up. Processing in successive layers of the neural network requires the presence of the results generated by its connected input neurons from the previous layer. Internal processing can, however, already start when only a subset of all results is available. The main idea is now to interleave operation between successive layers, as presented in figure 4.8, to optimize general operation. In this, time multiplexing of neurons is applied at each layer. This leads to different times of arrival for partial results generated by a neuron within the hidden layer of the network. The subsequent neuron from the output layer is then operated in a way that it is processing these partial results before the remaining results of all neurons are ready. By doing such interleaving of layers, the overall latency can be reduced since the interleaving clock cycles are saved. On the other hand, processing within the output layer can be stretched across several clock cycles, as illustrated in figure 4.8. The possible options are listed in table 4.4, with the simple pipeline representing no interleaving operation between layers and the other options representing either the

resource or latency optimized option. The pipelining methods described here are based on the master thesis Ref. [Poe18].

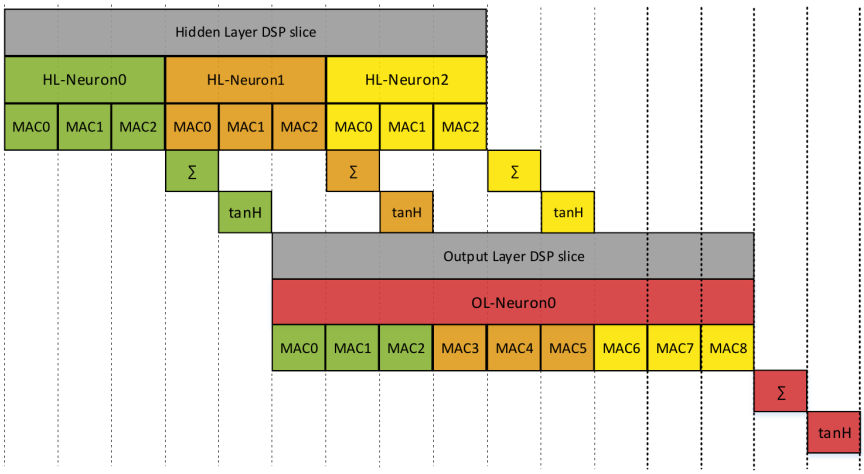


Figure 4.8.: Schedule for pipelining across neural network layers when using time-multiplexing of neurons.

Implementation	Resources	Latency	Throughput
Simple Pipeline	$Neuron_{total}$ DSPs at OL	Inputs/fixedDSPs	200-300MHz
Resource Pipeline	$Neuron_{mux}$ DSPs at OL	No change	No change
Latency Pipeline	No Change	Interleaved clock Cycles saved	No change

Table 4.4.: Listing of used implementation options for pipelined operation of the MLP.

4.4.4. Network Architectures based on Heterogeneous Resources for Increase of the Performance

So far, all presented designs for implementing neurons were based on DSPs. This is mainly due to their higher efficiency when using variable inputs compared to Slice-based implementations. Even though they represent the best solutions with regard to all non-functional aspects, DSPs are rather scarce on modern FPGAs. In case that Slices are in

less demand and spare resources are available on the FPGA, a heterogeneous architecture using both resources can be used. Such heterogeneous architectures can help with spreading the processing demand across all types of resources. Due to the different characteristics of both, the implementation of such architectures is not straightforward. For such an architecture, mainly the differences in achievable operating frequencies and resource inefficiency have to be addressed. First, it is better to separate network layers into the type of resource to be used in order to achieve uniform characteristics within the layer and minimize the clock slack. In addition, it allows using separate clock domains at the intersections of layers with different resource types in order to bridge the difference in frequencies. The second measure taken to implement such architectures is to use Slices exclusively at the output layer of the network. These layers are consisting of much less MAC operations, compared to hidden layers at, for example, the NNT that is featured within this thesis. As a result, a smaller amount of MAC operations are implemented in resource-inefficient Slices compared to implementation in other layers, thus the overall penalty is kept low. However, this is strongly dependent on the used network topology, with the topology used at the NNT representing one network in which it is beneficial. The resulting heterogeneous architecture is shown in figure 4.9, with the clock domain crossing being addressed by using FIFOs at the intersection. It is hereby assumed that the preprocessing is most likely achieving a lower operation frequency than the DSP-based neurons, as they can achieve the highest possible frequencies on the FPGA. Such heterogeneous architectures are mostly explored within the investigation of future operation of the NNT featured in section 5.4.4. The heterogeneous architecture that is described here is based on the master thesis Ref. [Poe18].

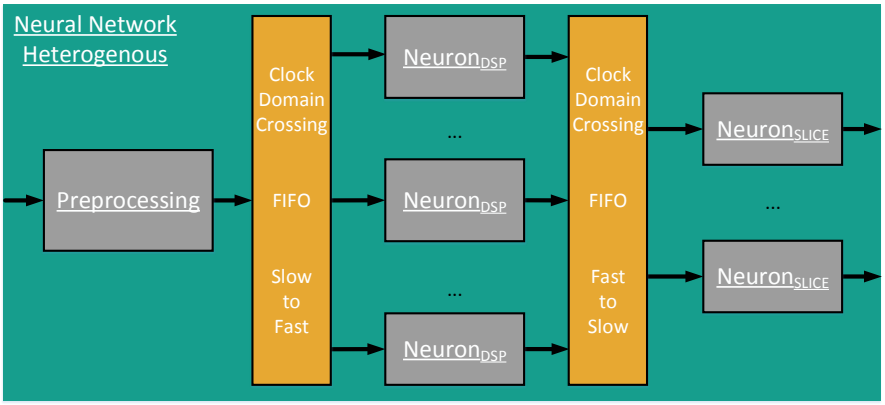


Figure 4.9.: Implementation of a two-layer MLP using both DSPs and Slices. In this case the layers have to operated within different clock domains, for this additional modules supporting the clock domain crossing are used.

4.4.5. Summary

Multiple different design strategies for bringing neural networks onto FPGAs were discussed within this section. Artificial neurons represent the basis for all processing within the layers of the networks. These already have many implementation alternatives on modern FPGA architectures. The best operational characteristics can be in general achieved by relying on DSPs since they are providing the highest operation frequencies, lowest latency and are already equipped MAC optimizations such as an internal accumulator as well as cascading data transfer. Neurons can generally be mapped either onto a cascaded or a tree architecture. While the former is achieving the highest frequency and is thus suitable to achieve high throughput, the later is achieving the lowest latency at the cost of throughput and resource efficiency.

Neural networks can easily exceed the number of available DSPs on an FPGA when using these architectures as these resources are fairly limited. To compensate for this, time-multiplexing of inputs, as well as, neurons themselves, can be applied. This effectively allows stretching the overall processing across multiple clock cycles while significantly reducing resource consumption.

Besides the heavy demand for MAC operations, each neuron has to implement an activation function. For the use cases investigated within this thesis, it proved to be infeasible to follow the state-of-the-art approaches that are simplifying the function in order to ease implementation. The activation function is not reduced to a simplified model but rather fully implemented as either a LUT or based on using linear interpolation. The former solution is using multiple optimizations that make use of the inherent capabilities provided by BRAM resources that are typically widely available on an FPGA as well as the function's characteristics.

An optimization strategy addressing the overall architecture that is discussed is the usage of pipelining across separate layers of the network in a way that multiple layers are processing the same input data set concurrently. This strategy can be hereby used to either reduce overall latency or resources by again stretching processing across the newly saved clock cycles.

The presented strategies are implemented and used for the later use cases of the NNT, discussed in chapter 5, and the OCA that is discussed in chapter 8.

5. The Neural z-Vertex Track Trigger

This chapter discusses the FPGA-based realization of the NNT for the Belle II experiment. Initially, only the requirements dictated by the experiment and the trigger system were provided. The tasks of selecting a suitable FPGA platform, developing an architecture, and defining the implementation process were all established as part of this thesis. These aspects are examined and discussed in this chapter. Since the NNT evolved throughout the development process, multiple used setups that were used in different stages of the experiment's operation will be presented.

5.1. Background Suppression using a Neural z-Vertex Estimation

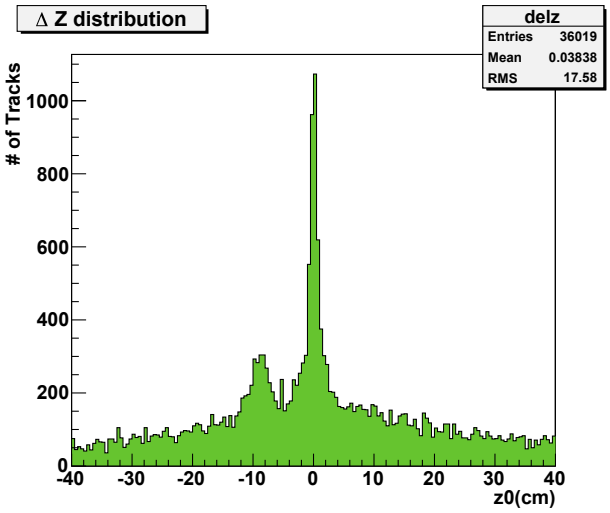


Figure 5.1.: z-Distribution from the Belle experiment showing the recorded background events represented by the peaks outside of $z = 0$ [5].

A consequence of the increased target luminosity for Belle II is the increased expected occurrence of background events compared to its predecessor. Background events are hereby not of interest to the search for new physics and are often not even part of the intended experiments. One of the most significant contributors to background events are particles that travel through the detector but have their origin well outside of the

interaction point. These were already a problem at Belle, however there was no online processing mechanism that took the origin relative to the z-axis into consideration. The problem is highlighted by the z-Vertex distribution of particle tracks that were recorded using a random trigger. This distribution is shown for the Belle experiment in figure 5.1. Here, most of the recorded particle tracks had their point of origin at around z=0, which represents the interaction point from which sought after decays are originating. However, a significant amount of the observed tracks are well outside of the interaction point. This, for example, can be seen by another maximum at around z=-10 cm. These tracks do not relate to collisions, but are rather the result of unwanted effects. Optimally those tracks can be detected and suppressed in order to reduce the outgoing data rates, which have to be below the maximum rate supported by the DAQ. The dominant part of the background events can be attributed to Touschek and Beam-Gas effects, which were introduced and shortly described in section 2.3.2. Looking at early results from Belle II operation show that the same situation is present here as shown in figure 5.2.

In order to suppress such tracks, algorithms for the reconstruction of the z-Vertex using detector data were investigated for Belle II. These are based on the data recorded at the CDC, which can be used to reconstruct a track in 3D. The main algorithm used for this reconstruction is based on a conventional approach that was described in section 3.2. In parallel to this approach, the NNT was developed to address this task by applying machine learning approaches, which are expected to fare well with the currently unknown background events. Both approaches are hereby pursuing the same goal and have the same requirements.

The advantages and reasons for using a neural trigger are that the algorithm is easily adaptable to the current behaviour of the experiment and that it is expected to be more robust when dealing with data that was not seen before during the training phase. The neural network can meanwhile be continuously re-trained with data from the experiment.

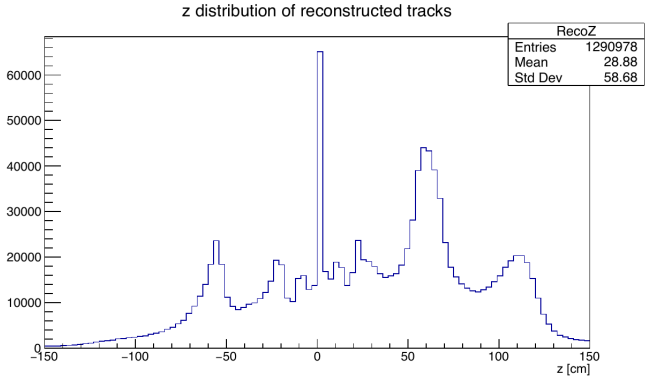


Figure 5.2.: z-Vertex distribution recorded from an early run during experiment 7 of Belle II is showing a similar but more pronounced distribution.

5.1.1. Estimation of Efficiency and Network Topology Studies

An investigation of the usability of neural networks-based algorithms was performed in Ref. [92]. Different methods were considered here, whereby MLPs showed particularly good properties. In addition, a suitable preprocessing was investigated, in which suitable input values were calculated using the detector's data. Additionally, different topologies of the MLP were explored and characterized. The results of the investigations are summarized in the following. At first, an efficiency metric is defined in order to assess the performance of the MLP. The efficiency is hereby described as the ratio between the number of tracks whose actual origin lies within z = +/-6 cm around the interaction point and the number of tracks for which the neural network is estimating the z-Vertex correctly into the same range. The definition of efficiency is meanwhile independent of conditions under which the observation took place that is independent of the used data set, either simulated or experimental.

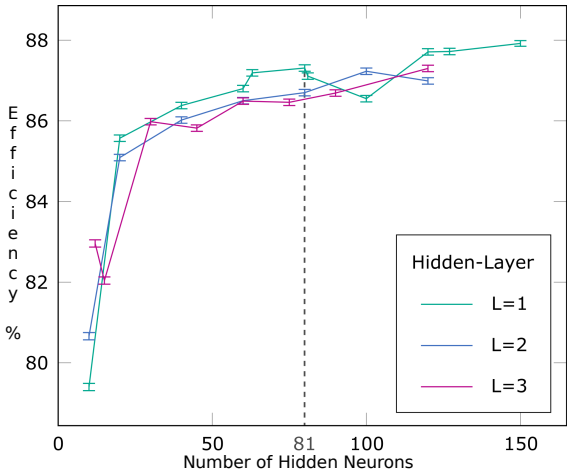


Figure 5.3.: Plot of the efficiency for MLPs estimating the z-Vertex. It is shown for different numbers of neurons and hidden layers [92].

The results for the achieved efficiency of the MLP are shown in figure 5.3. Different topologies are shown here, exploring the different options for operation. Looking to the results, it becomes apparent that the network becomes more efficient with an increasing number of neurons. Looking ahead to the resources of the available FPGAs, a network with 81 neurons and one HL was chosen as the reference topology. The total amount of MAC operations of this topology would demand around 90% of the DSP resources when using time-multiplexing on the UT3, which would theoretically achieve feasibility for implementation. Meanwhile, this network achieves an efficiency of around 87% [92]. Although adding more neurons improves the efficiency, the improvement is rather marginal, which leads to using the 81 neuron version for first iterations of the NNT. However, increasing the number of neurons can be investigated for future operation after the initial system was set up.

5.1.2. Functional Description of the neural z-Vertex Trigger

Before getting into the details of the FPGA-based realization of the trigger, the functional description is presented and discussed.

5.1.2.1. Preprocessing

In its first version, the trigger was based on using the data from the TSF directly as input values for the neural network. Each individual TS represented an input node of the network, while the drift time was used as the value. From the realization point of view on the FPGA, this required a large amount of MAC operations per neuron, as shown in equation 5.1. For one neuron, 2338 MAC operations would have to be performed. Assuming that the used FPGA can actually only calculate about 600 MAC operations per clock cycle, as is the case for the available FPGAs, such a high number of inputs cannot be realized within the targeted latency budget.

$$\begin{aligned} MAC_{Neuron} &= \#Inputs + 1 \\ &= \#TS + 1 = 2337 + 1 = 2338 \end{aligned} \tag{5.1}$$

An alternative approach to the initial idea was to reduce the number of inputs by preprocessing the data from the CDC as a first step. Here the first approach was to partition the geometrical space of the CDC that is processed by a network into a set of sectors. These sectors then only contained about 20 to 30 TSs each, instead of the entire 2337 TSs. For each sector, a separate net was trained, which had to be stored on the FPGA. However, the resulting memory space required to implement this, is larger than the available on-Chip memory on the available FPGAs. As a consequence, this setup relied on using external memory to be realized, which, however, added additional latency for the transfer of the weights.

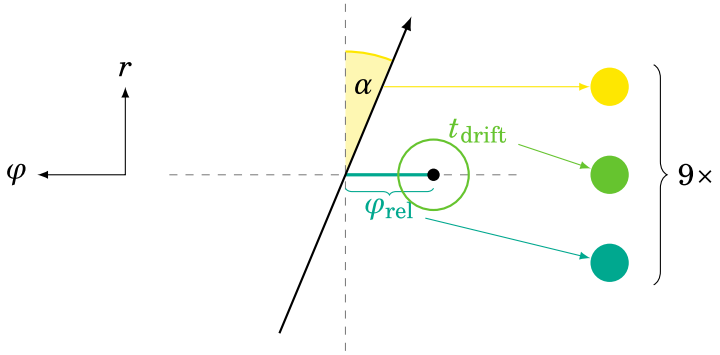


Figure 5.4.: Geometric depiction of the three input signals of the MLP relative to the wires of the CDC [92]. These inputs are used in the operational systems developed in this thesis.

Name	Description
alpha	Crossing angle of the track relative to the normal of the crossing point of the track with the circular path of the layer.
$\pm t_{drift}$	Drift time of the TS. The sign indicates the direction of the passing particle either left or right. It is not used in case of an unknown direction as defined by the TSF.
phi_rel	Azimuth angle of the particle relative to the angle of the sense-wire.

Table 5.1.: Listing and description of the three input values used for each of the SLs [92].

A better approach was developed afterwards in which input values that are optimized for the network were generated from the CDC data. For this purpose, the active TSs within a SL were transformed into triples of input values. These values are listed together with a description in table 5.1. Their graphical representation relative to the CDC is additionally shown in figure 5.4. Here, instead of considering just the TSs on their own, their relationship to both the event time and a found 2D track are considered. For this, all TSs are at first matched to a 2D track that was found by the 2DS. This matching is hereby already partially provided by the 2DS as it relates the found track to the axial TSs that are located the closest. For this approach, the matching is extended by including the previously unused stereo TSs. The matching procedure that was developed checks whether an active TS is geometrically within the vicinity of the estimated 2D track. Only if it is close enough to the track, is the TS considered for estimation of the z-Vertex.

Two of the defined input values for the MLP are afterwards calculated on the basis of the found TS relative to the estimated track. These values are the crossing angle alpha and the relative azimuth angle phi_rel. These two variables are then supplemented by the drift time of the TS. Using this triple of input values allowed to reduce the size of single neural networks and subsequently to fit it entirely into the on-Chip memory. This allowed achieving reasonable performance, both in terms of efficiency and processing latency.

5.1.2.2. Configuration of Networks at Runtime

Training a single neural network for operation will yield a solution that can achieve reasonable performance for all possible input combinations. Further improvements, however, can be achieved by training additional networks that are specialised on specific subsets of the possible input data. This can be used to address situations in the experiment, in which a general network is achieving a rather poor efficiency. This opportunity was additionally explored to improve performance [92] further.

The first step for this is the definition of the subsets of data for which specialised networks shall be deployed. One approach is partitioning in the geometric phase space of the CDC. Depending on the part of the CDC in which a track was observed, different networks are to be loaded. A better approach to this is to train dedicated networks such that they could compensate for the absence of a viable TS in one of the stereo SLs. For this purpose, four additional networks were trained. Each of them is specialized in compensating the

respective TS. Since the presence of a 2D track is mandatory for operation, which requires matching TSs in at least four of the axial SLs, the focus was put on the stereo TS. For this, each of the four networks were trained without using data from one specified stereo SL, for example, a network was trained without using any TSs from SL1.

With regard to the implementation on FPGAs, this means that the weight sets of five different networks have to be available and swapped at runtime. However, due to the latency of external memory, these must also be stored entirely in on-chip memory. Another consequence of this is that optimization of the weight set in order to save resources on the FPGA is severely limited. Since weights inhibiting different values can be loaded into the same processing unit during runtime, the implementation must be kept flexible enough to load multiple sets of weights. This is especially influencing the usage of compression methods. A general investigation of possible compression was carried out in Ref. [Reu18], in which simple pruning and bit width reduction approaches were prototyped. These provided only marginally better resource usage results that could not justify the ensuing loss of efficiency.

5.1.2.3. Configuration of the Multi Layer Perceptron

Several different algorithms and variants of neural networks were investigated for their suitability to be used as a z-Trigger. Reasonable performance is already achieved by using an MLP. It was configured with all of the neurons having a bias weight in addition to the weighting of the inputs. Meanwhile, the TANH function was chosen as the activation function throughout the network as its non-linear properties allowed to achieve good efficiency. The topology was configured with respect to the analysis in section 5.1.1 to consist of two layers, one HL and one Output Layer (OL). The HL is configured to consist of the previously mentioned 81 neurons, while the OL is set to two neurons. These two neurons are providing the final results of the NNT representing the estimated z-Vertex and cotangent of theta, which allows for more in detail analysis of the event and could be used by the GDL.

The definition of the topology is mainly driven by the hardware limitations of the UT3 platform, which theoretically could process the complete network within three clock cycles when using fully parallel processing, whereby approximately 90% utilization is reached for the DSPs. This implementation is also used in the first variants of the NNT, as described in section 5.4.2.1.

In addition to the topology, the bit widths for processing the neurons were investigated. In its full configuration, the NNT uses 14 bit to represent the input values of the neurons and 18 bit to represent the weights with 10 bit fractional. With these widths, analyses showed good results for estimating the z-Vertex.

5.1.3. Requirements for Trigger Operation

This section examines the concrete requirements for the NNT that are defined by the experiment and L1 trigger system. It will focus on connectivity, latency and throughput,

which are key for integration and operation.

Connectivity

For the NNT to be integrated into the CDCTRG, it has to provide two types of interfaces. On one hand, interfaces for the transmission of detector data and the establishing the data flow are necessary. For this, the CDCTRG is relying on the usage of optical transmission due to the data rates to be supported. For integration, the NNT has to provide the correct optical modules and support the requested data rates. Besides the interfaces for the transmission of the CDC data, additional interfaces used for configuration and monitoring are required.

Since the NNT is receiving its input data from multiple different data sources, its hosting platform must offer a variety of interfaces. The CDCTRG is mainly based on using GTH transceivers since these have the most capabilities of all the IO resources on the targeted FPGAs. Newer platforms, for example, based on the Ultrascale architecture, are meanwhile supporting more powerful transceivers like GTY and GTZ. These are especially important for an upgrade of the system such as it is scheduled with the introduction of the UT4 platform. While the input is requiring high data rate interfaces, data to be sent from the NNT has to fulfil lower data rate requirements due to the data concentration that an z-Vertex estimation is representing. For these interfaces the lower-performance GTX transceivers can be used within the CDCTRG.

The number of transceivers that are to be provided by the NNT can then be determined by analysing all data channels to be supported. The number of GTX transceivers that have to be provided consists of one input channel for the data of the ETF and one channel for the B2L. Meanwhile, data streams received from the 2DS and the stereo TSFs are possessing significantly higher data rates that require the usage of GTH transceivers. Data representing the axial TSFs are transmitted together with the 2DS and thus do not need additional consideration. The data for the four incoming stereo TSFs, on the other hand, are received via the ETF. Whereby the data is not changed there, but only forwarded and delayed. In total 64 GTH lanes must be supported by the NNT to cover the complete CDC. This is, however, not possible to be realized with any of the available FPGA platforms. To solve this, several FPGAs are planned to be used with a strategy partitioning the detector's space. The in-detail discussion of this topic can be found in section 5.2.1. The summarized calculations related to amount of needed GT lanes are additionally shown in the equations 5.2.

$$\begin{aligned} TotalGTX &= ETF + Belle2Link = 2GTX\ Transceiver \\ TotalGTHstereo &= TSFs \cdot LanesTSF = 4 \cdot 8 = 32Lanes \\ TotalGTH2D &= 2D \cdot Lanes2D = 4 \cdot 8 = 32Lanes \\ TotalGTH &= TotalGTHstereo + TotalGTH2D = 64Lanes \end{aligned} \tag{5.2}$$

Latency

The latency until trigger signals provided by the trigger system are ready for usage, is one of the critical requirements to be fulfilled. Under no circumstances may it lead to the CDCTRG exceeding the latency budget of the L1 trigger system. The NNT must hereby guarantee a deterministic latency within the budget. The budget available for the NNT is meanwhile derived from the accumulation of the latencies of the individual components of the CDCTRG. This is including the individual data transfers coupled with the internal latency of the data processing. The NNT must then generate its trigger signals at a point in time so that it can be sent just in time for the subsequent GDL and GRL to consider it. Both of these components have to perform their internal processing of trigger signals before the fixed deadline.

The NNT is highly dependent on the latencies of its different data sources. Processing can only commence when the required data is available. Across all used components of the CDCTRG, the 2DS has the highest latency to be considered, since it has to wait for the TSF. Only when a track is available, processing can start. The latency budget of the NNT is therefore given by the worst case latency achieved by the 2DS and the processing latency of GRL/GDL.

The separate latencies of the important components at the beginning of operation during phase 3 of the experiment are shown in the following equations 5.4 and depicted in figure 5.5 that was provided by the CDCTRG group [63]. These latencies are represented as the remaining time between the arrival time and deadline set by the GRL. As a result, the shown numbers are all negative, which means they arrive before the deadline. As they all have an inherent variance due to the measured transmission delays, the estimated worst case time is considered. It is hereby only estimated as it was determined empirically from experiments. Referencing the worst case latency of the 2DS and estimation for communication between NNT and GRL, a remaining time budget of around 350 ns is available for all of the internal processing to be carried out at the NNT.

$$Latency_{TRG} > Latency_{GDL} + Latency_{GRL} + Latency_{NT} \tag{5.3}$$

$$Latency_{NNT} = MAX(Latency_{TSF}, Latency_{2D}, Latency_{ETF}) + Latency_{Comm} \tag{5.4}$$

$$\begin{aligned} Latency_{sTSF} &= 1000ns \\ Latency_{2D} &= 750ns \\ Latency_{Comm} &= 400ns \\ MaxProcessingTime_{NNT} &= 350ns \end{aligned} \tag{5.5}$$

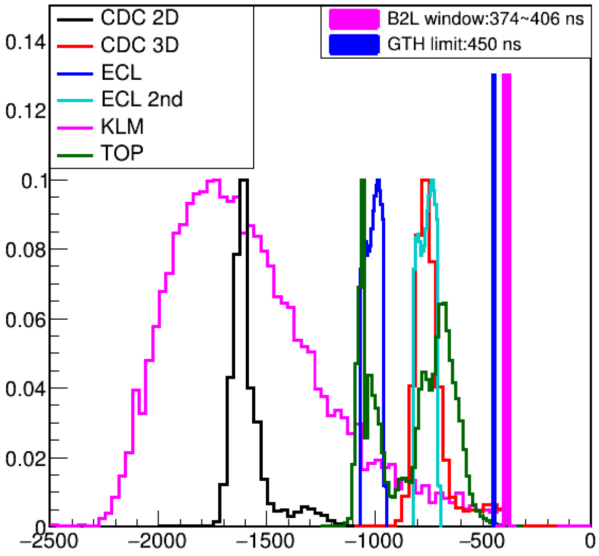


Figure 5.5.: Plotted histogram showing latencies of different components of the CDCTRG recorded at the GRL [63].

Throughput

In contrast to latency, the throughput that has to be achieved is not fixed to a certain number but rather has to be within a defined range. This range is defined as the number of 2D tracks processed at each data clock cycle. While optimal operation is achieved when all found 2D tracks are processed, the NNT can already be used when only one track is supported. The throughput is described as the number of processed 2D tracks per time unit. The maximum number of tracks is determined by the update rate of the implementation used for the 2D tracking. Since resources are limited for both the outgoing data rates and the internal processing, the 2DS is only sending a subset of the possible tracks. The selection of this subset is made on the basis of the estimated momentum for which the tracks with the highest momentum are preferred.

$$\begin{aligned} Throughput &= Tracks \cdot Frequency_{2D} \\ Throughput_{Max} &= 6\,Tracks \cdot 31.75\,MHz = 190.5\,millionTracks/second \\ Throughput_{Half} &= 4\,Tracks \cdot 31.75\,MHz = 127\,millionTracks/second \end{aligned} \tag{5.6}$$

The number of tracks meanwhile depends on the data rates supported by the hosting FPGA platform. The 2DS might be implemented on both the UT3 and UT4, which could achieve different rates. Considering the highest data rate for transmission across platforms, six tracks can be sent at each data clock cycle. Since the UT3 can only achieve half

of the maximum data rate, it is sending up to four tracks. Using these numbers as the update rates, the throughput can be calculated as described in equation 5.6. The importance of both presented throughputs depends on the operation status of the experiment. At first, only $Throughput_{Half}$ will be achieved within the CDCTRG. Later on, with upgraded electronics, $Throughput_{Max}$ may be achieved and necessary to cope with high luminosities.

Considering the NNT, an estimation for each track requires the calculation of the complete processing chain from preprocessing to the MLP. This must then be either instantiated multiple times to allow processing of all tracks in parallel or operated with a high frequency to allow pipelined time-multiplexing while reusing the same processing structures. This results in a trade-off between resource consumption and throughput or rather achievable suppression of background events. Maximum throughput provides the best suppression but requires more resources. Resources can be saved by reducing throughput, but the suppression capabilities of the trigger will worsen since not all tracks will have an estimation of the z-Vertex.

The minimum goal for the NNT is set to estimating at least the 2D track with the highest momentum in any given data clock cycle. The 2DS, however, does not generate a new track at every clock cycle during collisions, it is rather arriving in bursts of at most four active successive clock cycles for a given event. Each burst is usually followed up by a window of inactivity, which can be exploited to defer the processing of subsequent tracks to later clock cycles in case that the NNT's processing pipeline is occupied at the moment.

Summary

The NNT has a high overall demand for GTH/GTX ports in order to receive and send all the data. In addition to the dataflow, it requires interfaces to use service functionality such as DQM or SC. Accordingly, a selected board must provide the necessary number of connectors. At the same time, the NNT must meet hard real-time requirements for latency and throughput. It has only about 300 ns time to estimate the track parameters so that the entire CDCTRG remains within its time budget. At the same time, it must be able to process at least one track per data clock cycle and must not deviate from these requirements.

5.2. Realization and Implementation of the neural z-Vertex Trigger

The focus of this section is put on the implementation of the NNT. This is introduced by the presentation of the aspects regarding the integration into the CDCTRG and the selection of a suitable platform. Subsequently, the treatment of the different transmission protocols of the CDCTRG is presented.

5.2.1. Integration into the Trigger System

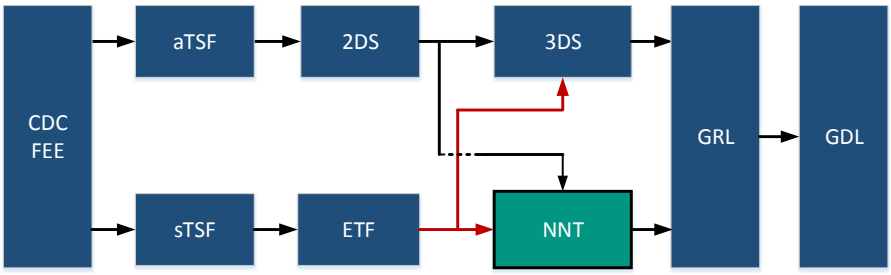


Figure 5.6.: System architecture for the L1 trigger system based on the CDC.

The NNT is performing the estimation of the z-Vertex for particle tracks based on the data of the CDC. For this, it has to be integrated into the CDCTRG to be able to receive the needed data and send the requested trigger signals. Receiving the raw data directly from the FEE of the detector is not feasible due to the limited number of IO ports provided by single FPGA platforms and the shared transmission infrastructure. Inputs are received from the three components TSF, 2DS, and ETF. Due to the importance of reliable z-Vertex estimation, the 3DS is generating trigger signals in parallel to the NNT. The system architecture of the CDCTRG, depicting all of its components, is shown in figure 5.6. The NNT hereby represents the third and last step in the generation of trigger signals.

One difference to all other components that has to be addressed at both the NNT and 3DS is that they are the only components receiving data from multiple data sources. These sources have their own latencies for processing data, which leads to several parallel data streams with different delays for data related to the same physics event. As an event is processed as a whole by the NNT, it is necessary to synchronize these separate data streams. Mechanisms must be introduced on the receiving side to compensate for the different latencies. To solve this issue, the so-called unsyncher was designed and developed [61]. The unsychner's task is to delay the data streams of the sTSFs for a given time interval. Additionally, all of the sTSFs are not directly connected. Their data streams are rather forwarded through the ETF. Such forwarding has two advantages, on the one hand, it reduces the number of required IO ports for the sTSF as it only has to send the data once instead to both NNT and 3DS. Contrary to the TSF, which is at the limit of the available GT lanes, the ETF still has free ports available to transmit data in parallel to both z-Vertex estimators. The other advantage resulting from this approach is that the data from the sTSF is delayed. The connection to ETF introduces additional delay for the transmission of the data. This is about the same delay for the data transmission to and from 2DS. This delay thus makes it easier to synchronize as the difference between sources is reduced. Additionally, data has to be buffered for shorter time intervals, keeping the footprint for necessary synchronization overhead low. Only the processing latency of the 2DS and jitter of the communication have to be additionally balanced.

The high demand for GTH lanes that are required to be supported by the reception of the entire CDC is higher than the resources provided by any available FPGA platform. This not only applies to the NNT but already to the 2DS, so the same solution that was found there will be adapted for the NNT. Instead of using the data from the entire detector, it is hereby partitioned into separate regions that are processed separately. Here, instead of multiplexing the transmission, which would introduce additional latency, multiple boards are installed that receive the data in parallel. The maximum coverage that can be realized within the IO resource budget is about half of the CDC for one FPGA platform. So theoretically, two boards should be sufficient to cover the entire detector.

However, since particles will not travel through the detector in a straight line but are rather bending, this partitioning is not employed. The reason is that a particle that is passing through both halves cannot be correctly processed, as its positional data is divided. The number of such particles is substantial enough to justify the usage of another partitioning. Instead of two halves, the readout is divided into four parallel data streams that are related to quadrants. These are not strictly disjointed but are rather overlapping. The entire space of one quadrant is overlapping with its two neighbouring quadrants. In the CDCTRG, these quadrants are sent in parallel to separate FPGA platforms. Due to the overlap, the same space of the CDC is always received by two separate boards. Although this is redundant, it has two important advantages. The first is a better approach to process particle tracks that are bending in space. The second advantage is that this redundant data can be used for validation of correct operation. Redundant data should appear in both of the overlapping regions across the physical borders of the transmission lines at the separate FPGAs. Additionally, this allows smoother recreation of TSF data at the borders of the quadrants, in which unusual activity is used as an indicator for problematic operation by the subsequent DQM.

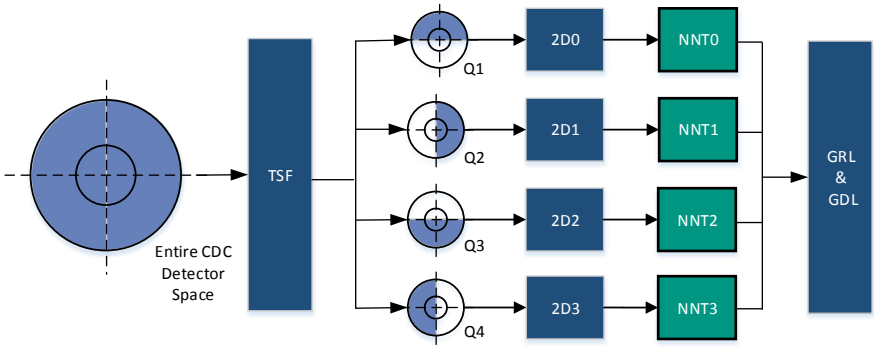


Figure 5.7.: Schematic representation of the partitioning of the CDC's space.

The graphical representation of the partitioning of the CDC is shown in figure 5.7. The representation shown there is an exact replication of the real setup, in which the edges of a quadrant's borders are bent. SLs that are located farther outside are covering larger

ranges. However, the representation is sufficient to explain the principle and region processed by each board. The details of the correct assignment on the granularity of single TS IDs can be found in the Appendix A.3.2.

Using this partitioning of the CDC's readout has a severe impact on the integration of the NNT as it significantly decreases the number of required GTH ports. A system's level view of one NNT board as a module with the GTH ports shown as interfaces is hereby depicted in figure 5.8.

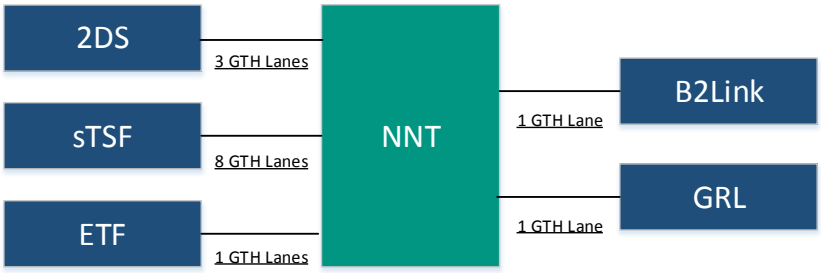


Figure 5.8.: Interfaces supplemented by the amount of required GTH lanes for one NNT board covering one of the CDC's quadrants.

5.2.2. Selection of a Hosting Hardware Platform

The NNT is not bound to a predefined hardware platform to be hosted on. Above all, the selected platform must meet the requirements of CDCTRG. In addition, however, the cost of the platform and needed time until operational readiness have to be considered, as each platform not sanctioned by the budget of the KEK, has to be funded through external means.

In order to be integrated into the CDCTRG, a suitable FPGA-based platform has to be found. Its FPGA has to be powerful enough to host the NNT such that its algorithms can be implemented within the defined latency and throughput requirements. In addition, the used FPGA has to provide enough GTs that support the required data rates. The hosting carrier board has then to route a sufficient amount of those ports to its IO interface for connection of the necessary optical links. Optimally, it also contains the necessary interfaces to be connected to the services outside of the essential data flow without introducing demand for significant additional implementations. These services include slow control, clock distribution, and a synchronized readout for DQM. These services are highly application-specific compared to the rather general optical transmission used for the data flow. They are often using additional physical interfaces such as VME or need to support an additional add-on board, as is the case for B2L. Additionally, they require dedicated custom IP cores that are implementing the corresponding protocols. These, can

be dedicated to a specific FPGA when primitives are used. When using platforms that are not supported, these IP cores have to be adapted to the hosted FPGA.

The number of available FPGA platforms is rather large. However, the special requirements for integration reduce the number of choices significantly. In the following, a selection of suitable solutions for a platform will be presented, and their suitability discussed. The selection is limited to five possible platforms, which all meet the basic requirements of the NNT. It consists of the UT3 and 4, the VC709 [134], and the Pulsar Board [10]. Additionally, the possibility of developing a custom platform just for the NNT is an option. The possible solutions are evaluated according to five main criteria. These criteria consist of the provided performance, IO capabilities, external support in form of the availability of IP cores, the availability to be ready for early operation, and the monetary cost for obtaining the platform.

The generation and architecture of an FPGA ultimately determine the achievable performance in terms of latency and throughput. The fact that modern FPGAs are typically using the smallest available technology node on its own already reduces the signal delays within the processing architectures. This applies to all relevant resources such as DSPs, BRAMs and LUTs. At the same time, they usually offer more resources due to the increased density of the circuits. Especially regarding both LUTs and BRAMs, FPGAs have much more resources available across successive generations. However, the increase in DSPs was stagnating until recently. Their availability is currently increasing due to the ascension of machine learning as one of the main targeted application domains. However, such platforms were not released in time to be used at the first stages of the experiment's operation.

In section 4.4, the implementation of an MLP on FPGAs was discussed. One of the main points made there is that DSPs are particularly suitable for the implementation of neurons in order to achieve high performance. As a result, their availability on a possible FPGA platform is of particular interest in the selection. Remaining resources such as LUTs are meanwhile of particular interest in the implementation of the preprocessing and communication infrastructure. A high amount of available LUTs is typically implicitly scaling with the routing resources, which are important to resolve timing violations more easily. Last but not least, BRAMs are needed in two areas. For one, they are used for storing the network's weights. A high amount of BRAM can allow more networks to be loaded during runtime. However, this is not only dependent on the memory resources, but routing as well to achieve timing closure. The other area in which BRAM is used is for the implementation of all functions that require significant processing resources when implemented in LUTs. An example of this is the activation function, which is usually not linear and at the same time might require good precision. The function can be stored in memory instead of being calculated online.

One important factor with regard to integration is the form factor of the hardware platform and its supply connections. In the end, it has to be integrated into the infrastructure available at the experiment's facility. A dedicated area called the E-Hut is designated to provide the space for online processing hardware including the trigger system. For this purpose, it is equipped with patch panels and crates that encompass the readout of the CDC among all other detectors. The available crates are based on communication through a VME backplane and support boards with a standardized form factor. In case a platform

that is not complying with this standard is used, an alternative solution outside of the crates has to be found. This can be a complex task since both space and cabling have to be solved outside of the crate instead of using the already provided resources.

A tabular comparison of the considered platforms with regard to the criteria defined for their suitability is shown in table 5.2. In the following, the differences and ultimately the chosen platform will be discussed in more detail.

Platform	Performance	IOs	Support	Availability	Cost
UT3	−−	++	++	++	++
UT4	+	++	++	−	++
VC709	−	−	−−	+	+
Pulsar	−	+	−−	−	−
Custom	++	++	−−	−−	−−

Table 5.2.: Comparison of available FPGA platforms for suitability to host the NNT.

The UT3 is a custom FPGA platform produced by an external contractor for the KEK. It was primarily designed to meet the IO port requirements for all sub-systems of the entire trigger system. At the start of the trigger system's development, no alternative platform with a sufficient number of IO was available. The only suitable FPGA is part of the Virtex-6 HXT family. This family is a special branch of the Virtex line that is providing the highest number of GTH ports. For these FPGAs, however, no hosting platform that actually routes a sufficient amount of interfaces for external usage was available on the market, thus a custom board had to be developed.

Since the board is primarily used at KEK, the development of modules implementing functionality necessary for integration is carried out across the entire collaboration of the experiment. As a result, it is already equipped with a library of IP cores that support the used FPGA. These primarily provide the functionality necessary for integration. An example is an IP core that implements the B2L protocol handling. Since many of these functions can also be used by the NNT, the presence of the library is a big advantage of the platform. Because the library is used within the collaboration, it is continuously updated and maintained. The disadvantage is somewhat the resulting dependency when using it. IP cores must be checked for their correct behaviour with each update. Since they are external IP cores, the possibilities for manual adjustment for specialization or updates are limited.

At the beginning of the development of the NNT, the UT3 was already available. It represents the only solution that is immediately available for both integration and testing within the detector's infrastructure. At the same time, it is bought and provided by the KEK, which makes it the cheapest solution out of all possibilities. In addition, there are several spare UT3 boards available that are part of the equipment available at KEK.

Although the UT3 is excelling at most of the relevant criteria, it has two significant disadvantages. At the current state of FPGA development, the Virtex-6 is rather old and significantly less powerful than the modern alternatives. Another significant problem is an board-level error affecting the GTH connections. At best the board is able to achieve

only half of the maximum data rate when compared to the specification. This is already impacting the CDCTRG at the stage of the TSF, which can only send a part of the maximum amount of active TSs within an SL of the detector. The causes of the problem are known, but there is no way to solve it post-production. A smaller disadvantage is that the board is scheduled for use at the experiment and thus has to be at the facility. This makes testing of updates more difficult as a remote connection and sometimes physical access are required.

Some of the disadvantages of the UT3 are addressed by its successor the UT4. It is based on a newer generation of FPGAs. As Xilinx made the decision to release two successive generations with only slight architectural changes, both the Ultrascale and Ultrascale+ models are pin-compatible with the design of the board. Generally, it is a significantly better choice as it provides both more processing resources and GTs. As it is based on a new board layout, it should solve the problems of data transfer, such that the transceiver links should able to be operated with the maximum specified data rates.

At the same time, the UT4 is going to be supported by the KEK just like its predecessor and will thus possess the same advantages regarding the usage of both the IP core library and integration services. The big disadvantage of the UT4 is its availability. At the point of creating this thesis, its final version was still under development. Using it for hosting the NNT is therefore only possible for later stages of the experiment.

Another possible platform is the VC709 that is directly available from Xilinx. It is fulfilling the basic requirements by providing a sufficient number of GTH ports. However, it is not complying with the desired VME form factor and does not provide a physical connection to the backplane of the crates used within the E-Hut. As a result, the typical way of implementing both SC and DQM has to be conceptualized and implemented. The final integration is additionally requiring an adapter board to make some of the GTH ports available for usage and proper location within the E-Hut.

The big advantage of this platform is the enormous number of available DSPs provided by the hosted Virtex-7 FPGA. These allow for fully parallel implementation of the NNT's MLP, which is in turn leading to the lowest possible latency and highest operating frequency. At the same time, the platform is capable of operation with its GT set to maximum data rates. In addition, the platform has a high level of availability, as a sufficient amount of boards are in stock and can be ordered within a short delivery time. Due to its wide availability and presence, it was chosen to serve as a host for a local demonstration setup, which is further described in section 5.4.3.

Designing and producing a custom platform is also a possibility for the NNT. The big advantage is that this platform can be completely tailored to the special requirements. All other possible variants are developed for more general purposes and as such are always including some form of either overhead or disadvantages. On this platform a new generation FPGA, Ultrascale+ could be used, which has significant advantages for latency and resources. The major drawback of this solution is, of course, the cost of development and time until it is operational.

In the end, a combination of UT3 and UT4 was chosen for hosting the NNT. The UT3 is used in the first phases of the experiment, while the transition to the more powerful UT4 is performed as soon as it is available and allowed by the maintenance schedule for

experimental operation. The reason for this choice is mainly driven by the advantages related to the availability, cost, and integration. Since the whole project is rather time-critical, as the experiment's schedule is determined by more important components and financial factors, short development time is paramount for success.

5.2.3. Interfacing to the Trigger System

The NNT receives its input data from three independent sub-systems. Each of them is using their own protocols and formats. As they are representing the data collected from the CDC, the protocol is depending on the physical behaviour of the detector. In summary the NNT has to implement the protocol, data format, and timing of the individual components. These three aspects are presented for the respective sources before the implementation of the handling of the interfaces is described.

5.2.3.1. Protocols of the Interfaces

In order to understand the interfaces, it is necessary to introduce two terms used in the CDCTRG. One of these is the data clock, the frequency with which data from the optical links is passed from the transceiver modules to the processing on the FPGA. This frequency is defined at 32.125 MHz throughout all the systems. The system clock, on the other hand, is the clock frequency with which the logic for processing is intended to be operated with. It is set to 125 MHz, being a multiple of the data clock. Most of the subsystems within the CDCTRG are internally operated with this frequency. Meanwhile, two key figures are used for the data rate. Firstly, the full data rate of 10.28 GBit/s. This data rate was originally intended for the CDCTRG, but cannot be stably maintained over time on any UT3 board. To work around this problem, half the data rate at 5.14 GBit/s is used for the first stages of operation, with the outlook that the upgraded hardware will fix this data rate problem. In the following the handling of the data for both possible data rates are shown. In table 5.3 the introduced key values are shown again.

Name	Value
Data Clock	32.125 MHz
System Clock	125 MHz
Full Rate	10.28 GBit/s
Half Rate	5.14 GBit/s

Table 5.3.: Terminology for data transmission within the CDCTRG.

Stereo Track Segment Finder Connection to the NNT

Considering communication with the TSF, only the stereo Track Segment Finder (sTSF)s has are directly connected to the NNT with axial information being sent as part of the data sent by the 2DS. The data is hereby received at the processing side with the data clock. The maximum number of TS that has to be processed per clock cycle is hereby depending on the used data rate and version of the TSF. Up to 20 TS can be received per clock cycle when using the full rate. Meanwhile, when using the half rate there are to possibilities, that is either 10 or 15 TS. A configuration with 10 TS is intended to be the base configuration for the first stages of operation. In case that the occupation of the CDC is going to be high, an additional configuration supporting up to 15 TS was implemented retrospectively. This configuration is utilizing the complete half rate whereby the 10 TS variant was using only part of the available bandwidth. To summarize, both variants are shown in table 5.4.

The position of an individual TS within the CDC is uniquely identified by a TS Identification Number (ID). Geometrically, this TS ID describes the position of the primary priority wire around which a segment can be constructed. Besides the TS ID, the estimated priority timing is transmitted as well as the type of active priority wire and left/right information.

The NNT uses all this information for its preprocessing to generate the appropriate inputs for the MLP. It is also used to decide which of the found stereo segments matches the best to the estimated 2D-Track. When receiving the data, however, additional properties of TSF and CDCTRG have to be considered that are due to the nature of the used detector, as it is based around the drift of particles.

Configuration	TSs per clock cycle
Half rate standard	10
Half rate maximum	15
Full rate	20

Table 5.4.: Possible data transfer configurations for the sTSF.

The most difficult aspect of the communication with the TSF is that the data of TS is not synchronized across all SLs. Segments that belong to the same event can be detected and sent by the TSF in different clock cycles. This is mostly due to the characteristics of the CDC as wires are not immediately active after a particle passes through. They rather experience drift in which it takes time depending on the position of pass-through until the generated charge is detected in the detector. Besides that behaviour the actual structure of a TS is important as it is only considered to be active when a sufficient number of wires are active within the segment. Combined with the drift, a TS is changing it is characteristics over time, possibly changing from inactive to active rather late compared to the other matching TS or even leading to more precise resolution of the particles pass-through, by updates of priority times and the associated priority wire.

Since one TSF module is only observing the wires within one of the SLs, it is not possible to synchronize TS across the entire detector at this level. This is only worsened by the

differing transmission delays across different trigger hardware platforms [99]. The behaviour observed at the interface of the NNT is shown as a timing diagram in figure 5.9. In this, several TS are turning active that are all part of the same event. However, they arrive at different points in time, so that they have to buffered and matched.

As this is not problematic enough, the NNT has to deal with the presence of background events, which will intermix itself with collision events making it more difficult to detect the correct TS. This is somewhat mitigated by suppression strategies, as described in section 7.2.2, however, they do not cover all possibilities.

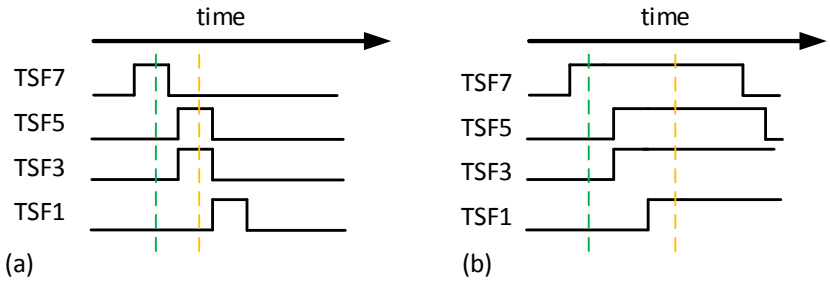


Figure 5.9.: Active TSs arriving at different points in time at the NNT with (b) and without a persistor (a).

This behaviour is problematic for the NNT since it only processes the received TS together with a matching track from the event. However, it is not possible to define an exact clock cycle for which the appropriate TS are arriving. The NNT, therefore, needs a receiving module for each TSF that temporarily stores incoming data for a certain amount of clock cycles, so that they are processed together with the 2D-Tracks that are arriving later. This module is called persistor because it stores incoming TS over a defined time interval. The implementation of the persistor module for the NNT is discussed in section 5.2.3.2.

Reception of 2D Track Finder Data

The data sent by 2DS is centred around the estimated track parameters phi and omega. These values are representing the cell of the Hough space representing the parameters matching the observed segments the most. While omega can be negative, phi is represented as an unsigned value. Each 2DS board is capable of finding up to six track estimates, however, due to the mentioned problems with the data rates it is limited to just four at the beginning of operation. Each track estimate that is sent is accompanied by status information and a selection of TS that are matching with the track. The same information as provided by the TSF is sent by the 2DS, thus they are indistinguishable in the formatting and processing at the NNT. However as they are already matched to the track, there is no need for implementation of a persistor for these TS. As with the TSF, the 2DS is also impacted by the drift times of the CDC. As a result, the TSs belonging to

the same event are updated at different clock cycles, in which the 2DS in turn also updates its estimation of the track. Additional status bits are sent with each track to indicate this. These status bits encode the information indicating whether a sent track is new or an updated old one. This information can, for example, be used by the GRL/GDL to detect trigger signals generated by the same track, which can be used to reduce trigger rates. Sub-systems using this data can also use it to determine whether to process the track again or perform additional optimization using values that were calculated the first time this track appeared.

5.2.3.2. Realization of the Persistor for the Track Segment Finder

The persistor is an architectural module for handling the time-variant reception of TSF data. This section describes the requirements, architecture, parametrization, and implementation of this module for the NNT.

Requirements

The basic task of the persistor is to temporarily store a received TS over a defined time interval. The moment a 2D-Track arrives, all TSs that were received in this pre-defined interval are subsequently passed on to the actual data processing of the NNT. This module has to be carefully considered for the implementation on the FPGA. When receiving the data, in the worst case not only the maximum number of incoming TS per clock cycles has to be stored, but the total amount of TSs that were accumulated over a period of time T. This means that a larger memory must be provided depending on the covered time interval. The depth is hereby given by the formula 5.7. Here T is defined in clock cycles and is determined by the readout of the CDC.

$$MemoryDepthP = T_{Buffer} \cdot TS/ClockCycle \tag{5.7}$$

An analysis of the detector's behaviour was performed to determine a suitable time window T that allows buffering a sufficient amount of TS. Here the arrival times of the TS across all SLs were analysed. These were then matched to the 2D-Track that is representing the physics event. Using the ensuing distribution of arrival times relative to the 2D-Track, the buffer size can be determined. The used data was meanwhile acquired through both experiment and cosmic ray operation. As the 2DS needs to perform the same task for axial TS, the analysis was conducted there as well which can be reused for the S3D discussed later on within this thesis.

The earliest point in time at which TS that are related to a 2D-track are arriving is at -28 clock cycles, before the reception of the track parameters from 2DS. Segments that arrived even earlier are hereby mostly due to background events and other effects not related to the experiment. The analysis showed that even an arrival time of -28 clock cycles is a rather rare occurrence for targeted events. The two most reasonable time windows have an earliest time of arrival of -24 and -16 data clock cycles, with which most of the related hits are covered.

The incentive behind keeping the persistor's time interval as small as possible has two reasons. For one it reduces the mixing of TS not related to the event with those that are related since they are only occurring around the event. Additionally storing for longer time intervals has a severe impact on the implementation on FPGAs, as more storage and logic resources have to be allocated to fulfil this task.

The amount of TS to be stored is additionally depending on the data rate of the used TSF. The required memory depths relative to the configuration are shown in table 5.5 with the assumption that the maximum amount of segments is received at every data clock cycle. However, observations of the TSF's behaviour have shown that only around 24 to 32 TSs are actually received, as most event data is occurring in time-local bundles of many hits within one clock cycle. Outside of these bundles single TS are received only sporadically. Experimental investigations showed that memory depths of 16 to 32 TSs are sufficient when half the data rate is used.

TSF Configuration	Memory Depth
Half rate 10 TS	320 TS
Half rate 15 TS	480 TS
Full rate	640 TS

Table 5.5.: Required depths of persistor memory for different possible output configurations of the TSF.

Realization

Each sTSF is sending its own independent data stream that is representing the different SLs. As those are completely independent of each other, each can be processed by their own instance of the persistor which is allowing fully parallel processing. As the NNT only receives the stereo data, a maximum of four persistor instances is hereby required. In comparison, the S3D discussed in chapter 6.1 additionally requires the five aTSF. As the S3D is part of this thesis, the overall architecture of the persistor is therefore designed to be flexible and parametrizable in order to be reused again

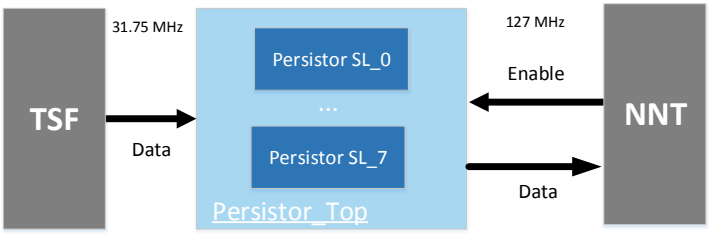


Figure 5.10.: Architecture persistor at the NNT.

Since the TSF is transmitting data with the data clock, the receiving side of the persistor is implemented to be clocked with $31.75MHz$. Valid input can hereby be received in each clock cycle. As a result, the persistor can spend at maximum one data clock cycle for receiving and buffering in order to keep up with the pipeline of the trigger system.

Readout of TSs that are buffered is controlled by the reception of tracks from 2DS. Buffered TSs are only processed when such a track arrives. In this case, a read enable signal is implemented at the readout side. As the readout is basically decoupled from the communication with the CDCTRG and rather used for internal processing on the FPGA, it is clocked with the internal clock used for processing across the NNT. Its clock frequency is much higher compared to the data clock, but mostly dependent on the implementation of the algorithms. The output of the presistor is then the combination of all TS observed within a specified time interval.

In order to support the required different memory depths and data rates, the persistor is designed to be parametrizable. The parameters consist of the number of TS received at each clock cycle, the time window of buffering and the total memory depth representing the total amount of stored TS. The parametrization is defined in a VHDL package, that is included throughout the processing architecture. This way, the module can be easily configured by external tools such as the generation framework discussed in section 5.3.1.

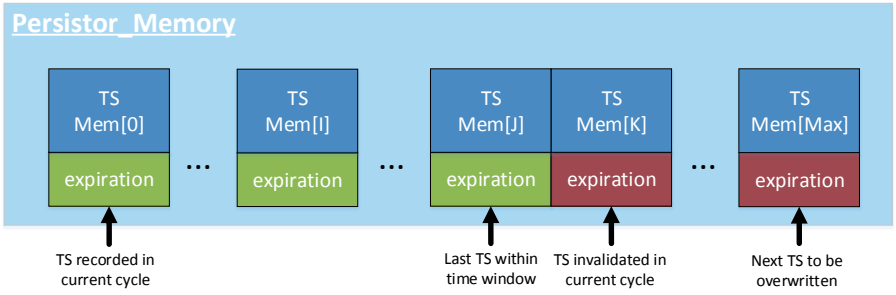


Figure 5.11.: Structure of the persistor's buffering of TSs together with the notation of internal pointers used for memory management. Valid entries are indicated by green expiration, while invalid entries are marked red. The status of the most important entries is additionally described with text below arrows pointing to them.

Functionally the persistor examines a defined number of TS from the respective input stream at each data clock cycle to determine whether they are active or not. Depending on the provided priority information of each TS, the persistor will tag single entries as valid and buffer them. Each valid TS is stored together with a counter that represents the point in time at which it was recorded. This time counter is used to implement the persistor's time window mechanism. As soon as the value of the counter is exceeding a pre-defined limit of temporal validity it is marked as expired and invalidated. At the point in time in which a 2D track arrives, expired entries will not be used for further

processing. The buffering itself is hereby realised as a ring memory, which is managed by a set of pointers that are indicating the position of the most important entries such as the start and end locations. Additionally, there is no protection against valid TS being overwritten. This is due to the fact that newer TS are more likely to be related to the event than older ones. This is supplemented by the selection of suitable TS discussed in section 5.2.4.4. The selection prioritizes segments with smaller drift times, that typically arriver later than others. The memory structure and functionality are meanwhile both illustrated in figure 5.11.

Evaluation

The most important question in the evaluation of the persistor is which configurations are actually feasible for implementation. In the best case with regards to the efficiency of the NNT a rather large time window and storage is the most beneficial since it is not known at which point in time a segment is exactly occurring. In the following, this interval is referred to as DW for the data window. In addition to the supported time interval, the total number of stored TS is also of interest.

Since the available memory on an FPGA is constrained, only a limited number of TS can actually be stored within a time interval. The maximum number of TS to be stored by a persistor at a given point in time is referred to as ST. The last configuration parameter that is examined is the number of TS supported or received per clock cycle. This is generally variable due to different configurations of the CDCTRG, however, the NNT can decide to use only a part of the maximum number of TS itself to reduce the resource demand. In the following, different configurations are examined.

In order to investigate the influence of the different parameters on the resources and the achievable clock frequency, they were explored by implementing the resulting persistor configuration. First, the module is evaluated serving just one SL which represents sequential processing with regard to the achievable clock frequency and resource consumption. The results are shown in table 5.6. They are the result of the Synplify tools for the synthesis and the tools provided by Xilinx for place and route. The FPGA used as a reference is the one mounted on the smaller UT3, which is the XC6VHX380T.

DW/ST/NT	Frequency in MHz	LUTs total	LUTs in %
16/16/10	242	650	<1
32/24/10	191	8 319	2
32/32/10	188	5 093	1
32/32/15	148	14 800	3
32/32/20	138	18 386	5

Table 5.6.: Figures of merit for frequency and resources of different persistor configurations that can be used during operation for one sTSF.

From the results, it can be seen that the persistor's resource consumption increases significantly with the increase of configuration parameters. Especially when considering the amount of TS to be received in a certain clock cycle, with the maximum configuration being 20 TS, a single persistor is already demanding a lot of resources with around 5%. At

the same time, the achievable frequency of 138 MHz is quite low but is still fine for operation with a system clock frequency at 127 MHz on the readout side. The final selection of the parameters is depending on the remaining system's implementation and will thus be determined for each of the used setups separately. The used setups and configurations are discussed in section 5.4.2.

The NNT requires the implementation of several persistors for the reception of all TS. The lowest overall latency can be meanwhile achieved by instantiating four persistors for all of the sTSFs. Exploration of the implementation parameters with parallel reception of all sTSFs were conducted for this purpose. Table 5.7 shows the results of these tests. The parameter SL is representing then how many persistors were implemented in parallel. The same tools and FPGA were used as before.

Looking at the achieved values the big influence of the number of TS per clock cycle NT is obvious here. In the maximum configuration, the persistors of the NNT are already demanding up to 27% of the available resources on the FPGA. In anticipation of the final setup, the maximum configuration will not feasible for implementation together with the core algorithms and is thus unsuitable for operation, since high resource utilization is leading to routing bottlenecks and preventing timing closure. Meanwhile, a configuration with 10 supported TS is much more resource-efficient with only around 1% resources used, even reaching almost 200 MHz for some configurations. At the same time, a setting of DW at 32 is probably too demanding, thus 24 seems to be a choice more likely to be feasible.

The results are especially interesting considering the transition to the S3D. Due to its support of all nine SLs the resources demand is going to be extremely high. An alternative approach could be advantageous here, that is tailored to the algorithms used there. Such an approach is presented in section 6.3.2.1. For the NNT however there is no alternative solution without redesigning the CDCTRG, a concept and investigation for this was conducted in section 6.4.

SL/DW/ST/NT	Frequency in MHz	LUTs total	LUTs in %
4/16/16/10	242	3 837	1
4/32/24/15	214	3 571	2
4/32/24/10	191	5 093	1
4/32/32/20	130	97 260	27

Table 5.7.: Figures of merit for using multiple parallel persistors for all of the sTSF.

5.2.4. Architecture of the Preprocessing

After the reception of the separate input streams, the raw data from the sub-components of the CDCTRG are transformed into a more suitable representation to be used by the neural network. The input transformation consists of several separate processing steps, some of which can be performed in parallel and pipelined. The architecture showing the data dependencies between the individual steps is shown in figure 5.12.

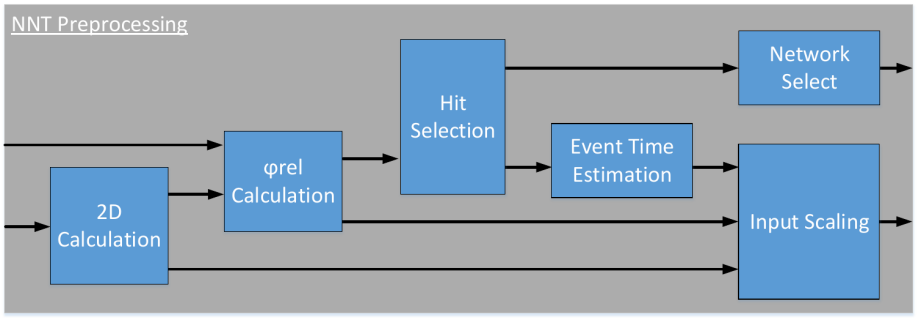


Figure 5.12.: Overall internal architecture of the preprocessing.

The entire processing pipeline is controlled by the arrival of 2D-Tracks. As soon as a track arrives, data processing of the NNT is started and the pipeline is filled. This starts with loading the TS as described before from the persistors. At the same time, the parameters of the incoming track are loaded and transferred to the preprocessing.

5.2.4.1. Preprocessing of 2D Track Parameters

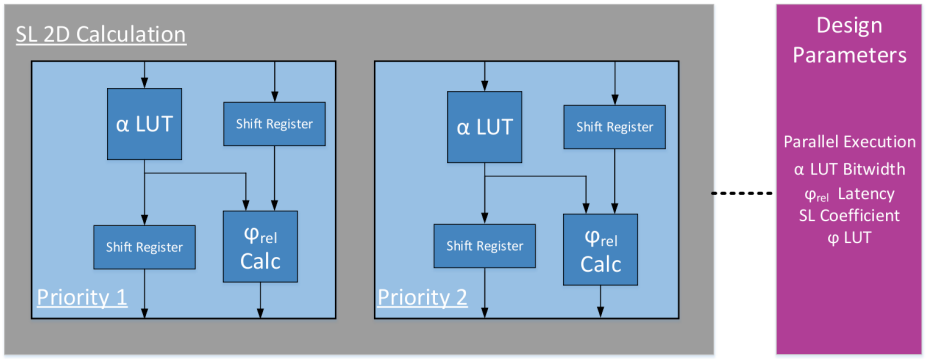


Figure 5.13.: Architecture of the preprocessing of 2D tracks with parallel processing of the SL's priority position.

The preprocessing of 2D-Tracks has two responsibilities in the overall architecture. Firstly it calculates the angle alpha of the track for each SL. This value is used as a direct input to the neural network. The second responsibility is the calculation of intermediate values that are used for determining the other two remaining inputs of the network, either directly or indirectly as it is also important for the selection of the most suitable TS for the

estimated track. This intermediate value is the reference angle for phi, which is used later on to determine the distance to a single TS referenced as phi_rel.

While alpha can be calculated independently from the others, the value is needed to calculate phi_rel. This means that the two output values of this module cannot be calculated in parallel and must be performed sequentially. For this reason, the entire architecture was divided into two parts reflecting the calculation of alpha and phi_rel. An illustration of the architecture is shown in figure 5.13. In the following, the implementation is discussed, whereby the realization of alpha is presented first, reflecting the internal data flow

Implementation of the alpha Calculation

The input value alpha is calculated directly from the 2D-Track parameter omega. The omega value received from 2DS hereby represents the coordinate in the respective 2D-Hough Map. To be used for internal processing it has to be converted from the Hough cell coordinate. An arithmetic conversion on the FPGA is quite complex since it features divisions as shown in equation 5.8 [100]. After that conversion, the value has to be converted to alpha according to equation 5.9 [91]. The best way to avoid this calculation is to implement the mapping in a look up table. Here all the values are calculated in advance and stored in memory. The calculation and instantiation of the respective modules are then part of the semi-automated framework presented in section 5.3.1. Overall the input parameter omega can assume 67 different values using signed representation. The look up table based implementation is hereby supporting the entire input range to achieve the best possible solution. Lower coverage can be used to reduce resource consumption but is only investigated as a fallback solution in case implementation is not feasible otherwise.

$$pt = 0.3 * 34/\omega_{input} \tag{5.8}$$

$$\alpha = \arcsin((r^{SL,pr} * pt)/2) \tag{5.9}$$

Since alpha must be calculated separately for each SL and is also depending on the priority position of the active TS, the look up table must be calculated and created 9*2 times in total. However, the calculation for the different SLs is completely independent of each other, such that they all can be processed in parallel. Since a TS belonging to a track can only inhibit one of the types of the priority position, only one of the alpha values calculated for each SL is actually is required and used for further calculation.

The used value of alpha thus depends on the selected TS as the priority position needs to be considered. The selection is meanwhile calculated in the hit selection, which is performed in a later step within the processing pipeline. Its result is then only available after preprocessing of the 2D track. To compensate for this difference in time, alpha is calculated in advance independent of the received TS for both types of priority, primary and secondary position, in parallel. The value to be used for the neural network is then selected later on based on the selected TS, while the other value is discarded. The main parameter of this module is the precision in bit width which is used for the representation of the value.

The module was synthesized with the Synplify tools for three FPGAs, to evaluate its key characteristics. These FPGAs are the possible options available for the UT3, UT4 and the UT4+. The synthesis results for the module are shown in table 5.8. Additionally to the different possible FPGAs, another test with 24 bit representation of the value α was evaluated. The minimum number of bits for which no major deviations were determined at the estimation of the z-Vertex is at 13 bits. However, this is a truncated representation as 24 bits are needed to recreate the arithmetic solution entirely. The extended representation is primarily investigated in case higher precision is needed for operation in which the smaller a bit width significantly impacts the estimation's efficiency. As this might only occur in later stages of the experiments, it is reasonable to explore this configuration beforehand to have the numbers readily available. The module is characterised in terms of slices and BRAM. The BRAM is hereby used for storing the results of the ω conversion. This is typically the more appropriate solution when compared to Slice-based memory in the presence of relatively high bit- and address widths. Overall the implementation requires one single BRAM for each instance while achieving fairly high clock frequencies of up to $300MHz$. The demand for BRAM is doubled when a 24 bit representation is used, while the frequency is slightly reduced. Both implementations yield reasonable characteristics in order to be used within the final system, the same holds true for the UT4 versions. The overall latency for processing is hereby at one clock cycle throughout all versions.

Category	UT3 13 Bit	UT3 24 Bit	UT4 13 Bit	UT4+ 13 Bit
Slice LUTs total	86	152	83	141
BRAM	1	2	1	2
Frequency in MHz	304	289	331	318
Latency in Clock Cycles	1	1	1	1

Table 5.8.: Synthesis results for the calculation of crossing angle alpha.

Implementation of the phi_rel Calculation

Similar to the processing of alpha it is necessary to convert the track parameter phi that is sent by the 2DS. It is as well sent encoded as the coordinate within the hough map. This approach of transmitting coordinates might seem to be unnecessary overhead since conversion approaches have to be implemented at every receiver. However, it is significantly reducing the required bandwidth, which is, in turn, freeing up bandwidth for additional track estimations. Considering that achieving a high bandwidth is a problematic task since the optical transmission does not always operate at its maximum capacity, this approach is much preferred over sending the complete value.

Again repeating the conclusions from the implementation of the conversion of alpha, it is unwise to implement the conversion algorithmically due to its complexity [100]. As a result, it is similarly implemented in a look up table. This table has 83 entries, which represents the estimated phi value for a track using 13 bit resolution as a fixed point value. Since this step of the processing can be done independently of alpha and in order to keep the overall latency as small as possible, it is scheduled into the first clock cycles of the entire processing and performed in parallel to the calculation of alpha. In the next clock

cycle both values, alpha and the translated phi, are available and used. The operation in the second stage of processing is described in equation 5.10 [91]. With regards to the schedule, at first, the subtraction is performed while the coefficient $N_{Wires}^{SL}/2\text{pi}$ is loaded. This coefficient depends on the number of wires in a respective SL and thus varies across instances of this module dedicated to service one SL. The result is a pre-defined TS ID that represents the best selection in terms of matching the 2D-Track.

$$ID_{ref}^{SL,pr} = (phi - alpha^{SL,pr}) * (N_{Wires}^{SL}/2 \cdot pi) \tag{5.10}$$

The synthesis results for this module are shown in table 5.9 and were created similarly to the approach taken for alpha. Two separate bit widths were considered here, that is 13 and 24. These bit widths were chosen since they represent the best options in terms of the resource-accuracy trade-off and the maximum bit width. In all cases, a frequency of over $200MHz$ can be achieved. To implement the conversion of phi, one BRAM is required for the look up table. The multiplication can be meanwhile implemented by either using a DSP or slices. The latter option is especially of interest when looking forward towards the later processing for the neural network that is mainly relying on the usage of DSPs. The problem with using slices is typically the reduction of the maximum clock frequency. However, results show that a clock frequency of over $200MHz$ can still be achieved, while the resources remain relatively low. This option is considered in the configurations used for operation.

Category	UT3	UT3 SRAM	UT3 24 Bit	UT4
Slice LUTs total	31	425	77	31
BRAM	1	1	1	1
DSP	1	0	1	1
Frequency in MHz	284	223	248	313
Latency in Clock Cycles	3	3	3	3

Table 5.9.: Synthesis results for calculating the reference IDs that are matching the received track.

Evaluation of the Complete 2D Processing

The synthesis results for the entire transformation of the 2D-Track parameters are shown in table 5.10. Here the results are encompassing the implementation for all SLs and priorities. These are additionally representing fully parallel operation and thus the best characteristics in terms of latency. In this configuration, the maximum latency is at three clock cycles. It is hereby not the sum of both sub-modules as their processing is interleaved to save one clock cycle. This latency is achieved with a possible clock frequency of over 200 MHz. Slightly lower frequencies are meanwhile achieved when using an implementation based on slices.

Category	UT3	UT3 slices	UT3 24 Bit	UT4
Slice LUTs total	681	3 832	1 147	631
BRAM	27	27	36	27
DSP	18	0	18	18
Frequency in MHz	268	219	220	295
Latency in Clock Cycles	3	3	3	3

Table 5.10.: Synthesis results for the entire preprocessing of 2D track parameters.

5.2.4.2. Calculation of the phi_rel Input Variable

As mentioned at the beginning, the value $ID_{ref}^{SL,pr}$ that is one of the results generated by the preprocessing of the 2D-Track, is only an intermediate value used to calculate phi_rel. For all TS, this input determines the deviation of the found 2D-Track relative to the position of the TS within the CDC. At this point, there is a separation of the processing into axial and stereo TS. As the former are already matched with the track and as a result are unique throughout the entire preprocessing, as such only one TS has to be processed here. Stereo TSs on the hand are different since they are not matched with the track. At this stage, it is unknown whether they are related at all since the matching is performed later on at the hit selection. As a result, all present TSs have to be processed within this stage of the preprocessing.

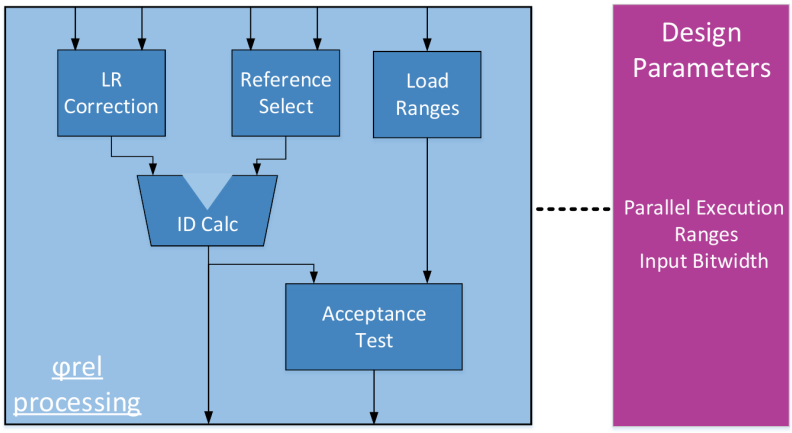


Figure 5.14.: Architecture of the module for the calculation of phi_rel.

The result of this calculation is the phi_rel value as well as a flag indicating whether the difference between received TS and the reference is within a pre-defined acceptance range defined by the respective SL. These acceptance ranges depend on the trained net and have

to be updated at the design time continuously during the lifetime of the NNT. Updating these ranges across the entire processing structures of the NNT is the responsibility of the semi-automated framework, which is generating the configuration parameters as part of included packages. In case that a TS is out of range, it is invalidated and not used further in the processing. The consequences are different depending on the orientation of the invalidated TS. In the case of an axial, this means that the whole SL is not processed further. With regards to stereo orientation, this has less severe implications, as invalidated TS will just be removed from the pool of possible choices for the hit selection.

The calculation of phi_rel additionally depends on the hit position of the priority wire and its supplementing Left Right information (LR) bits. Depending on whether it is left or right, an offset representing this deviation is added or to the difference between $ID_{ref}^{SL,pr}$ and the TS ID. This offset is stored in registers and loaded depending on the current LR information. The algorithmic operations are additionally described in equation 5.11 [91].

$$\phi_{rel} = (TSID + 0.5 \cdot LR^{-}ID_{ref}^{SL,pr}) \% N_{wires}^{SL} \tag{5.11}$$

Processing of all SLs is independent of each other at this stage and can thus be implemented for fully parallel execution. The degree of parallelism is one of the possible parameters of this module together with different bit widths for the reference ID and acceptance ranges. The architecture together with its parameters is illustrated in figure 5.14.

The synthesis results for the module are shown in table 5.11. With full parallel processing, the overall latency is at one clock cycle at an achievable frequency of up to $200MHz$. This is hardly reduced by increasing the bit width to 32 bits instead of 20. At the same time, resource consumption is very low with even fully parallel execution.

Category	UT3 20 Bit	UT3 32 Bit	UT4 20 Bit
Slice LUTs total	10 246	23 630	9 504
Slice Register	181	348	179
Frequency in MHz	254	244	271
Latency in Clock Cycles	1	1	1

Table 5.11.: Synthesis results for the calculation of phi_rel.

5.2.4.3. Estimation of the Event Time

One of the inputs for the MLP is the estimated drift time of each TS. This can be calculated by analysing the individual drift times of the TSF. Within the CDCTRG this task is the responsibility of the ETF. However, that component is still under development and thus cannot be used in the early stages of the NNT's operation. In order to still have an operational version of the NNT that can be evaluated even before the ETF is available, an alternative estimator was developed which is the focus of this section.

The alternative approach is based around an event time estimator module that is implemented within the NNT. Functionally it analyses all remaining TS after the hit selection and chooses the smallest priority timing that is then assumed to be the event time. More detailed analysis and comparison of the viability of such an estimated event time were investigated, which showed that the approach is able to achieve reasonably good precision [57].

Architecturally, the module receives the TSs that were selected by the hit selector as the input data stream to be processed. The amount of TS arriving at each clock cycle is hereby always limited to a maximum of nine TS. As a result, the estimator does not have to be designed to be flexible, as this number is representing the present SLs that will not change over time. The logical implementation is then based around a tree-structure comparison of the separate priority timings. Here, invalid TS are annotated with a timing value of -1 so that they are not taken into account. Otherwise, the default value for invalid TS would be set to zero, which would be incorrectly taken into consideration and even be selected as the smallest and thus the event time.

The synthesis results for the event time estimator are shown in table 5.12. Since the resources are very small, below 1% of the available SRAM-LUTs, they are given as absolute values. The module is hereby operating within a latency of one clock cycle at a frequency of nearly 250 MHz for both versions of the UT.

Category	UT3	UT4
Slice LUTs total	337	337
Slice Register	90	90
Frequency in MHz	249	281
Latency in clock cycles	1	1

Table 5.12.: Synthesis results of the event time estimation.

A special configuration of the event time estimation was developed later on in the experiment, after studying the drift times recorded from collisions. Many of the observed events included selected TS that had priority timings much smaller than most of the other TS. While this is not a problem in the original concept of the NNT that is using the ETF, it leads to high drift times when the smallest timing is assumed to be the event time. To address this issue, there are two options for using the drift time at the MLP in case they are exceeding the maximum allowed value. These options are either discarding the respective TS or setting its value to pre-defined bounds that represent the maximum/minimum allowed values. Further analysis showed that a significant fraction of these TS with high deviation in their priority timings are having stereo orientation. As a result, the issue can be additionally addressed by only using the axial TS for the estimation of the event time. While there is still a chance that these posses the same problem, it is occurring fat less often compared to stereo TS. Additionally, this issue can be addressed at the 2DS with an update of the firmware. While the presented solutions are mitigating the effect of such priority timings, they are not completely solving the issue. The best solution in terms of the impact on the accuracy of the estimation is the extension of the priority timing bit width by incorporating the clock counters of the respective TSF. This is directly address-

ing the source of the problem, which is an overflow in the 9 bit encoded priority time, which can be solved by extension to 13 bit as is done at the 2DS [62].

5.2.4.4. Track Segment Selection

At any given point in time, several TS can be active simultaneously within the same SL. While only segments with axial orientation are unique and already matched to the associated 2D-Track, one suitable TS out of the pool of available TS must be selected for each layer in case of stereo orientation. The selection of this TS is algorithmically defined by a set of rules [91]. With regard to the implementation on the FPGA, the implementation of these rules is the responsibility of the hit selection module, which is the focal point of this section.

Firstly the interfaces of the module are defined. The input data are all of the TSs that have been read from the persistor when a 2D-Track arrived. Since this number is variable, the selection must be capable of processing a variable number of TS. The data flow between the persistor and the following components is meanwhile controlled by synchronous enable and valid signals. The output of the module is then the complete set of the selected TS to be used for the neural network to estimate the z-Vertex.

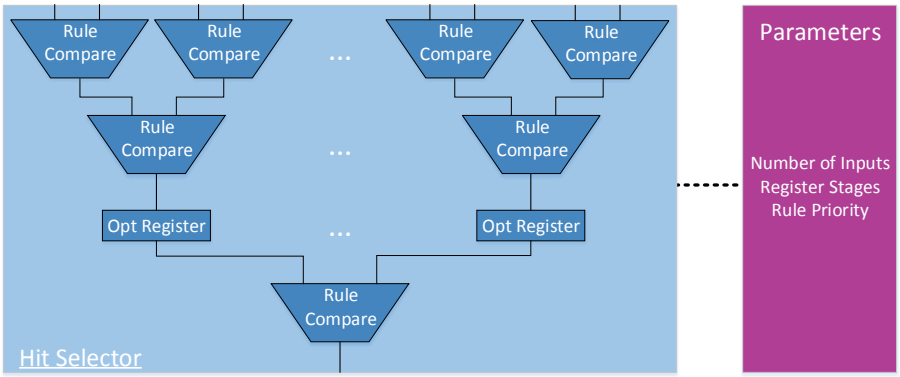


Figure 5.15.: Architecture of the track segment selection module.

For each SL, the module must select exactly one TS, whereby there are no dependencies between the individual SLs. This means that the selection can be carried out in parallel for all SL. In addition, the rule-based logic for selecting the TS is independent of the layer thus the module can be reused, without the need for additional parameters. The overall architecture is illustrated in figure 5.15.

The synthesis results of the module are shown for both relevant platforms UT3, in table 5.13, and UT4 in table 5.14. Three configurations of supported numbers of TS were hereby examined, namely for 8, 16 and 32. In addition, the resource consumption for all

fully parallel solutions and one sequential solution are shown, whereby the sequential solution requires four clock cycles to process all SL. The hit selector processing a pool of TS that is bound to 8 is thereby able to achieve a clock frequency close to 200 MHz. However, typically more than eight inputs can be expected during operation with collisions, as such higher numbers were explored. The 8 TS version is thus primarily of interest for operation with a smaller amount of detector data, for example during cosmic ray tests. The 16 input option is meanwhile already only operational with much lower clock frequencies, though they are still sufficient for operation with the $127MHz$ system clock. Achieving this frequency becomes more critical when supporting up to 32 inputs. In this option, not even the system clock can be reached as the maximum frequency is estimated to be at 120 MHz. When using the UT3, the only solution for this is to introduce another clock cycle into the processing pipeline in order to increase the achievable frequency, although this will, in turn, increase the overall latency. With such a setting, it is possible to achieve a frequency of over 200 MHz for 32 TS while the even more demanding 64 TS can be still operated with system clock. However, the resource utilization of 64 TS configuration is very high with nearly 12% of the FPGA's SRAM-LUTs. Due to the newer technology, the characteristics of the UT4 are consistently better, here even the 16 TS option can be operated within one clock at $200MHz$, while the 32 TS option can be operated with the system clock within one clock cycle latency.

UT3 Category	8 TS	16 TS	32 TS	64 TS	32 TS 2Clk	64 TS 2Clk
LUTs sequential	532	781	1 031	4 482	1 296	4 782
LUTs parallel	4 770	9 594	18 801	40 338	11 664	42 338
Frequency in MHz	206	194	120	99	221	172
Latency in Cycles	1	1	1	1	2	2

Table 5.13.: Synthesis results of the hit selection for the UT3.

UT4 Category	8 TS	16 TS	32 TS
Slice LUTs total	4 424	8 900	17 440
Frequency in MHz	249	231	154

Table 5.14.: Synthesis results of the hit selection for the UT4.

5.2.4.5. Scaling of Network Input Variables

Before the preprocessed inputs can be used by the MLP, they must be scaled to a value range within [-1:1] [91]. Here, the individual SLs can be processed separately again, so that all nine triples of inputs can be processed in parallel. In addition, the individual components of the triples are also independent of each other and can thus be processed in parallel. A separate module has been implemented for each individual component of the triple. Requirements with regard to flexibility are only required here due to different bit

widths for the inputs and outputs. In the following, the implementation and characteristics are discussed. Special consideration belongs hereby to the scaling of alpha, which is algorithmically simple, as it can be implemented by bit shift operations. The scaling of other inputs is more demanding thus separate modules were designed. The entire architecture of the scaling is additionally shown in figure 5.16.

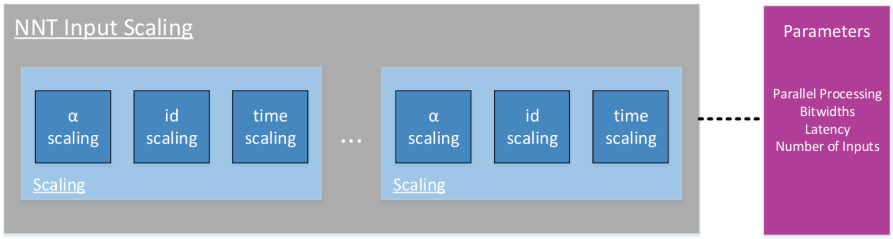


Figure 5.16.: Architecture for scaling the input triples before being processed by the MLP.

Scaling of the Drift Times

Scaling of the drift time is based on the relationship between the priority timing of a single TS and the used event time. It is basically the difference in event and priority time. The sign of the result is then determined by the LR information of that TS. Left will turn the value negative while right information will keep it positive. The result is then further processed with a correction that shifts the fractional part of the value to the correct position. This is implemented by a bit shift that is depending on the configured bit width.

Besides the scaling operation itself, this module includes the bounding of the drift time to the minimum and maximum allowed values. If the difference between event time and drift time is higher than a defined maximum threshold, the value is set to this threshold. In the base implementation, this value is defined as 256 which corresponds to a resolution in 2 ns. The threshold can however also be configured using the VME configuration registers. Bounding of the drift time to the minimum value only applies when axial-only event time estimation is used. In this case, it is possible that selected stereo TS have smaller drift times than the estimated event time. The drift time of this TS is then set to zero.

Scaling of phi

Scaling the value phi_rel is performed by subtracting an offset that is depending on the available TS ID within the respective SL. The offset is stored in memory beforehand and calculated at design time for quick access. It is dependent on the current CDC configuration and is implemented as part of the firmware generation framework. When adjusting the algorithm or re-training the network for a different detector geometry, the framework will update the respective parameters.

Evaluation of the Scaling Module

The synthesis results of the scaling module for both full parallel and sequential processing are presented for the UT3 in table 5.15. Generally, the resource consumption is small for both options. One thing to consider is that DSPs are used in this implementation. As one of the general ideas for the later MLP is to allocate DSP for neuron processing, here again, an implementation as slices can be further investigated is saving of DSP resources is required. The achievable clock frequency when using DSPs is at 272 MHz, well above the minimum target of 127 MHz.

Category	Sequential	Parallel
Slice LUTs total	76	895
Slice Register	90	398
DSPs	1	9
Frequency in MHz	272	272
Latency in Clock Cycles	9	1

Table 5.15.: Synthesis results for the implementation of the scaling of input variables.

5.2.4.6. Overall Evaluation of the Preprocessing

In the previous sections, all components of the preprocessing were presented and characterized individually. In this section, the entire preprocessing is evaluated. For this, the overall resource and latency characteristics are shown in table 5.16. On the UT3, the highest clock frequency that can be reached is achieved by using an implementation of multiplications as slices across all modules. This comes with the cost of additional demand for slices resources, however, it can be argued that freeing up DSPs for the MLP is of higher priority. Overall, the additionally required resources amount to around 2%. Due to the relatively low increase in resources, the option using entirely slices for logic operations is preferred. Even for the option using DSPs the utilization is as well quite low, as it only demands around 7% of the available slices of the UT3. The same holds true for the implementation on the UT4 but with even better resources utilization ratios.

The results show that the preprocessing is implementable on the chosen UT3 without demanded excessive resources. As the NNT has to fulfil strict latency requirements, the focus is shifted on that part of the architecture. The longest path of preprocessing in terms of required clock cycles is represented by the calculation of the event time estimation. The reason for this is that it can only be performed after all internal processing stages, except for the scaling, have finished. The latency of this path is at 13 clock cycles with an operating frequency of around 200 MHz. However, it is either only slightly above that frequency. Looking ahead at the integrated NNT, it is highly unlikely that these frequencies will be achieved when the MLP is included, which is going to demand the highest amount of the resources. However, it is highly likely to support operation with the targeted system clock frequency at 127 MHz. The bottleneck for the frequency within the preprocessing is located at the selection of the TS to be used. However, at this point,

there are still possibilities to achieve higher frequencies if necessary. One such possibility is the usage of the UT4, which is capable of achieving much higher frequencies in general due to the newer FPGA. The other option is to request the bigger UT3 that is based on the XC6VHX565T Virtex-6 FPGA, as it offers significantly more slices. However, this FPGA is allocated to the 3DS and should only be used in the worst case.

Category	UT3 Slices	UT3 DSP	UT4 VU80	UT4 VU125	UT4+
Slice LUTs total	27 515	19 179	25 506	25 506	25 199
Slice LUTs in %	7	5	6 %	4%	4%
Slice Registers	14 143	10 500	13 095	13 095	12 937
DSPs	0	18	0	0	0
BRAM	27	27	27	27	27
Frequency in MHz	206	199	254	254	272
Refresh Rate	1	1	1	1	1
Latency in Cycles	13	13	13	13	13

Table 5.16.: Synthesis results for the entire preprocessing of the NNT.

Besides the latency and clock frequency, the implementation is processing particle tracks at every single data clock cycle. It is thus capable to accept incoming 2D-tracks at every cycle. Since it can process a track at every data clock cycle, multiplexing can be employed to process the in total four tracks at the higher internal clock frequency within one data clock cycle. As it will be shown in section 5.2.5 it is due to the refresh rate of the MLP, that multiplexed processing of multiple tracks per data cycle is not employed for the implementation on UT3, as it does not offer significant improvements. However such multiplexing can be used for processing on UT4, which can allow for instantiation of multiple MLPs.

5.2.4.7. Summary

This section introduced the FPGA-based realization of the preprocessing for the NNT. It consists of three parallel data paths representing the different inputs to the MLP. Each data path is processed by a set of components that are specific to this application, but are designed to be flexibly configured in case changes to the architecture are to be made. The flexibility covers parameters such as the bit width, degree of parallelism and number of processed TS. Typical parameters to be used during operation were evaluated, with the key figures about resources, latency, and throughput provided for the targeted FPGAs. The feasibility of the architecture mainly depends on the number of TS to be processed in parallel. This number is the limiting factor for both latency and resources. The base configurations of 16 and 24 TS are well within reach of being realizable for even the older UT3 platform. Higher TS counts that are 32, are on the other hand infeasible for the UT3, but possible for the UT4, and thus for future upgrades.

5.2.5. Architecture of the Multi Layer Perceptron

The architecture of the MLP is mainly based on the considerations presented in section 4.4. As low latency is representing the most strict requirement to be fulfilled for the NNT, the tree-based architecture introduced in section 4.4.1 is used for the realization. In addition to this choice, multiplexed operation as discussed in section 4.4.1 will be employed. The reason for this choice is the total amount of MAC operations to be performed within the targeted network topology. This amount is exceeding the total number of DSPs available on the FPGAs supported by both the UT3 and UT4. Hence a fully parallel implementation is not possible for the platforms that are available for hosting the NNT. This choice is accompanied by the selection of an appropriate multiplexing factor that allows implementation without reducing the performance below the established requirements. The lowest latency can hereby be achieved by using a multiplexing factor of three. As a result, three MAC operations are mapped onto one DSP with the processing of one neuron being stretched across three clock cycles. This factor will result in resource utilization of 90% for the DSPs. Such a high utilization ratio is highly unlikely to culminate in timing closure for the FPGA design. This was observed for all setups of the NNT except the early prototype presented in section 5.4.2.1.

In order to increase the likeliness of achieving timing closure, a multiplexing factor of five is used in all of the setups used for operation. While this is increasing overall latency, the remaining latency budget is sufficient to compensate for it. Using this factor, both latency and resources are within the available resource budget on the targeted FPGAs, however, this choice results in a throughput below the maximum input rate. Using a multiplexing factor of five leads to an input refresh latency of six clock cycles. The high utilization ratio, meanwhile, allows only to implement one instance of the network. To match the incoming rate of tracks, a clock frequency of around 200 MHz has to be achieved. Implementation results show that this is not feasible for the Virtex-6 that is mounted on the UT3. However, it is feasible for the Ultrascale FPGA that is used on the UT4. The overall strategy here is to wait for the integration of the UT4 until the NNT is going to reach the required throughput. Meanwhile, during the time in which operation has to be fulfilled by the UT3, a reduced setup is used, that is not capable of reaching this throughput. However, FIFOs are used to delay newly incoming tracks for later processing. This approach is overall feasible and in accordance with the early stages of the experiment. In these stages only a rather low luminosity is achieved and thus fewer tracks are successively passing through the pipeline. Experimental data showed hereby that tracks are received at most for four successive clock cycles, which defines the sizing of the FIFOs in order to not lose any arriving tracks. In addition, this approach does not lead to exceeding latencies as the measured arrival times for trigger signals at the GDL are still within the targeted time window. The trigger system is then scheduled to be transitioned to the UT4 by the time high luminosity operation is going to start. This configuration together with the upgrade schedule for higher throughput is already sufficient to fulfil the minimum requirements for the NNT during high luminosity. The activation function is meanwhile implemented using pre-calculated content with all optimization options listed in section 4.3 being enabled. The reasoning behind that is the sufficiently low bit width used for the activation function, which avoids inefficient implementation.

While the smaller feature size of the UT4 is solving the throughput problems of the implementation for the UT3, it is still below the maximum achievable throughput. Further improvement, however, has to be achieved using parallel instances of the NNT. The limiting factor for this is the resource utilization. To address this shortcoming two additional optimizations, that were introduced in section 4.4, will be considered. These are pipelining across different layers of the network and heterogeneous resource usage. Additionally in order to facilitate resource saving, neuron multiplexing will be employed. The best configuration was achieved for using a MAC multiplexing factor of three and neuron multiplexing factor of three. Using these parameters a significant reduction of required DSPs can be achieved.

Category	UT3	UT4	UT4_Pipe	UT4_Pipe_H
MAC_MUX_HL	5	5	3	3
MAC_MUX_OL	5	5	3	3
Neuron_MUX_HL	1	1	3	3
DSPs	433	433	261	243
Slice LUTs in %	33	12	23	26
Frequency in MHz	127	250	225	150
Refresh Rate	6	6	6	6
Throughput in MTracks	21	32	64	41

Table 5.17.: Listing of the properties for different investigated architectures used for the MLP.

All these configurations are summarized in table 5.17. They are hereby all preferable for one of the optimization dimensions. Architectures based on tree-based neuron processing without pipelining are denoted as UT3 and UT4, which is representing the platform for which these results were achieved. Meanwhile, UT4_Pipe is representing layer pipelining and UT4_Pipe_H the usage of heterogeneous resources. The presented throughputs for these two options were hereby generated by using two parallel instances of the NNT, which was made possible by the reduction of required DSP resources. In addition, all schedules are illustrated in figures 5.17, 5.18 and 5.19.

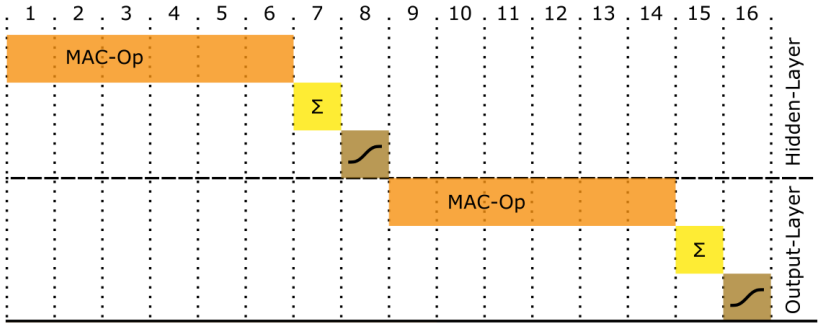


Figure 5.17.: Schedule of the base configuration used at the beginning of NNT operation [Poe18].

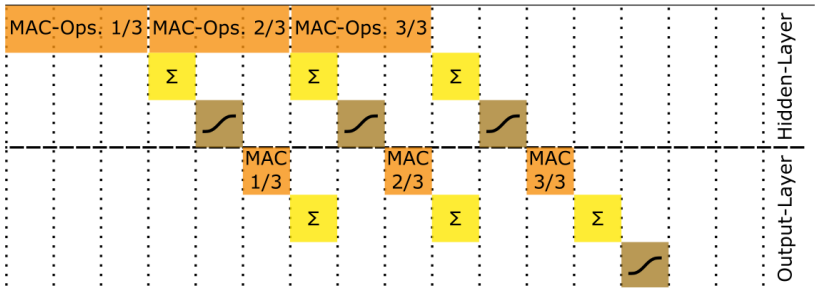


Figure 5.18.: Schedule of the pipelined configuration that achieves the lowest latency [Poe18].

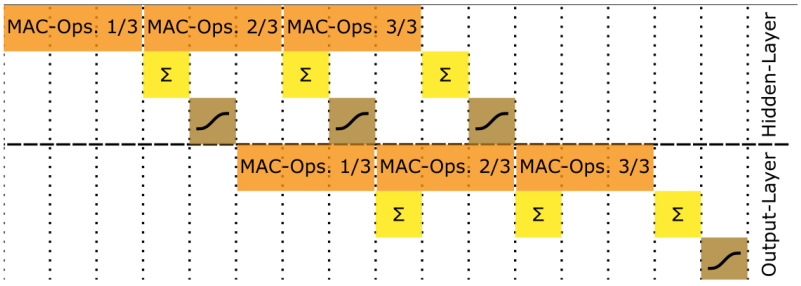


Figure 5.19.: Schedule of the pipelined configuration that achieves the lowest resources. This schedule is used for both UT4_Pipe and UT4_H [Poe18].

5.2.5.1. Analysis of the Memory Footprint

Runtime loading of weights is one of the key concepts of the NNT. The initial use case is hereby the loading of specialized networks for compensation of missing CDC data. This idea can be expanded beyond that use case by for example using specialized networks depending on predefined geometric areas of the detector, similar to the initial concept of the trigger. The general question is hereby, how many networks can be stored and loaded online during the experiment's operation. In the following both off- and on-Chip memory usage is analysed in order to answer that question. This investigation will however be constrained to the general topology used for the neural networks of the NNT.

Off-Chip Memory for Storing Runtime Networks

Off-Chip memory is providing the most resources for the storage of a network's weights. However, its usage is coupled with much higher latency added to the overall processing. In addition to this, it is impacting the architecture on the FPGA especially in terms of the required resources. In order to interact with off-Chip memory, additional logic is required that is handling the protocol. The additional latency is meanwhile depending on the amount of data to be transmitted. For this, the total data size of a network according to the targeted topology is calculated using equations 5.12.

$$\begin{aligned} WeightMemory_{HL} &= (Inputs_{MLP} \cdot Neurons_{HL} + Bias) \cdot BitWidthHL \\ &= 28 \cdot 81 \cdot 18Bit = 40\,824Bit \\ \\ WeightMemory_{OL} &= (Neurons_{HL} + Bias) \cdot Neurons_{OL} \cdot BitWidthOL \\ &= 82 \cdot 2 \cdot 18Bit = 2\,952Bit \\ \\ WeightMemory_{MLP} &= WeightMemory_{HL} + WeightMemory_{OL} \\ &= 40\,824Bit + 2\,952Bit = 43\,776Bit \end{aligned} \tag{5.12}$$

Assuming the basic configuration of the NNT that uses a bit width of 18 bits for each weight, 81 neurons in the HL, 27 input values and 2 neurons in the OL, the footprint of the network to be stored, without any compression, is at about 5 KByte. Using this value it is theoretically possible to store up to 16 different networks in the BRAM of the UT3. This is meanwhile not considering the impact of the effects on timing and routing for the signals. FPGAs generally have comparatively small on-Chip memory, but this allows access with very low latency. However, to store more than the theoretically possible 16 networks, external memory must be used. Since the latency for the calculation of the z-Vertex is a real-time requirement, it has to be examined whether the transfer of weights from the external memory is fast enough in order to stay within the latency budget.

Off-Chip memory can only be used when using a dedicated memory controller module. Such a module is provided by Xilinx with the MIG IP core [119]. In order to study the latencies for using off-Chip memory, this IP core is evaluated. The total latency for load-

ing the weights of an entire network can be determined by equation 5.13. The equation consists of two parts that are on the one hand the ReadPreparationTime and the DataTransferTime on the other. The ReadPreparationTime is constant and necessary to set up the transfer while the DataTransferTime is representing the actual latency for the data transfer and is depending on the amount of data to be transmitted. The ReadPreparationTime is fixed to 48 clock cycles when using the UT3. Meanwhile, when using the targeted topology the DataTransferTime is at 171 clock cycles until all data is transferred. The total latency for loading one network is thus at 219 clock cycles. The IP core can meanwhile be operated with a clock frequency of up to 400 MHz. This results in an update of the weights with a latency of 500 ns. This is already well above the maximum processing latency available for the NNT in its base configuration using the 2DS, which is set to around 300 ns. However, an upgraded overall architecture is presented throughout chapter 6, which is implementing both tracking approaches together on one FPGA. This will increase the maximum latency budget to around 800 ns putting usage of off-chip memory well within reach of being feasible. Further investigation for using standard off-Chip memory is part of the future work.

$$\begin{aligned} WeightUpdateTime &= ReadPreparationTime + DataTransferTime \\ &= 48cycles + 171cycles = 219cycles \end{aligned} \tag{5.13}$$

$$\begin{aligned} DataTransferTime &= \frac{NumWeights * BitWidth}{MemoryBusWidth} \\ &= \frac{2432 * 18Bit}{256bit/cycles} = 171cycles \end{aligned} \tag{5.14}$$

$$\begin{aligned} TotalNetworkTime &= InterfaceTime + WeightUpdateTime + MLPTime \\ &= 10cycles + 219cycles + 20cycles \end{aligned} \tag{5.15}$$

On-Chip Memory for Storing Runtime Networks

Considering the previous analysis it seems impossible to use off-Chip memory for the NNT with the initially allocated latency budget. This does not mean that the idea of dynamically loaded networks is not feasible, instead, on-Chip memory can be explored. While being much smaller in terms of the provided storage, the question remains how many networks could be stored. This number was investigated for both possible platforms, UT3 and UT4. In both cases, network weights were stored in BRAM and not in distributed RAM, as it is less efficient for this type of data. The FPGA available on the UT3 is capable of storing up to 8 networks in parallel, with the overall BRAM demand increasing to 66% [Wu16]. The UT4 is much more powerful with its upgraded FPGA, providing six times more BRAM blocks compared to the UT3. As a result, it is capable of hosting up to 48 networks, without consideration of resource optimization due to weight reuse across all these networks.

5.3. Tools and Monitoring for the Neural z-Vertex trigger

One of the requirements of the NNT is flexibility with regards to being reconfigured with a specific set of network weights depending on the current state of the experiment. This applies to not only the network itself but also its preprocessing. An adaptation of the overall architecture should be carried out as automatically as possible to save time and reduce the risk of errors. For this, a semi-automated framework for firmware generation was developed and is presented in the following. In addition to this, all aspects regarding the DQM will be presented, together with the toolflow and methodology for verification.

5.3.1. Semi-Automated Generation of Firmware

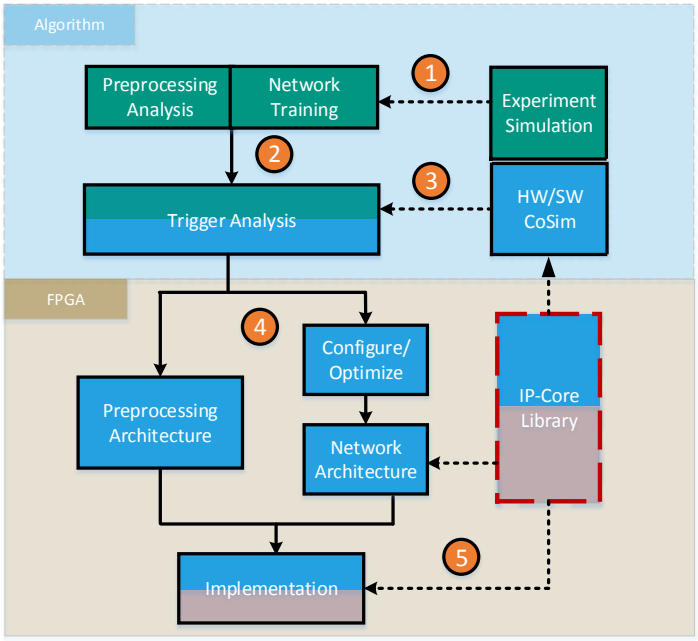


Figure 5.20.: Semi-automated design flow used for the generation of NNT firmware. The sequence of the design flow is indicated with numbered circles.

The FPGA-based implementation of the NNT has several parameters that have to be chosen during the design time, such as the bit widths to be used internally and the weight sets of the neural networks. Especially the latter type of parameters is depending on the current status of the experiment and is going to change frequently over time, which is the main motivation for developing a framework that generates all files related to VHDL

without too much additional manual involvement. A graphical representation of the entire design flow that is supported by this framework is shown in figure 5.20 and will be discussed in more detail in the following. The presented framework is hereby based on the Ref. [Bae18a] by this thesis's author.

The two domains of algorithm and FPGA-based design are separated in the design flow. Here, the algorithmic side is using a simulation of the experiment to train the network with simulated or experimental trigger data and mostly handled by physics experts. At the same time, it is the part at which the preprocessing is refined and evaluated. This process is going to be active and iterated throughout the lifetime of the NNT. Examples of iterative changes after the initial concepts are the TS-based event time estimation and the drift time threshold approach. Information between both domains is conducted by a self-defined configuration file, which is described in section A.1.2.

The FPGA-based design is then supported by a custom framework written in C++ that takes care of all tasks related to translating the configuration file into an architecture specification that can be implemented. Based on a predefined and implemented IP core library the framework selects and parametrizes the required IP cores as defined in the configuration file. In addition to that, it is loading SW-based models of the IP cores, that were custom-developed in order to emulate the new processing architecture. This has the goal to generate reference input and output data samples to be used for subsequent validation.

As two groups are working parallel to each other abstraction is used for the fast and efficient design of trigger algorithms. Tasks in the flow are then separated in algorithm and physics specific which are shown in green, while tasks that are part of the FPGA-based design are coloured blue. Following traditional approaches to digital design, development for FPGAs is additionally separated into RTL and technology-specific steps, highlighted in light rose. The sequence of tasks to be performed is partitioned into five separate steps. An additional option, inspired by the state-of-the-art neural network generation for FPGAs is evaluating the influence of bit width truncation and general compression of the network [Reu18]. However, these are often surpassed by the ability of the FPGA tools to minimize resource consumption. A comparison in z-Vertex estimation is at the end performed by using HW/SW Co-Simulation using the concepts described in Ref. [99] that even allow the coupling to the trigger simulation framework.

5.3.2. Interfaces and Levels of Monitoring

The NNT has far-reaching influence over the entire DAQ of the experiment as it has the capability to generate triggers signals that are used for the decision whether to read out the detector for a given event. In case that the signals are wrongly estimated for an observed track which has an origin within z=0, a valuable physics event might be discarded. This makes correct functionality critical for the entire experiment. In order to not harm subsequent physics analysis, it must be ensured that the estimations of the NNT do not deviate too far from the real origin for a given track. In case that large deviations are occurring during online operation there must be mechanisms to ensure a quick reaction, for example, by excluding the NNT from the decision logic. As a result, the experiment may lose its z-Vertex trigger, however, it is ensured that no data is lost without knowl-

edge about its reasons. An online monitoring concept is hereby required to provide this functionality, which is the main focus of this section.

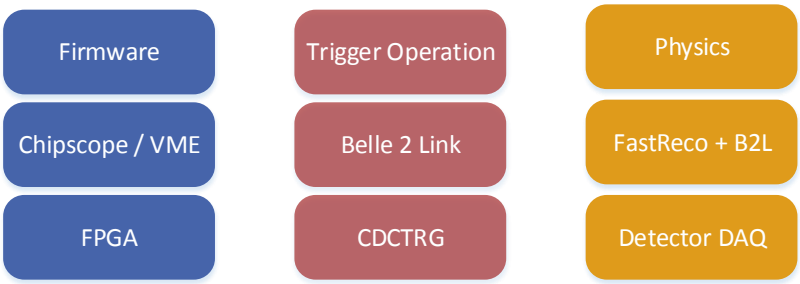


Figure 5.21.: Different of scopes of monitoring at the NNT together with the responsible interfaces.

The overall monitoring of the NNT can be divided into several areas. These are divided by the scope of functionality that is possible to be monitored, as shown in figure 5.21. Hardware monitoring is describing mechanisms that can capture internal processing within the FPGA. It can be used to debug, validate and analyse the firmware together with its data sources interfaces. However, it is limited to one specific FPGA board, as such monitoring cannot be performed beyond the boundaries of a single hardware platform. These limits, for example, the analysis of the ratios of tracks processed by the 2DS and the NNT which might indicate problems due to loss of tracks. Monitoring across different platforms is the scope of the second area, which is in turn not able to generate in-detail data about the internal processing on the FPGA. It rather provides a data collection in which reduced event data can be analysed across all boards. This allows the validation across all systems and thus qualitative statements about the NNT. The final area of monitoring is encompassing the entire detector, which is the trigger readout, detector readout, and subsequent physics analysis. Precise statements about the quality of the NNT are possible when both the online and the offline data from the precise reconstruction of an event using detector data are available.

Depending on the respective area, suitable interfaces on the FPGA must be selected that provide the necessary data. In addition, they also require expertise in the domain under consideration. For example, the clock cycle accurate verification of the hardware processing must be performed by the firmware designer or expert, while communication details additionally require knowledge about the CDCTRG. The final validation that needs event reconstruction requires knowledge about the physics experiment and BASF2 expertise. As this thesis is focused on the hardware part of the NNT, this layer will not be further discussed but can be found within documentation provided in Ref. [90].

First of all, suitable interfaces must be established in order to gather the necessary data from operation. There are two possibilities to get the necessary data for validation of the firmware, these are VME and Chipscope that using a Joint Test Action Group (JTAG) con-

nection. For validation across the CDCTRG and even take CDC data into consideration, only the B2L remains a viable choice as it is the only interface that can provide synchronization of data across all systems. However, even that is not sufficient as it requires substantial software implementations for offline data extraction. The entire process of monitoring based on B2L is shown in figure 5.22.

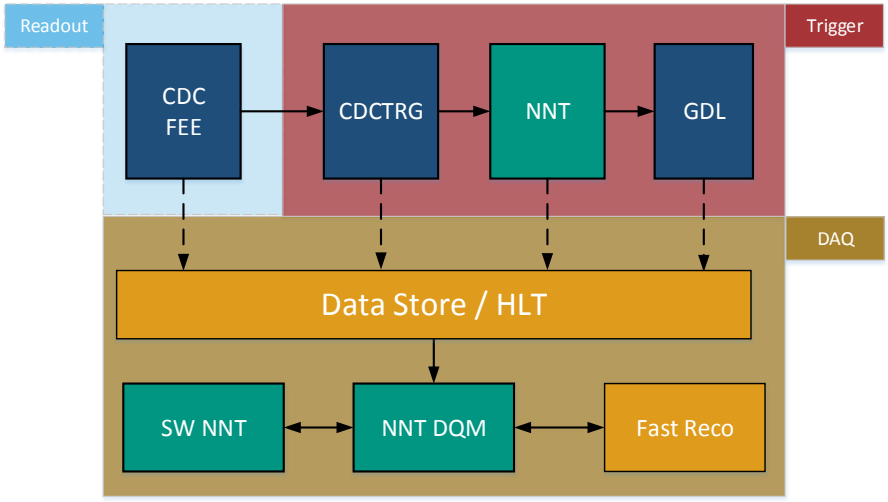


Figure 5.22.: Overview of B2L-based DQM. Boxes shown in green are dependent on the NNT and its current configuration. In the equivalent system for S3D these boxes are replaced accordingly.

Hardware Interfaces for Monitoring the NNT

Many interfaces are available in the CDCTRG for implementing monitoring of the NNT. While it is possible to use custom interfaces, the NNT is mostly relying on the commonly used options as they are already sufficient for providing the necessary features. None of these interfaces alone is able to cover all levels of monitoring, as a result a combination of all of them is used. The individual interfaces Chipscope, VME Bus and B2L were evaluated with regard to their relevant properties.

One of the key characteristics to be considered is the resolution in time at which data can be recorded and received from the NNT. Verification of the design's internal data processing needs a clock cycle precise recording of internal signals. The only option allowing this, is the usage of Chipscope. It is capable of recording data with a custom frequency on the FPGA and is therefore scalable with the operating frequency selected for the NNT. In contrast, any kind of data recorded via B2L is fixed to a frequency of 31.75 MHz. Coupled with the size of internal signals, this basically prevents high-resolution verification of the processing architecture, as it is targeted to be operated with much higher frequency. The

VME bus is meanwhile accessed via relatively slow register accesses that have a latency of around 0.8 s that was measured on the respective computer within the BDAQ. It is thus not suitable for this task. Such high latency access can be used to read out general information about operation, such as the health of communication links or operational statistics such as a distribution of the z-Vertex estimation created online .

Another criterion is the availability and accessibility of the respective interface during operation. Herein lies the advantages of the VME bus as it is always available after the board is powered up within the hosting crate. For operation on the UT3 an additional IP core has to used in order to use VME. This situation is improved with the transition to its successor UT4, which is equipped with a dedicated Artix-7 FPGA that is handling all related tasks. This has the advantage that the VME bus can be used even if no valid bitstream is loaded on the main FPGA. The Chipscope interface, on the other hand, requires a connection to a local computer via JTAG. In principle a continuous availability can be established, however, it can only be reached while being connected to the local network of the KEK as the computers are part of BDAQ. Therefore a tunnel has to be set up for access from outside of Japan. In addition, the JTAG interfaces on the local computer are shared across several modules and therefore only available to a limited extent. In theory, the B2L also has continuous availability, however, it is shared by all participants in the overall system and combined in the DAQ. The problem here is that incorrect use by one participant can cause an error state for the entire system. Such error states are not only the result of the wrong usage of the interface, rather even the smallest timing violations are sufficient to invoke instability.

The different interfaces of choice have different scopes of covering the data flow across the CDCTRG. Both VME and Chipscope are limited to one board. This allows the recording of local processing signals and IO behaviour. However, there is no easy procedure of combining these signals across the boundaries of individual FPGA platforms. Herein lies the main advantage of the B2L as it is the means of recording data of the complete readout chain starting from the FEE of the CDC all the way to the GRL. At the same time, the B2L contains the information about each platform's relative delay via B2TT and can thus be used to match locally recorded data across the entire data processing chain.

Using each interface on the board usually requires an implementation of the communication protocol on the FPGA. Their implementations can have a significant impact on the overall design due to the increase of resource consumption or the impact on the achievable performance. Out of all interfaces, only the VME access is transparent to the entire design, when using the UT4, as it is implemented at a dedicated external FPGA. On the UT3 its implementation is requiring resources, however, overall demand is quite low. The more problematic aspect here is its influence on the achievable operating clock frequency, as the current implementation often introduces timing violations. Using debug cores as part of Chipscope, on the other hand, requires significant additional resources while severely influencing the routing of the signals and thus influence the internal signal propagation time. As a result, the generation of the new firmware is in general much more time-consuming when enabling this interface. The B2L related IP cores have the biggest influence on the overall design. Not only are these cores requiring a lot of resources, but they also have a significant negative impact on the timing of the entire system. During all implementation runs, timing violations at the B2L related transceiver ports were occur-

ring regularly and could thus far only be solved by the exploration of different place and route strategies.

Another important characteristic of an interface is its capability to capture specific time windows around interesting events. Additionally, high sample size is needed to achieve large scale validation, as single events are insufficient to achieve high test coverage. The B2L is particularly suitable for this since it is capable of recording a large fraction of the events during a run of the experiment. Large-scale verification with statistical evaluation can then be carried out afterwards. Chipscope, on the other hand, is capable of buffering data across at most up to 4096 clocks following a custom trigger that starts the recording. Data taking is hereby rather tedious compared to B2L since it is not automatically performed. VME hereby has the lowest capabilities as it is very slow and limited to 32 bit register accesses.

This concludes the evaluation of the available interfaces with regard to the most important aspects. An overview of them is provided in table 5.18.

Property/Interface	VME	Chipscope	B2L
Precision	seconds	clock cycles	single events
Availability	always	always	limited
Design Impact	none	high	very high
Scope	local	local	entire DAQ
Data rates	32 Bit	4096 Bit	B2L data width
Time granularity	arbitrary	triggered	per event
Accessibility	BDaq	local computer	KEKCC

Table 5.18.: Overview of the different available interfaces with their characteristics.

Offline-Validation

The data written out by the NNT via B2L is available for usage within the internal DAQ and KEK computing cluster, depending on the time passed after the experiment. At both stages, this can be used to facilitate the validation of NNT hardware. The mechanisms that were developed for offline validation are discussed in the following.

Data captured during an experiment is at first available at worker nodes within the internal DAQ network. Only after a certain amount of days or by request are they made available in the collaboration's computing cluster. As data size is quite substantial it is not feasible to do any heavy processing within the DAQ network, thus this should be in general done at the computing cluster. The first step is here to either wait for the availability of the data or to initiate manual transport to the cluster via the worker nodes.

The data to be processed is stored in the root format, which is typically used within data storage in physics experiments as many tools are based on the accompanying root toolkit [11]. Using this format and tools is, for example, the basis for most of the generated distributions shown throughout section 5.4. However, this format is not particularly suitable for debugging of the hardware, since it required knowledge about the toolkit.

As such it is not possible to directly generate signal diagrams that are typically used for debugging hardware as they show the behaviour over time. At the same time, the format is not directly supported by any hardware simulator. These aspects are addressed by the usage of the B2VCD tool that is providing functionality for converting [101]. It is able to convert the B2L data in the sroot format into a Value Change Dump (VCD) file. VCD is a standard format used to describe signal behaviour over time and can be directly used by most hardware simulation tools. Furthermore, additional conversion into simulation scripts that can be used with Modelsim tools is quite simple. For the conversion, the signal definition of the read out data is required together with the information necessary for identification of the board to be used.

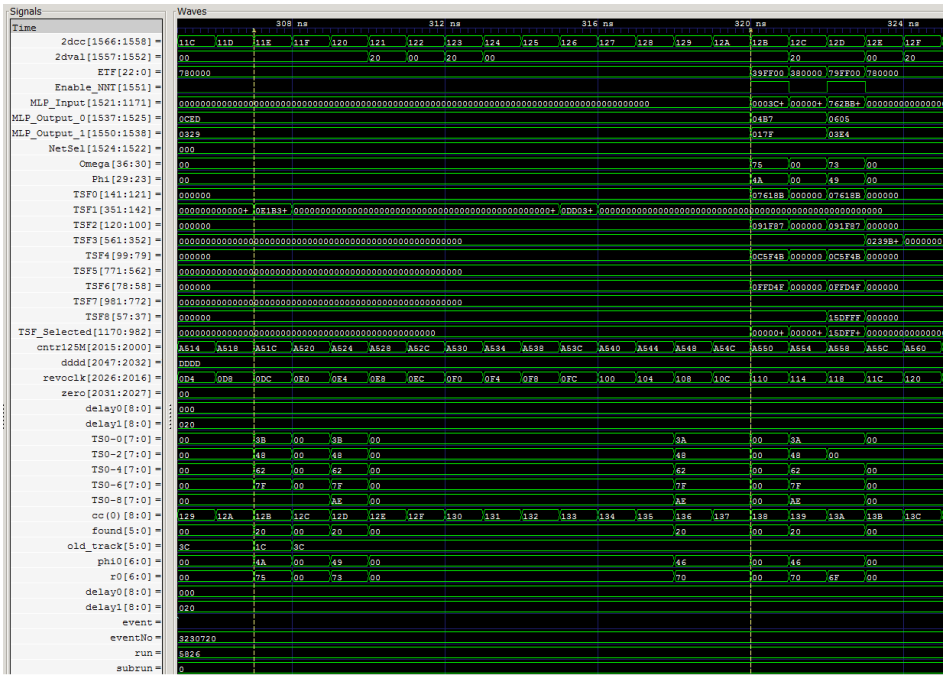


Figure 5.23.: A waveform showing data signals received via B2L from one board that is hosting the NNT. The data was recorded during run 05826 in experiment 10. The GTKWAVE tool is used to visualize the waveform. Data belonging together is indicated by the yellow marker, with 2DS data being valid before NNT data.

The identification information is represented by the COPPER and FINESSE IDs. These IDs are unique for each board in the DAQ of the experiment. They are not only limited to identifying the respective boards that are hosting the NNT but also every other component of the CDCTRG using a B2L connection. Another advantage of using VCD files is that they can be easily visualized with the help of software tools such as GTKWAVE [30]. A visualization using this tool is shown in figure 5.23. In this case, the information read

out from one NNT and 2DS quadrant is shown together with status information about the experiment such as event and run number.

The basis of the employed offline validation for hardware-specific operation was laid out with the introduction of the data conversion from root files to VCD. Before the used process is discussed in more detail it shall be noted that it is a rather general approach independent of the NNT itself. It can thus also be used for the validation of the S3D that is discussed throughout chapter 6. The difference to the NNT is solely in the usage of different sub-components to be read out for validation.

The validation process is based around the recreation of the behaviour at the interfaces of the respective board as it was observed during the experiment for usage in hardware near simulation. For this, the script SimGen was developed that is using the respective VCD files and translates them for the simulation. This translation includes the conversion of the VCD format to the data format and protocol that the respective data sources are using for transmission over GT. With this approach, the design can be simulated while recreating input behaviour on clock cycle granularity as it is during operation. With regard to the implementation, processing of VCD files is based on using the PERL module Verilog::Vcd [105]. It provides internal tree structures for easier organization of single signals read from these files. The script is then translating the contents of these tree strictures to the used simulator's specific formats, for example, the commands interpreted by the Modelsim simulator. In addition to this translation, the scripts is supported the verification process by automating necessary tasks for setting up the conversion. These tasks include the selection of the required data source and splitting up large simulation files into smaller chunks in order to reduce runtime for simulation of single time windows. As the data from the different sources does not include any temporal relationship information, they are rather processed independently. The script is merging these different sources and combing them with regard to a global time scale.

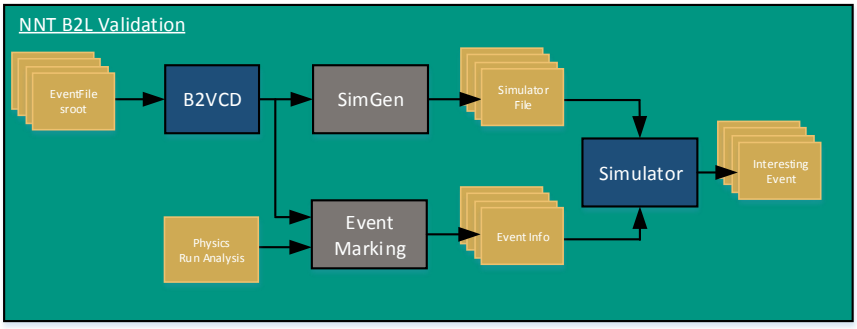


Figure 5.24.: Used process for hardware-specific offline verification based on B2L readout. As it is not specific to a certain trigger component it can be used for both the NNT and the S3D.

With the generated simulation scripts, it is possible to recreate operation on the granularity of single clock cycles. This level of detail in simulation is allowing to study operation with the help of HDL simulators. Such a simulation is provided more detailed insight into the internal processing than any kind of debugging interface since all signals can be recreated while the read out data is representing a snapshot of the entire internal processing. This allows to find and understand the problematic section of the implementation more easily and thus to verify correct firmware operation with regards to the HDL description.

These scripts on their own are however not enough to enable efficient debugging and validation. This is mostly due to the massive amount of debug data to be processed, as each run is consisting of many events in the order of more than 1000. It would be rather hard to identify single problematic events on low-level granularity. In addition, the software-based DQM is only processing a subset of the hardware due to its internal implementation, in which TSs have to match predefined criteria in order to be used. Large-scale validation of the entire run has its advantages, but there is a need to identify the events that are leading to high deviations between simulated and real NNT. To allow easy identification of these events, an additional script was developed. Its task is to merge the software-based DQM report with the simulation scripts in order to highlight every event that is of particular interest for debugging. This step of the process is named event marking. It is aware of the HDL simulator's timeline and provides points in time at which to look for in order to find the events of interest. The whole validation process is illustrated in figure 5.24.

5.3.3. Slow Control

As a part of the electronics controlling the detector's operation, the NNT must also support SC being a trigger component. This is implemented according to the scheme described in section 2.3.3. However, the concrete technical implementation for the related aspects such as the connection to the conditions database can be designed by each subsystem, for example, the trigger system. However, two constraints have to be considered for every sub-system in order for them to be included in the overall SC. First, the archiver can only be accessed using EPICS. The condition database, on the other hand, can only be accessed using the NSM protocol. The main application for the SC in trigger operation is to monitor, control and configure single boards. As the general trigger system is based around using VME crates to host each hardware platform, the communication concept for the SC is to access the systems over the backplane of the crate. Access is meanwhile performed by the master CPU within the crate. It is capable of issuing backplane access to every board in the crate, but also provide the connection to external computers. The responsibility for accessing certain boards is divided across several separate NSM processes running in parallel on the master CPU. Both control and status data is communicated with a dedicated trigger server over NSM. The software package available at the server includes the capability to convert data into EPICS in case it is necessary with the goal to access the archiver. Meanwhile, it is directly communicating with the conditions database using NSM.

When considering the implementation of the SC on the NNT, this approach in principle fulfils all requirements. It has the added advantages that tools and reference implemen-

tations for the implementation are either already available or developed in collaboration throughout the trigger system. It can thus be reused and customized to the special needs of the NNT. However, this scheme has one severe disadvantage, which led to the choice to develop a modified version. The weakness is present at the transition from NSM to EPICS. The introduced indirection towards the archiver is in principle unnecessary, as EPICS can be used directly and provides the same base functionality. The conversion can lead to problems as it requires the additional implementation of basic functionality, but even more significant is that EPICS is designed to record more information about the status of a component than NSM. One result of this difference is that lossless conversion of data towards EPICS is not possible nice since some status data is not recorded with NSM. The last disadvantage of this approach with regard to the technical implementation. The conversion leads to unstable operation during early operation and the development phases. Here, process variables were sometimes generated at every single second such that it exceeded the receiving capabilities of the archiver.

After evaluation of all possibilities, there are in principle three separate approaches that can be employed for the NNT. These are a solution following the trigger-based NSM approach, an approach in which only EPICS is used and a heterogeneous approach in which both EPICS and NSM are used by instantiating separate processes in parallel on the trigger server. As the NSM-only approach used throughout the trigger system was already discussed, the following discussion will focus on the alternatives.

Instead of using NSM for implementing SC for a trigger board, the board can instead be controlled directly by an EPICS process. As a result, the data intended for the archiver can be forwarded directly without having to be converted at the trigger server first. By bypassing the conversion, no information is lost and generating excessive workloads at the archiver can be avoided. However, the connection to the condition database must also be taken into account. In principle, it could also be operated with EPICS, but the tools required for this would have to be re-designed and re-written. The estimated effort required for this is quite high, as the implementation has to be thoroughly tested before it is allowed to be used in the processing environment of the detector. Alternatively, the approach used at the trigger system can be reversed at this point by converting the data from EPICS to NSM on the trigger server. Here, however, data is lost again due to the incompatibility, in addition, conversion software must be created. The implementation of this conversion will not be easy to implement, as the currently existing problems of the NSM to EPICS converter already showed. Compared to the general approach, improved connection to the archiver is achieved by avoiding the need to implement a connection of the conditions database. This approach has the additional disadvantage that software has to be developed without any collaborative support. Thus the EPICS-only approach is even more unsuitable than the general trigger solution.

Both of the previous approaches have their drawbacks due to the need to use both protocols in order to be compatible with the data sinks that are representing the SC in Belle II. For this reason, a heterogeneous approach makes sense in order to avoid the disadvantages of using strictly one protocol. The approach is in principle based on spawning two separate processes at the master CPU within the VME crate. One process is representing the EPICS functionality which is tasked with reading out the values related to the archiver via the backplane. These values are then forwarded to the archiver without any detour

through the trigger server as no conversion of the data is necessary anymore. At the same time, no information, such as timestamps, is lost when using this approach, since data is natively stored in the format supported by EPICS. In parallel to the first process, another process is instantiated, which has the responsibility of handling all tasks related to the conditions database. This process is implemented using NSM to avoid any additional conversion. Few things change with regard to the implementation on the trigger server. Here the master NSM is communicating the dedicated process present at the crate, while no additional processes for the conversion are needed anymore. An additional advantage of this approach is that this represents the original use case for both EPICS and NSM, as they are collecting and distributing data independently without any need for interaction. With this separation the expertise present within the collaboration accumulated by both groups of developers and their reference implementations can be used to ease implementation. The structural setup of this approach is shown in figure 5.25.

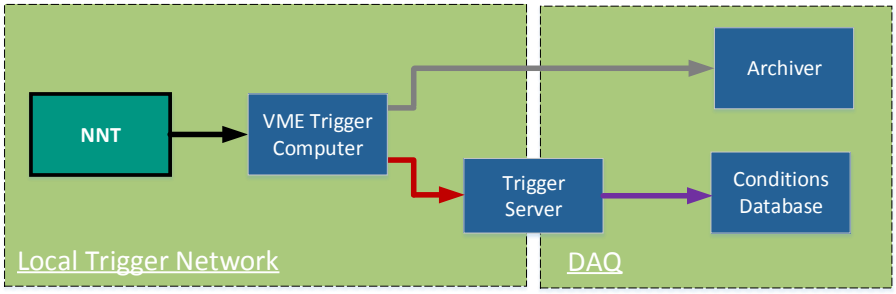


Figure 5.25.: Architecture of the slow control scheme used for the NNT. Direct access to the trigger board is performed over the backplane available at the VME local trigger computer. Two separate data streams are fed into DAQ, with data related to the Archiver indicated by a grey arrow, while condition variables are coloured red and summarized with all trigger data.

This approach is functionally the best solution for SC at the NNT since no data is lost and all conversion problems are bypassed. However, it is not entirely without its disadvantages. The first, is the increase of complexity for all future maintenance tasks, as two protocols have to be supported in parallel. Both alternative solutions are meanwhile relying on just one protocol. As this solution is requiring the instantiation of two processes in parallel on the local VME trigger computers, these computers will experience an increase in their workload. The same applies to the backplane of the crate, as two processes are now trying to access it in parallel. Both of these drawbacks should not be problematic during operation for the NNT as the used boards are sharing their crates with only a few additional boards as illustrated in section 2.3.3.

Functionality is the main priority for the final implementation of SC and thus has the highest impact on the choice of the used approach. Loss of data, as well as unstable operation, have to be avoided as far as possible. The other two criteria, such as implementation effort and maintainability play subordinate roles. Although the approach cannot be used in

case it exceeds the processing capabilities available at the VME crates, their master processors are not heavily utilized. For this reason, the heterogeneous approach is pursued for the NNT.

With regard to the hardware implementation of the NNT, this decision has no immediate impact. Access via the backplane is seen the same way for the SC software as such it is independent of the type of process. These accesses are represented by simple VME register access. Data that needs to be read out for SC is provided by a register set accessible over via VME. The outline of the address space for these registers is shown in table 5.19. It encompasses the register space that is dedicated to the slow control. The remaining address space is providing basic functionality for operation and test of the NNT. For example, the current build version of the firmware is part of the general information.

Address-Space	Description
0x0000-0x00FC	General UT3 information
0x0100-0x01FC	Flash Access
0x0200-0x023C	Play and Record
0x0300-0x06FC	CDCTRG Flow Control
0x0500-0x05FC	Configuration Registers
0x0600-0x06FC	Slow Control

Table 5.19.: Overview of the register address space accessible via VME that is encompassing the registers dedicated to the SC.

5.3.4. Data Quality Monitoring

As described in section 5.3.2 the only possible way to send data from a trigger module to be used for DQM is using B2L. This is the only interface within the DAQ that allows synchronizing data across the borders set by the separate hardware platforms. The realization of DQM can be divided into software and hardware related parts, while software has the most tasks to fulfil as it has to be integrated into the DAQ environment. This is always a delicate matter as it has to operate stably to avoid system-wide instability. With regards to the hardware-based implementation for the NNT, the tasks are to sample and transmit corresponding data to the IP core responsible for B2L communication that is provided by the developers of the DAQ. This part is going to be the focus of this section.

DQM Data Format for B2L

The data format and protocol used for transmission via the B2L is the responsibility of each sub-trigger, while the supported data rate is being dictated by the entire system. Data taken this way is meanwhile not only suitable for the validation of the functional logic on the NNT but also for validation of the interfaces to all of the connected sub-triggers. It can be matched across systems and can thus validate correct transmission. As a result,

data sent this way is generally divided into the two groups NNT and CDCTRG data. CDCTRG data contains the entire input data received from all used sub-components. This data is hereby without being captured by any stage of processing besides storage itself. The data is written to the buffers of the B2L directly after reception data and is aligned with data taken from other sub-components using shared clock counters. Whether data recorded using the B2L IP core is actually written out and made available for offline processing, is determined by predefined conditions. Data rates in the DAQ are hereby quite restricted. As a result, most of the data taken from trigger components during operation is suppressed. First of all, events are only written out in case a condition is fulfilled, for example, in the presence of a 2D-Track. Additionally only a fraction of the set is written out, for example, 10% of all events that fulfil the conditions. Additionally, data is not only written out for one specific clock cycle, but it also represents the accumulation across a predefined time window. In the early stages of operation, this window was set to 48 system clock cycles. The offline software that finally analyses the data must hereby know the corresponding protocol of all input components to process the data correctly.

The NNT data is used purely for the validation of the algorithm's implementation, at which a distinction between preprocessing and MLP is made. The most important components here are the intermediate values generated at the internal processing stages. These values are the selected TS, the network used for processing and all of the 27 input values to the MLP. The MLP is meanwhile validated by writing out its output values. These values can be additionally used to cross-check with the GRL for correct data transmission. A detailed list of the values that are written out via the B2L for the DQM can be found in figure A.1 for operation with both 10 TS and 15 TS.

Implementation of the Hardware-based Aspects of the DQM

With regard to the implementation of the DQM on the FPGA, an additional module named NNTScope was developed. Its main task is to sample all of the requested data and forward it to the B2L IP core. Since the NNT is operated in a pipelined way, intermediate values have to be stored in a way that makes it easy to be matched with all the data that is related to the processed track. Data to be matched has different latencies spanning across several system clock cycles. It can then be matched by using the characteristic that the implementation on the FPGA is dedicated and deterministic. The recorded data can then be matched offline with an analysing software that correctly applies the latency delays to each of the read out values. However, this proved to be difficult to implement during operation, as the internal processing is controlled by another clock source than the B2L. This led to different readout timings across the boards that are hosting the NNT. To avoid this problem, all data to be sent over B2L is synchronized internally within the module. As a result, it is buffered and delayed until the generation of the output values concludes, instead of being written out immediately.

5.4. Evaluation, Operation and Validation of the neural z-Vertex Trigger

5.4.1. Setups for Testing the NNT

As the development of the detector's infrastructure is a long term project sub-components only become available over time or even late into the process. These updates make it necessary to establish setups for early testing to evaluate the concept and implementation as early as possible. Since the NNT is not a standalone component and is depending on many data sources for its operation, the setup must reflect the final CDCTRG as realistically as possible to achieve representative results. For this purpose, two different test setups were set up at the experiment's facility that facilitates the testing of trigger components. Their main purpose is to provide an emulation or recreation of the data flow as it is designed to be during the operation of the experiment. These two setups are the experimental and merger play setup.

Category	Merger Setup	Experimental Setup
Test Data	simulated or pre-recorded	CDC readout data
Data Flow Coverage	complete	partial
Repeatability	deterministic	indeterministic
Robustness	errors not critical	errors influence entire readout
Availability	any time	depending on operation

Table 5.20.: Overview and comparison of both available test modes.

The difference between both options is in the emulation of the CDC's readout. In the experimental setup, the complete data flow of the CDCTRG is enabled, here data is read out from the detector and injected into the CDCTRG via the FEE. This setup corresponds to the one used in the actual experiment. It has the big advantage that the data taken here corresponds to the actual behaviour of the detector. Its main application case is for testing in combination with cosmic rays, in which activity distributions of the detectors are easier to correlate with simulation since the types of observed tracks are restricted, while background events are comparably rare as no beams are injected. Tracks are occurring with a much lower frequency and are typically going straight through the detector, making them easier to process.

Besides the validation of the algorithm's performance on the FPGA this setup can be used to test the connections to the entire data processing chain including the connection to the DAQ over B2L. An illustration of this setup is shown in figure 5.26.

While the experimental setup is representing the best test mode for validating the overall concept, it has some drawbacks when testing the FPGA implementation. For example, trigger components can only be tested if the FEE of the CDC is actively taking data. For this, the CDC has to be powered up which has to be coordinated with the experiment's operations group. At the same time, any component under test is influenced by the erroneous operation of the FEE or problematic data connections. During the early stages

of testing, many of the mergers responsible for SL3 were not connected to the detector's readout. As a result, no data was received from this SL, which limited the testing capabilities for the NNT as it is depending on all SLs. This is not the only drawback of this setup. Since the data is received directly from the experimental operation, repeatability of the tests is very difficult since the time intervals until input patterns are repeating are rather large. In addition, the configuration of the NNT in this setup is influencing the entire readout system. In case updated firmware is containing errors that influence, for example, the flow control, the entire trigger chain might be endangered of being compromised. As the system is shared and cosmic ray data is taken for multiple purposes, it is undesirable to use this test setup for early development.

These disadvantages are avoided with the help of the merger play setup. Here, test data patterns are first written to internal memory present at the mergers. These patterns are then periodically fed into the CDCTRG after the data flow has been established. The data used for such tests can be generated by the user and is thus used as test stimuli for the system by containing combinations of input values that are cover defined use cases. Another option is to use this setup to recreate the occurrence of previously recorded data sets that caused problems in order to gain more in-detail information in combination with Chipscope debugging. Recorded data can hereby be generated from the simulation of the experiment or data derived from actual operation.

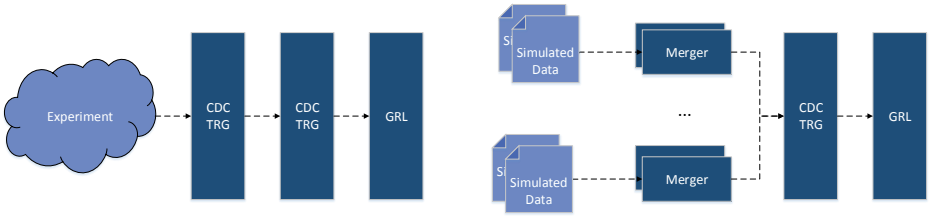


Figure 5.26.: System architecture for both the merger play (a) and experimental (b) test setup used for components of the CDCTRG.

The drawback of this setup is first and foremost that the real behaviour of the detector and coupling the NNT to the complete readout system cannot be investigated. However, it allows easier debugging of the interfaces and the functionality. Input data is always complete since it is independent of the FEE and its current cabling situation. The tests are additionally highly repeatable as the received data is defined by the user and repeated in a loop. Experimentation with new firmware is hereby, in general, easier to perform with this setup since errors in the NNT are not critically influencing the work of other developers as the CDCTRG is excluded from the global readout. The structure of this setup is shown in figure 5.26. A comparison between both test setups with respect to their properties is presented in table 5.4.1. As no setup is suitable to cover all kinds of tests, both are in general used throughout the development.

5.4.2. Configurations for Operation in Belle II

Several different versions of the NNT were developed during the early stages of the experiment. These were tailored to be operational with the current status of the readout chain. The main goal of these early stages of operation was to prove the early correctness of the concept with the available resources. Although the NNT was already operational with reduced functionality in these phases, only, later on, was it possible to actually send trigger signals. This chapter will present and discuss the main versions of the NNT that were integrated into the CDCTRG.

5.4.2.1. Reduced Setup for Tests with a Partially Available Trigger System

The first version of the NNT was developed to achieve first insights into the behaviour of the CDCTRG and the detector. The goal was here to have a version that could participate in tests with cosmic rays. This setup was based around using CDC data that was generated by cosmic rays going through the detector was used using neural networks and preprocessing specialized on cosmics that already provided conclusions about the correctness of the estimation without the need for an injection of particle beams. At the same time, the communication infrastructure was investigated, especially the correct implementation of the protocols for all of the interfaces to be used.

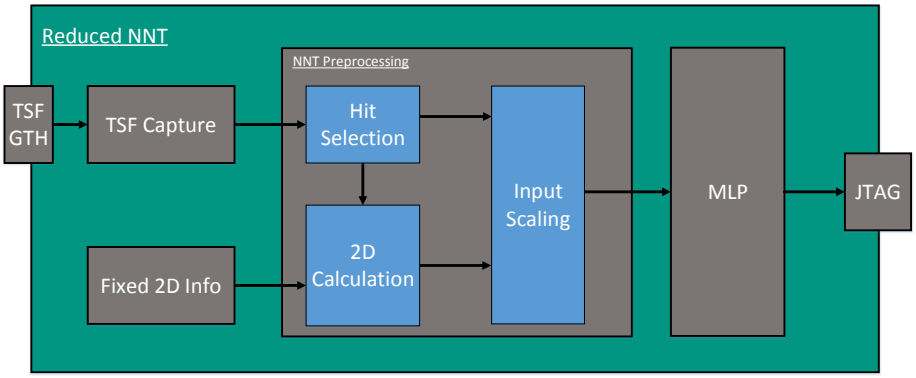


Figure 5.27.: Architecture of the reduced NNT that was used for operation in early cosmic ray testing.

The NNT used in this setup is quite different from the intended final version. Since the NNT is dependent on the availability of several data sources such as TSF, ETF, and 2DS, integration of the full setup has to wait until all of these components are ready to send data. Since all mentioned components are developed in parallel to the NNT, a simplified version was designed which is able to demonstrate correct basic functionality on the basis of a simplified CDCTRG. This version is using solely TSF data in combination with cosmic rays. In addition, it is only capable of providing a significantly reduced coverage of the

CDC's space by restricting the accepted data patterns. While these restrictions are not representative of the final system, they are reflecting the prevailing status of the CDCTRG in the early development stages of the experiment. In these, only a small part of the CDC had cabling connections usable for transmission of detector data. Additionally, only TSF modules connected to axial SLs were available. As a result, the setup is designed to be only dependent on the availability of these TSFs providing simplified functionality. Another important difference to the final setup is the unavailability of the 2DS which is designed to be sending matched TS. As an alternative to this, axial TSs are received directly from the respective TSF.

The first simplification is that the 2DS is compensated by emulating the incoming 2D tracks online at the NNT. This is done by assuming that its two track parameters phi and omega are most likely going to be highly restricted to few values viable for the restricted detector coverage that is present at this stage of detector operation. This is of course not valid for the real experiment but is a reasonable assumption fo the special restrictions present during early cosmic ray tests. First, there is no magnetic field during these tests. As a result, particle tracks passing through the CDC are not deflected from their path. As a consequence, the amount of tracks that would be found by 2DS is significantly reduced to just a few constant parameter pairs. The respective pairs of parameters can hereby be determined by a rough analysis of the received TSs. As only a small part of the detector is covered by TSF modules, the possibilities for different occurring particle tracks can be further restricted. In parallel, this is also reducing the set of TS that can be received. However, despite these simplifications, there are still enough tracks that are fulfilling all of the criteria in such that testing can commence. Besides the 2DS there was a working implementation of both ETF and aTSF throughout these tests. This was addressed by not considering stereo TS and compensating the ETF with the usage of the event time estimation as presented in section 5.2.4.3.

Figure 5.28.: Photograph of the integrated reduced NNT setup in the E-Hut at the experiment's facility.

In addition to revising the concepts of handling all of the input sources, the data sink for the NNT was similarly not available at this stage. Neither B2L nor GDL were available at this stage, which represent the main interfaces for analysing data of the NNT. The only remaining interfaces are therefore a JTAG [47] connection directly to the FPGA or communication over the VME backplane within the crate. Using JTAG is hereby easier to establish as no special software is required to be additionally developed for its usage. This connection can be comfortably used with the help of the Chipscope infrastructure for debugging FPGAs that is provided by Xilinx. Xilinx is providing a specific debugging software, the analyzer, and dedicated IP-Cores called Integrated Logic Analyzer (ILA) [120]. This core allows capturing arbitrary internal signals within the FPGA that are part of the same clock domain as the ILA. The JTAG interface of the UT3 is then connected to a PC installed at the E-Hut so that the NNT can be monitored remotely outside of the experiment's facility. A photo of the integrated UT3 used for testing is shown in figure 5.28.

The two major changes compared to the final setup, as well as the described limited geometry of the experiment, were taken into account in the MLP. This MLP was trained to be as efficient as possible for exactly these conditions. With the help of these changes, it was possible to take data at an early stage and perform initial tests independently of both 2DS and ETF.

As this setup's primary focus is not on being used in actual trigger operation, but rather to provide early benchmarks for operation, both throughput and latency were not measured. The possibilities to measure these characteristics are additionally restricted due to the unavailability of the necessary interfaces. Meanwhile, the characteristics of this setup in terms of clock frequency and resource consumption are listed in 5.21. Using this setup it was possible to achieve general verification of correct data reception and single events. Sample events that were evaluated are not explicitly presented in this thesis as they are superseded by the following operational setup.

Slices	Registers	DSPs	Frequency	Latency
13%	4%	91%	127 MHz	18 system cycles / 141 ns

Table 5.21.: Implementation characteristics for the reduced setup of the NNT.

5.4.2.2. Setup for Operation in the Complete Trigger System

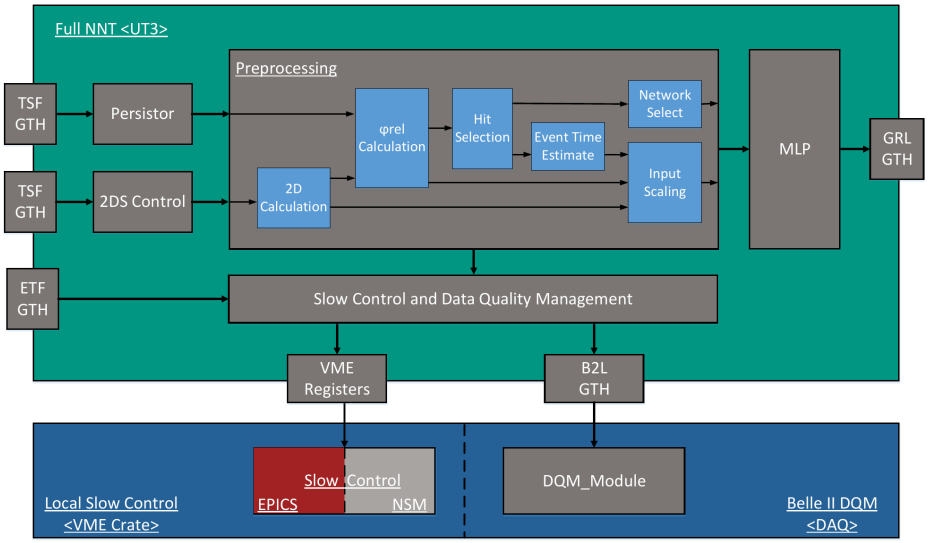


Figure 5.29.: Architecture of the final NNT for operation with beam injection. Included are the service interfaces for SC and DQM. It represents the culmination of all previous developments.

The NNT setup underwent many development iterations between the first tests with cosmic rays and its deployment during the first experiments with higher luminosity. However, these iterations were only incremental updates on the road to the final system that was used during the main part of operation. The architecture of this final system setup is shown in figure 5.29 and will be discussed in detail in the following.

Configuration of the Complete Setup

The setup of the NNT that was designed for operation with high luminosity and integrated into the complete CDCTRG provides all of the required functionality necessary to estimate the z-Vertex. The main difference of this setup compared to the conceptual design is hereby at the usage of the event time. In the initial design, it was intended to use the ETF in order to have the most precise estimation of the event time. The operational setup, however, is only receiving and recording the ETF's data for subsequent analysis. Instead of using it, the fastest priority time calculated and used as an estimation of the event time. The reason for this is that the ETF is still in a prototypical development state at the beginning of operation with collisions. However, recording the data allows the investigation of its behaviour. In addition, the connection to ETF is providing another

input channel that can be used to investigate whether the general data flow is operating correctly independent of the main data sources.

With respect to the internal architecture, the preprocessing is configured to the reception of 10 TS, whereby the persistence time window is set to 16 clock cycles. In the first analyses, this window width proved to be the best compromise between taking correct and incorrect TS into account. A configuration capable of using the full 15 TS is meanwhile ready to be deployed, but not used since recorded data did not show sufficient indications for improvement to justify the overhead for resources. The neural network is meanwhile configured to process a maximum of one track per clock cycle. This means that no parallel processing of different tracks is supported. Since the rates with which tracks are arriving are still rather low during early operation, this configuration is already sufficient for usage. The chosen configurations for both the neural network and the preprocessing are summarized in table 5.22 and 5.23.

Since this setup is the basis for the first functionally complete operation of the NNT, the presence of DQM is essential in order to prove correctness. Here all three input channels are recorded. In addition, all internally calculated and used singles from the preprocessing and the estimation of the z-Vertex are written out for subsequent verification. In addition to this, each track estimation is sent to the GRL. In order to allow easier identification of the track, the estimation of the z-Vertex is sent together with the corresponding 2D-Track parameters.

Using this setup of the NNT data generated online during the experiment was recorded. This is shown in figure 5.30 [49] for an early run of operation with cosmic rays. It shows the distribution of estimated z-Vertices for a selected set of suitable tracks. The observed distribution from the NNT is hereby agreeing with the expected distribution for the configuration used at this experiment, in which the majority of the observed tracks are dominated from the $z>0$ region due to the Trigger System of the ECL Detector (ECLTRG) providing the main trigger signals. At this point of development, the DQM was not yet ready to provide a more sophisticated analysis for verification. Such analysis will be shown in the following.

TS Depth	Persistence	alpha BW	phi_rel BW	Hit Selection	Multiplication
24	16	14	24	1 stage	Slices

Table 5.22.: Architecture configuration of the preprocessing for physics operation.

Weight Sets	Weight BW	Input BW	MAC	Activation	MUX
5	18	13	DSP	LUT full optimized	4

Table 5.23.: Architecture configuration of the MLP for physics operation.

The properties of the implementation are hereby based on the smaller UT3, the device XC6VHX380T, and are presented in table 5.24. The clock frequency is hereby set to 127 MHz, while the resource utilization for slices and DSPs is at about 50%. Although the

resources are not heavily utilized, timing closure for this setup is only achieved with significant additional effort due to the routing congestion induced by parallel processing of the input values. This usually required several iterations using design space exploration directed by SmartXplorer with the main strategy being set to optimize routing congestion. A firmware without timing violations is then typically achieved within 2-3 days on the available server infrastructure. In many cases, the required timing was achieved for the overall processing within the architecture, however, timing problems arose within the supporting modules such as the VME controller, especially for write access to the FPGA. However, this did not influence operation with the generated firmware.

Slices	Registers	DSPs	BRAM	Frequency	Latency
46%	14%	53%	49 %	127 MHz	9 data cycles / 288*ns*

Table 5.24.: Implementation characteristics for the full setup of the NNT.

Validation and Evaluation of Hardware-based z-Vertex Estimation

The validation and evaluation are presented separately for the two main operational configurations of the experiment, which are cosmic rays and physics collisions. As operation with cosmics was the first to be available its results will be presented first. The accumulated distributions from runs during experiment 6 are shown in figure 5.30. These were generated solely based on data sent by the four NNT boards and thus represent the hardware's capabilities. General proof of correctness can be seen by analysing the z-distribution. It includes two peaks, one around z=0 which is mainly produced by particle tracks generated by collisions. The second peak is meanwhile more strongly pronounced and present at around $z = +50$. This peak is expected for this configuration of the trigger system, as it is based on triggering in the presence of ECLTRG signals. The ECLTRG is meanwhile highly dominated by tracks from $+z$ during this setup, which leads to the observed distribution [49]. The plots mainly show two important aspects, first, that data can be correctly received within the expected ranges and that the derived distributions for both z and theta match with physics to be expected from the experiment's configuration during these runs. This is however only a light indicator of the correctness of both the hardware implementation and the entire concept, as it has to be compared in more detail with offline processing. However DQM was not fully available at this stage, in particular, it was not possible to match the tracks observed by the hardware with the tracks recorded and available at the offline storage. This was resolved in the later stages during collisions at experiment 8 and shown in the following.

After discussing operation during cosmics the focus is now put on operation with particle collisions. This mode was available during the later experiments and runs at which much more information was available about, for example, the latency of the entire CDCTRG and the coverage of the CDC. As this is necessary to show the mandatory functionality in order to be operational, this will be discussed first, before moving towards more qualitative measurements such as the estimation of the z-Vertex.

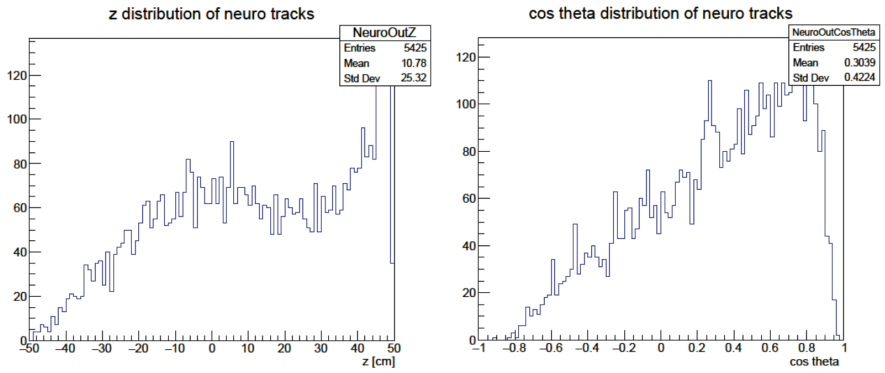


Figure 5.30.: Distributions for both z and theta that were read out via B2L from NNT hardware. Data was taken during cosmic runs as part of experiment 6 [49].

After discussing operation during cosmics the focus is now put on operation with particle collisions. This mode was available during the later experiments and runs at which much more information was available about, for example, the latency of the entire CDCTRG and the coverage of the CDC. As this is necessary to show the mandatory functionality in order to be operational, this will be discussed first, before moving towards more qualitative measurements such as the estimation of the z-Vertex.

At first, the focus is put on the receiving side of the NNT. Most indicative of correct integration is the reception of 2D-Tracks, as these are the main controlling element for internal processing. The distribution of received 2D-tracks at the NNT is shown in figure 5.31 with respect to the recorded angular distribution of phi. It shows the accumulated received tracks across all quadrants, with more detailed distributions for every single quadrant being shown in figure A.3. Most importantly it shows that the combination of all NNT boards is able to cover the entire CDC. Peaks are hereby observable and mostly present at the borders of each quadrant which can be explained by the overlapping of quadrants.

Besides the reception of 2D-tracks, the NNT is receiving unmodified stereo TS from the respective sTSF. Their distribution under the same conditions as for the 2D-tracks is shown in figure 5.32. The distribution hereby contains both the axial and stereo TSs. Again, it shows that the NNT is receiving all data and achieves complete coverage across the CDC as every ID is received. The distribution shows an irregular structure which is due to the difference in receiving axial and stereo TS. While stereo TSs are received directly and only once, axial TSs are sent together with their related 2D-track. The same TSs can meanwhile be sent multiple times by the 2DS in case an already found track is updated. This behaviour leads to the observed irregularity, in which IDs related to the axial SLs are received in much higher numbers. In addition, it can be observed that there are some TS ranges with high inefficiency since they are rarely observed, which might indicate problems at the respective FEE or susceptibility to background events.

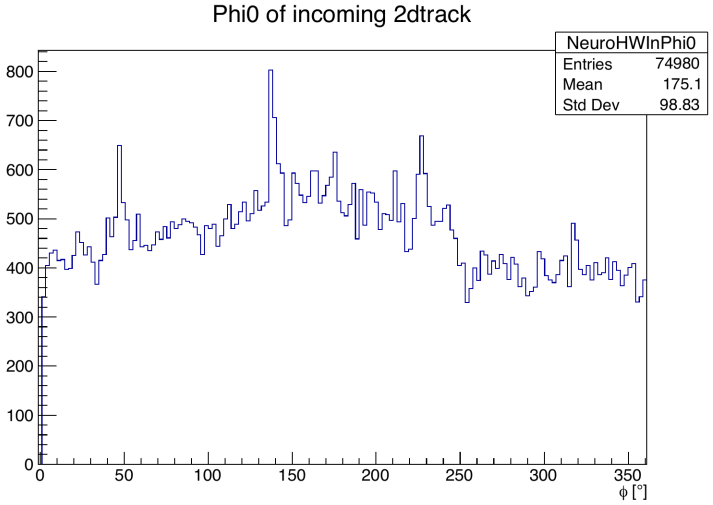


Figure 5.31.: Distribution of received phi at the NNT during run 1703 of experiment 8. It represents the accumulation of all four installed FPGA platforms.

One of the most critical requirements of the NNT is to stay within the time budget of the L1 trigger system. Its latency was previously only evaluated with respect to the clock cycles allocated to the internal processing. However, in the end, it all comes down to the latency measured during operation including all transmission delays and uncertainties to determine whether the NNT is fulfilling the requirements and can thus be used. To determine this latency, all trigger signals were measured at the GRL [64]. These measurements are shown in figure 5.33. The presented graph is showing the latencies for all sub-triggers. Components of the CDCTRG have a finer description as it is consisting out of multiple systems. Most importantly here is, of course, the latency of the NNT, which is shown by a dotted red line. The shown latency is hereby represented as the time of arrival at the GRL relative to the deadline for this event to be triggered in time. This deadline is represented with a solid black line at around -500 ns. It can be seen that the NNT is right at the border of this deadline. Overall all of the trigger signals generated by the NNT are arriving well within the latency budget, thus fulfilling the goals. There is an observable uncertainty for the time of arrival, here it has to be noted that this uncertainty is already present for the arriving 2D-tracks which are in turn controlling the start of processing at the NNT, thus accumulating at this stage.

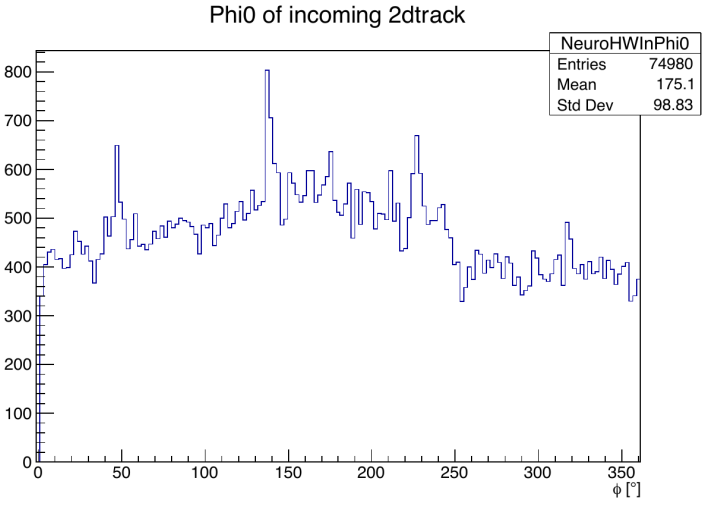


Figure 5.32.: Distribution of received TSs at the NNT during run 1703 of experiment 8. It represents the accumulation of all four installed FPGA platforms.

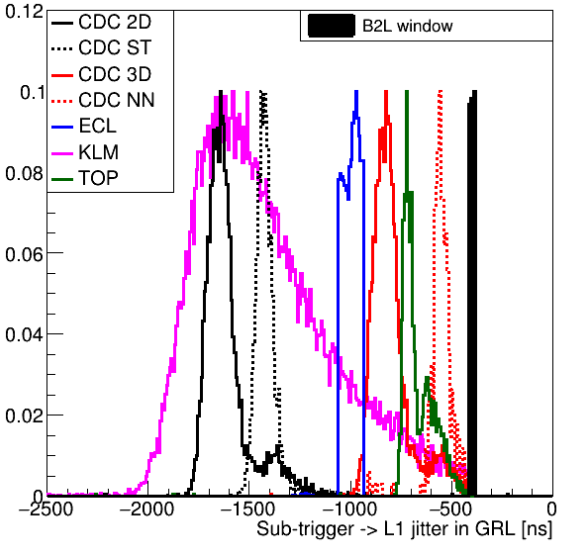


Figure 5.33.: Distributions of measured latencies for all sub-triggers. They are all plotted relative to the GRL at which they were measured. The shown latencies were measured during experiments 7 and 8 [64].

With basic operational functionality already shown to be fulfilled, the focus is now put on the estimations of the z-Vertex. Using the same experiment and run as before the values read out from hardware are shown as distributions in figure 5.35 for the estimation of the z-Vertex. In general, it shows a healthy distribution as was already observed during cosmic ray operation, however, this time a much more pronounced peak at z=0 is observable. This more pronounced peak is the result of the collisions that are now present in this configuration. As before it is again observable that most tracks observed in the detector are originating from +z. This is indicating a high occurrence of background events in the experiment and underlining the importance of the NNT as it will be able to suppress these events. To show the correctness of the hardware results, a distribution of estimated z-Vertices is shown in figure 5.34, however, this time not read out from the hardware, but generated with a simulated NNT using the same data set from this run but processed by an SW model. The distributions are generally very close to each other, with some slight differences. Additionally, the simulated NNT is generating more outputs then the real hardware. This is mostly due to the limited time window that is available to send a result for the hardware that can still be recorded within the B2L time window. Late 2D-tracks are not arriving in time to generate an output within the B2L time windows.

To have a better comparison of hardware and simulated NNT, two additional plots were generated. Figure 5.36 is showing the difference in the estimation of the z-Vertex between both options for the same track. The heavily dominant peak here is at zero, representing complete agreement between simulation and hardware as there is no difference between both versions estimations. Areas close to the peak are mostly due to the usage of fixed-point operation at the hardware when compared to the SW model. However, there are still larger deviations observable in the distribution. Before getting into the reasons for these, however, an even more detailed examination is possible by studying the scatter plot shown in figure 5.37. This plot is useful to show the agreement of both options within the respective areas of the experiment along the z-Axis. It shows that agreement is present across the entire z-Axis as both versions are generating very similar results throughout the entire -50:50 cm range.

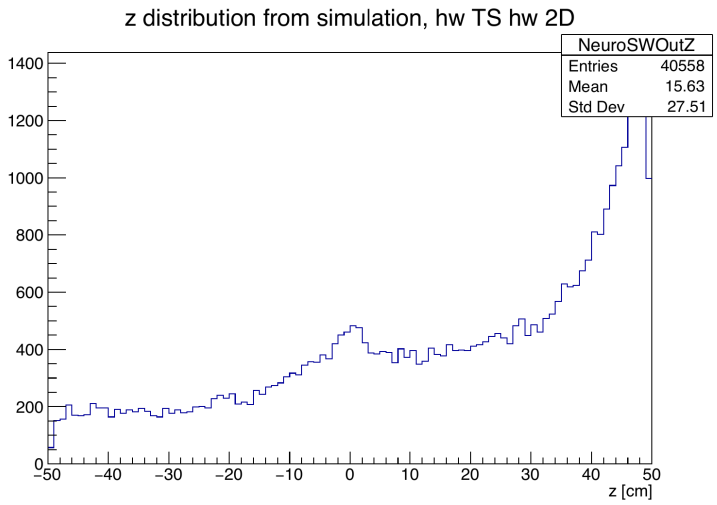


Figure 5.34.: Distribution of the estimated z-Vertex generated by software emulation of the NNT using data received from the input data sources during run 1703.

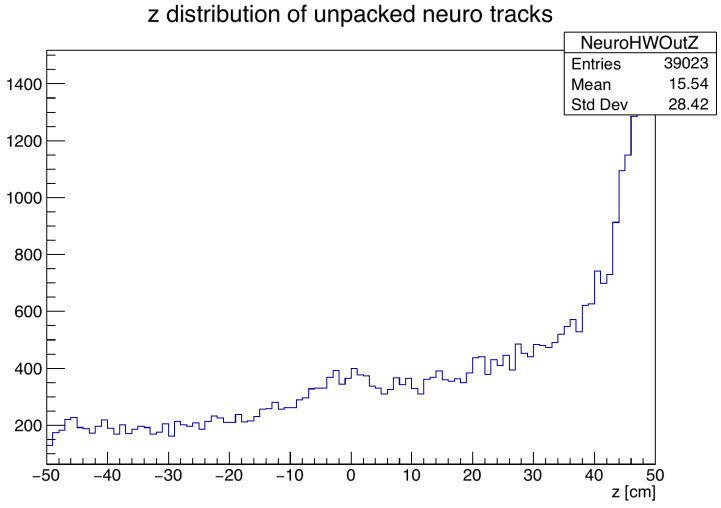


Figure 5.35.: Distribution of the estimated z-Vertex read out from the hardware during run 1703.

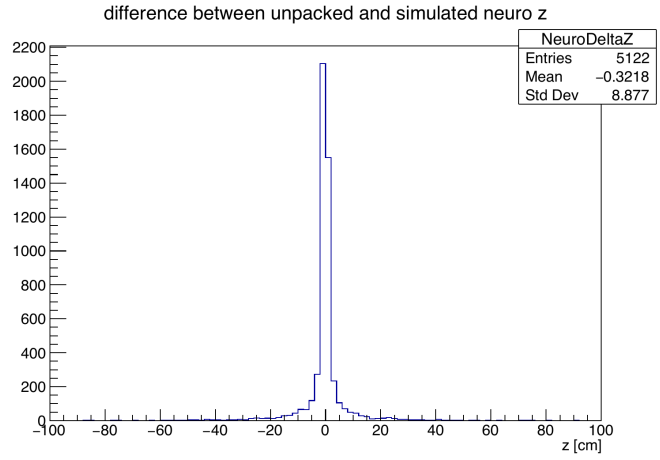


Figure 5.36.: Distribution of difference between software reference and hardware z-Vertex estimation that was read out from the NNT hardware during run 1703.

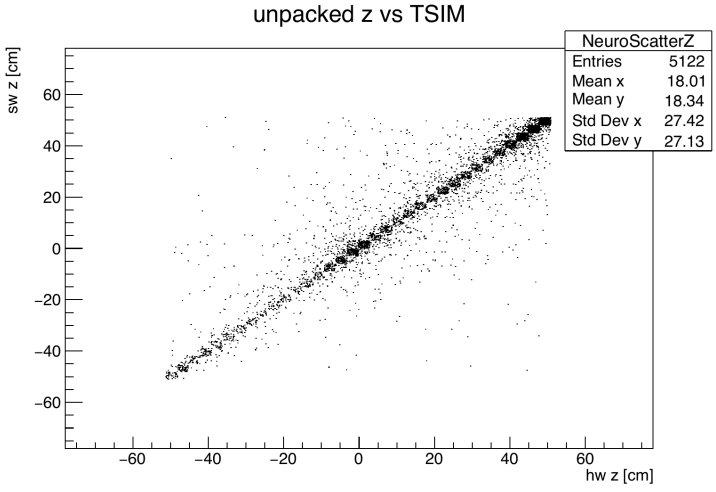


Figure 5.37.: Scatter plot of the z-Vertex showing the relationship between software and hardware implementation during run 1703 of Belle II operation.

Coming back to the reasons for the observed deviations, there are mainly two problematic aspects to be considered besides the general precision problems when using fixed point processing. The bright side here is that both problems are a result of problems in the synchronization process between the hardware's B2L readout and the implementation of the offline DQM and not due to operational or functional problems. Smaller deviations are due to the difference in the selection of TSs. This is however not the result of the algorithmic implementation of the HitSelection module, but due to the way, the persistence of stereo TS is implemented in both methods. While the hardware is operated using the persistors as described in section 5.2.3.2, DQM is not considering any time windows but is rather selecting from all occurring TS. As a result, some of the TS selected by DQM are considered invalid by the persistor. This leads to different input values at the MLP for the respective SL. The other source and the reason for large deviations is mismatching between the tracks. In this case, two separate tracks are present within the same event and time window of the B2L readout. In some cases, the DQM module mistook results read out from hardware as belonging to another track. Both of the mentioned issues are resolved as of experiment 10 run 5825 for which both versions are nearly completely agreeing, with the main source of disagreement being the fixed point operation. The new results based on revised DQM are shown in figure 5.38 and 5.39, again in the form of a z-distribution comparing HW, SW model and reconstruction, the estimation range was hereby set to -100:100 cm. An accompanying distribution for the theta estimation is shown in figure 5.41. However, DQM was not ready for comparison, as such no qualitative statement can be made at this stage of operation.

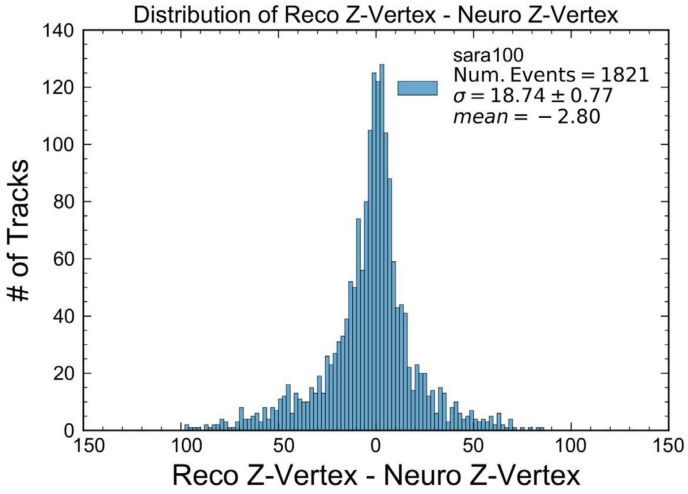


Figure 5.38.: Distribution of the difference between estimated z-Vertex read out from the NNT hardware and the estimations from the reconstruction using data from for experiment 10 [50].

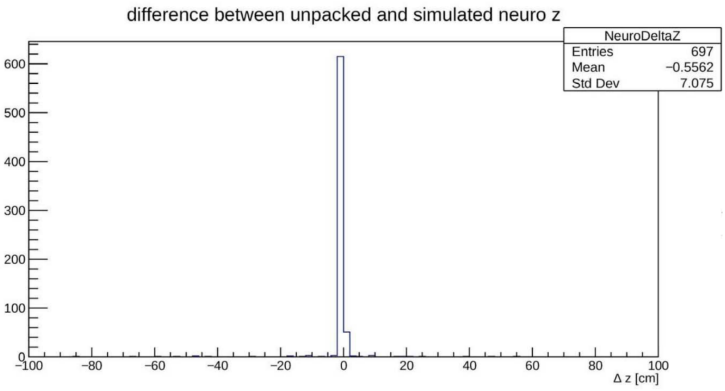


Figure 5.39.: Distribution of difference between estimated z-Vertex read out from the NNT hardware and the software model for experiment 10 [50].

The results are very promising showing nearly complete agreement between the reference implementation represented by a simulated software-based NNT and results read out online during operation from the hardware. In addition to this, results were compared to precise track reconstruction. Since only those events that are originating from z=0 are stored, it is not of value to plot its z-distribution, as it is a narrow peak at zero. More interesting is the difference between the hardware-based NNT and the reconstruction which is presented in figure 5.40. Again for most events, an agreement in the estimation can be observed, however, the deviations are much more pronounced here since the uncertainties of the concept itself are now added to the hardware implementation. The results are showing that cuts for the estimated z-Vertex should be set quite generously at this stage in order to not lose too much valuable data. However considering that the main peak is located at around z=50cm, a significant chunk of the background events can be suppressed by setting the suppression cut to around z=40cm without losing too much valuable data since the deviations beyond those values are less frequently occurring.

Further insight into the internal processing of the hardware NNT is gained by analysing the generated input signals for the MLP as well as the intermediate signals generated at the selection of TS. The data that was read out is presented as the difference between simulated and hardware NNT similar to before. The different types of inputs are discussed in the following, as they cover parallel processing streams that are partially independent.

The distribution of phi_rel is shown in figure 5.42. It is calculated by using both the 2D-track and stereo TS information. This value is thus influenced by all of the used input data sources of the CDCTRG. The plot shows a clear peak located at around zero, which indicates nearly complete agreement and correctness of the hardware implementation. As it is the most computationally intensive input value, it is affected the most by precision aspects of using fixed point computation. This value is additionally dependent on the selection of matching TS. The results of the selection are hereby shown in figure 5.43. Small deviations are observable here, which can be attributed to the differences in how simu-

lated and hardware NNT are implementing the time window of validity for TS. The next input under investigation is alpha, which is plotted in figure 5.44. This input is solely dependent on the estimated 2D-track parameters. Again agreement can be observed for the most part, however there are deviations. The reason for these deviations are mismatches in relating tracks hardware and simulated tracks in the DQM as described before. Since the input is only dependent on the current track parameters, deviations can only occur when different parameters are assumed in the comparison. These parameters are meanwhile directly taken from the incoming data stream from 2DS. Since these are matched with the DQM of the 2DS, different chosen values can only be appearing due to several 2D-tracks being present in the same event time window, the resulting freedom of choice led to the mismatch. The best matching is achieved for the estimation of the event time, which is shown in figure 5.45. It is basically always correct independent of any synchronization problems between hardware and DQM.

Summary

This section discussed the final setup of the NNT as used in higher luminosity operation of Belle II. It is fully integrated and can be operated within the resource budget of the used UT3. Additionally, this setup is capable of generating and sending its trigger signals in time towards its finals destination the GRL to be considered for the final readout decision. By fulfilling these requirements the NNT is in principle ready to be deployed. Accompanying the fulfilment of these requirements, it was shown that the chosen approach to integration is able to cover the entire space of the CDC detector as each SL is received correctly. While these only show that the NNT is capable to fulfil all basic requirements, qualitative statements about its capability to correctly estimate 3D-track parameters are made. In general, the integrated NNT is able to reproduce the expected physics distribution for the z-Vertex during online operation for both collisions and cosmic rays. It is additionally nearly completely matching the results generated by emulating the algorithms of the NNT offline in SW using the same detector data. The observed differences are hereby either the product of fixed point processing or the current status of synchronization with the DQM. There is no indication of internal problems or systematic errors within the hardware implementation. All of the internally generated values are either agreeing completely with the reference simulation or have slight deviations that can be explained by the mentioned effects. While the comparison with precise track reconstruction shows that the resolution is currently not hitting the targeted 6cm, it has to be noted that it still provides good enough performance to be used for suppression of background events for the current state of the experiment. Additionally, these results were generated with neural networks trained with simulated data. Training of networks with experiment data is under way and preliminary studies show much improved resolutions.

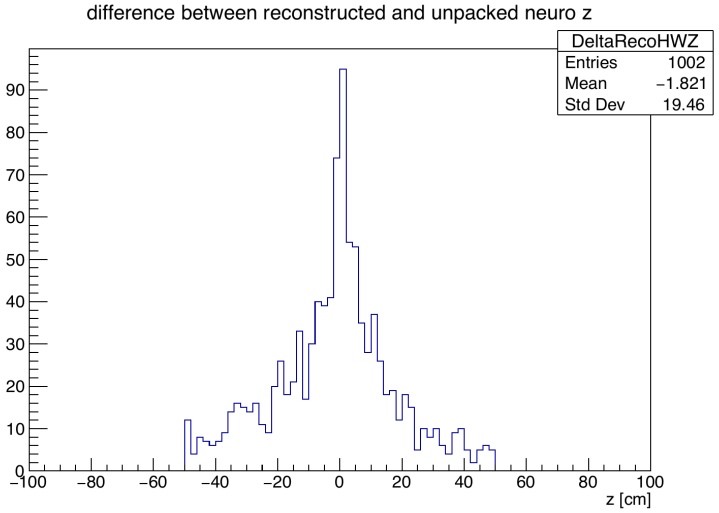


Figure 5.40.: Plotted distribution between the NNT hardware's z-Vertex estimation and the estimation of the reconstruction using the data received during run 1703 of Belle II operation.

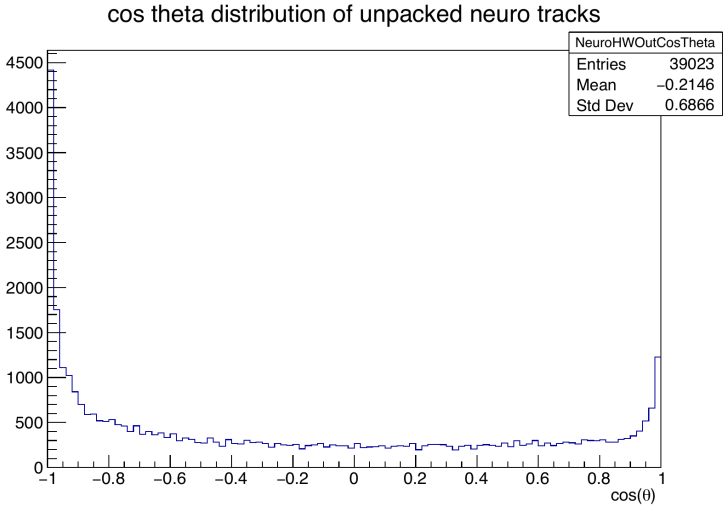


Figure 5.41.: Plotted distribution of the NNT hardware's theta-estimation using the data received during run 1703 of Belle II operation.

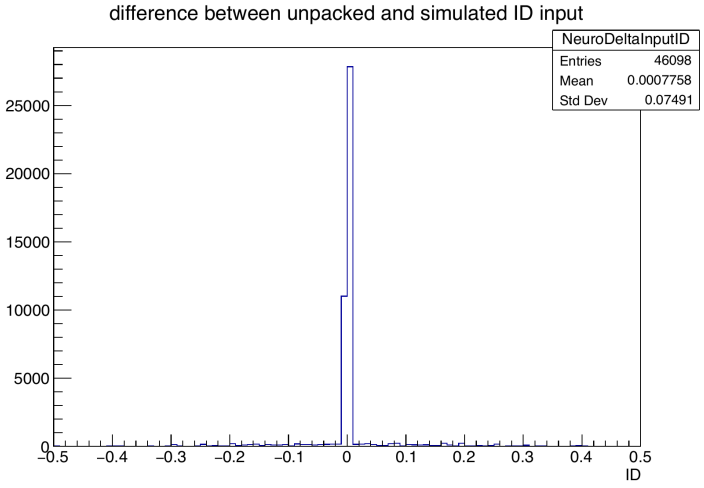


Figure 5.42.: Distribution of the difference between the input variable phi_rel calculated by the reference software and read out from hardware. The variable is used as input for the MLP. It was read out from the NNT hardware during run 1703 of Belle II Operation.

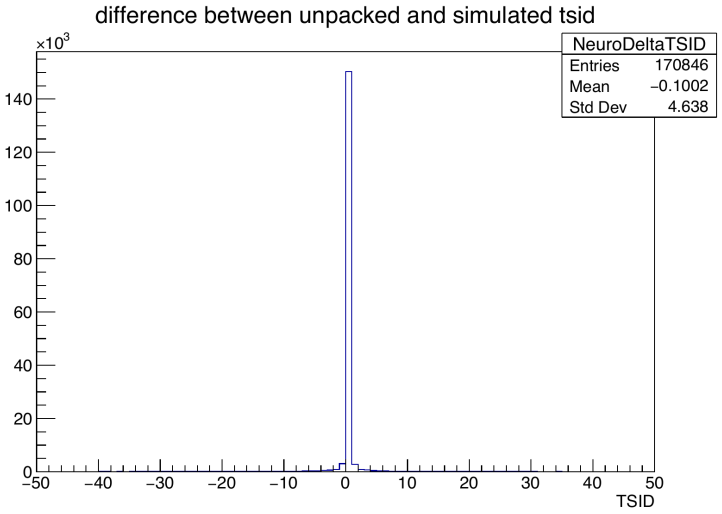


Figure 5.43.: Distribution of the difference between hit selection calculated by the reference software and read out from hardware. It was read out from the NNT hardware during run 1703 of Belle II Operation.

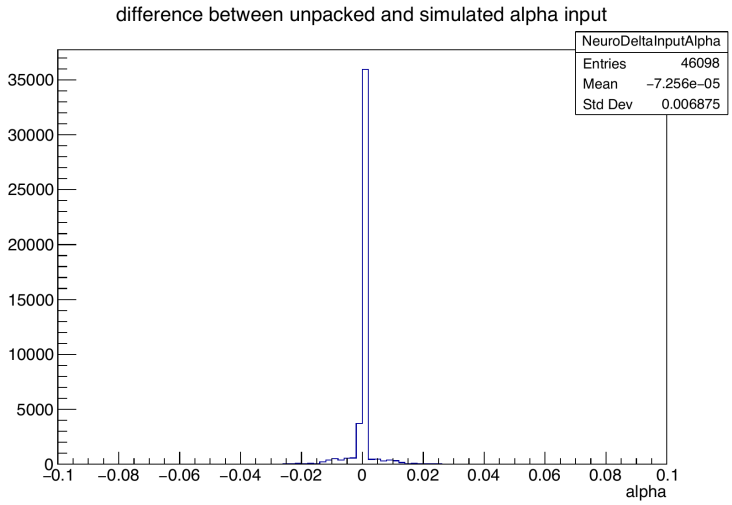


Figure 5.44.: Distribution of the difference between the input variable alpha calculated by the reference software and read out from hardware. The variable is used as input for the MLP. It was read out from the NNT hardware during run 1703 of Belle II Operation.

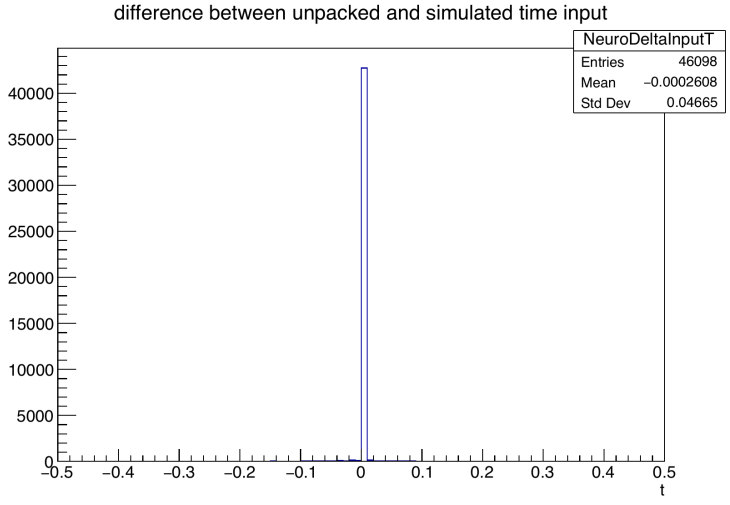


Figure 5.45.: Distribution of the difference between the input variable drift time calculated by the reference software and read out from hardware. The variable is used as input for the MLP. It was read out from the NNT hardware during run 1703 of Belle II Operation.

5.4.3. Local Setup for Testing and Demonstration

One of the disadvantages of the UT3 is that it is only available on site at the KEK. In order to test and evaluate the functionality on an FPGA platform outside of Japan, a local prototyping system was developed. One of the most important aspects of this is a representative emulation of the CDCTRG. To achieve this the local prototyping platform is divided into two parts, one that emulates the CDCTRG with its data flow and the NNT itself. The VC709 development platform is used as a host since it is widely available and provides a sufficient amount of GTH interfaces in order to replicate the CDCTRG's data transmission. The presented approach to a local setup for testing is hereby based on the bachelor thesis Ref. [Rin18].

The system architecture of this local setup is shown in figure 5.46. The TX board emulated all the input components ETF, TSF, and 2DS. Their behaviour is recreated by sending trigger data that is pre-recorded and stored locally on the FPGA in BRAM. This data derived from either the simulation or recorded from the experiment. This way it is possible to recreate special situations of trigger operation, that caused unexpected behaviour of the NNT. The data is sent via four GTH ports to the RX board, which contains the prototype implementation of the NNT that is currently under test.

In order to set up the data transmission, load the test data and also control the complete operation, a Picoblaze [121] processing system was instantiated on both boards. In addition to the provided features for easier operation, this is especially helpful for the configuration of the optical communication and their clocking resources, since it is much easier to solve these aspects in software. For this purpose, a dedicated clocking IC on the board is configured to provide the correct clock frequencies for the optical transceivers. This IC is configured for operation in the assembler instructions, which are stored at the Picoblaze.

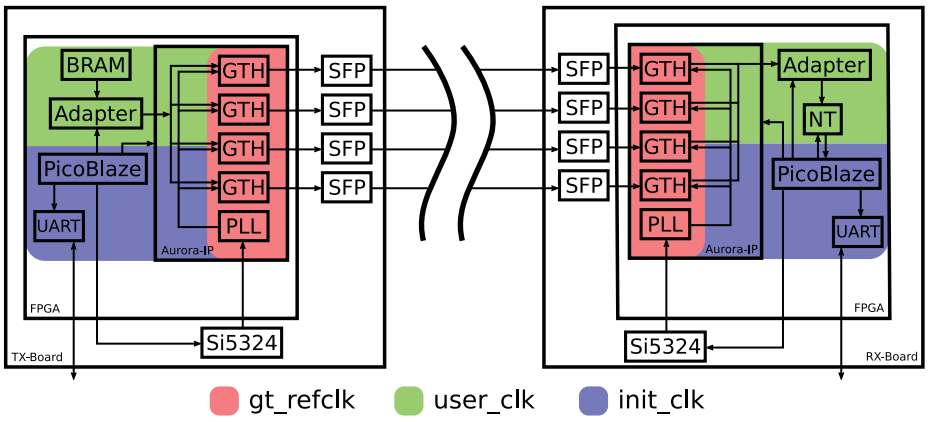


Figure 5.46.: System architecture of the local NNT setup for testing and prototyping [Rin18].

5.4.4. Investigation of Alternative Platforms and new Technologies

For the first collisions and tests at the experiment, a suitable platform based on the Virtex-6 FPGA was selected for the NNT. The main reason for this was the connectivity capabilities provided by the board, which met all the requirements for integration into the CDCTRG. Additionally, it was ensured that the boards were going to be available in time in order to be used at the early stages of operation. For the further operation of the experiment, however, it is possible to switch to alternative platforms. In the following, the prototypical developments for the newer UT4 are presented. The final version of the board was however not available for usage during the creation of this thesis, thus the results are still showing prototypical features. In addition, further possible choices are presented and discussed in the following.

Transition to the UT4

The current version of the NNT that was designed on the basis of the UT3 has in principle only one significant limitation, namely the throughput. It is only capable of processing one track at each system clock cycle, dropping alternative tracks provided by the 2DS. This is mostly due to the throughput and the resource consumption of the MLP.

Its succeeding platform, the UT4 is meanwhile based on newer FPGAs of the Ultrascale generation. Using this generation is opening up new possibilities for overcoming the throughput limitation. Due to the smaller feature size, it is possible to implement internal modules with a higher clock frequency as the signal propagation times are significantly reduced. In addition, it provides more LUTs that can be used for more logic or additional reduction of routing congestion. The UT4 is hereby supporting multiple FPGAs, the focus will hereby be put on the VU80 and VU125 [132]. These two options represent the early prototyping system and the production system for the final UT4.

The architecture presented in section 5.2.5 was adapted to the UT4 to investigate the possibilities of this platform. The investigation is hereby mostly focused on the achievable operating clock frequencies and the degree parallelism that are impacting the throughput. The achieved results are shown in table 5.25 for the VU80 option of the UT4. Three architecture configurations are hereby shown. The base configuration with one network, an extension with two networks either using only DSPs or as a heterogeneous configuration, denoted as $2NN\_H$. These results were derived by a design that only consists of the core processing logic excluding additional necessary infrastructure such as the B2L as there was no implementation available at the time this investigation was carried out.

Type	Slices	Registers	DSP	BRAM	Frequency	Latency
1NN	12%	3%	43%	1 %	250 MHz	9 data cycles / 288ns
2NN	23%	5%	86%	2 %	200 MHz	6 data cycles / 180ns
2NN_H	26%	6%	80%	2 %	150 MHz	9 data cycles / 288ns

Table 5.25.: Implementation characteristics for the full setup of the NNT based on the UT4.

The first result is that the basic architecture is easily within the resources provided by the platform. In addition, it is possible to achieve much higher operating frequencies throughout the entire processing architecture. A frequency of 200 MHz is possible and was used for testing on a prototype board. The most significant improvement is meanwhile that two neural networks can be implemented for parallel processing. Using this parallel configuration is reducing the frequency below the 200 MHz threshold on the VU80 however it still allows increasing the throughput compared to operation on the UT3.

The instantiation of two networks, however, is still inducing high resource utilization for the DSPs. They still fit into the available budget and are still achieving a sufficiently good timing for implementation. With the addition of the service infrastructure required for integration into the CDCTRG, however, achieving timing closure might become problematic. To address this possible bottleneck, the heterogeneous implementation of the MLP as introduced in section 4.4.4 is examined. Since the difference in achievable operating clock frequencies is even larger on the UT4, separate clock domains for Slice- and DSP-Neurons are used. Using this approach the utilization for the DSPs can be significantly reduced, whereby with a frequency of 150 MHz being reached.

Summarizing the results, it is possible to transport the NNT to the UT4 using two parallel neural networks and a higher clock frequency, which greatly improves the throughput, primarily due to the shorter signal propagation times on the UT4. The investigation of how to make use of the UT4's increased capabilities in this context is intensified in section 6.3.1, in which a unified integration with a more advances estimation of the supporting track parameters that serve the inputs of the NNT are investigated.

Alternative off-Chip Memory Solutions for Deterministic Reloading of Networks

In section 5.2.5.1 it was investigated to what extent it is possible for the NNT to use external memory to extend the number of neural networks that can be loaded dynamically. Using on-chip memory only the targeted UT3 platform is theoretically capable of supporting up to 16 networks. When using external memory in addition to this, the latency for loading all weights is set at around 500 ns. This exceeded the overall latency budget of the NNT and therefore cannot be used for the initial configuration of the CDCTRG. The external memory used within this considerations was based Double Data Rate (DDR) technology, which is the most popular choice right now. However, nowadays new technologies are emerging. Modern architectures, especially in the area of dedicated processing for ML applications, rely on alternative memory structures typically based on 3D manufacturing technology such as the High Bandwidth Memory (HBM) [45] or its counterpart Hybrid Memory Cube (HMC) [74]. Since the NNT is going to be operational long-term with a life-time spanning around 10 years, it is going to undergo several update cycles. It is thus worth investigating the promising forecasts for these novel memory technologies. For this, the usage of HMC was investigated and prototyped. The platform used for this investigation is an AC-510 module provided by Micron [74]. This provides access to 4 GB of HMC based storage that can be read out with 30 GB/s throughput. The module is equipped with a Kintex-7 FPGA that is handling all internal communication with the memory. It additionally includes GT connections for communication with the hosting system. The provided module is used together with a PCIe adaptor card, that allows easy integration into a normal computer. Before presenting the details of the evaluation, it is to

be noted that HBM is considered as well, however it was not available for easy prototyping at this stage. Using the described setup of the NNT it took around 950 ns to load the weights of a neural network with the packet size being set to the maximum value of 128 Byte. While this is not an encouraging result, it is to be noted that the network is stored in raw form without any optimization of the constant weights, which will bring latency further down. As this is unlikely to reach the latency goal, it was not further explored. The idea of using external memory is however not completely off limits, since unified integration of NNT with the S3D is expected to open up a significant additional latency budget.

5.5. Summary

This chapter covered all aspects of the neural z-Vertex Trigger that is tasked with estimating the z-Vertex of particle tracks using the CDC of the Belle II detector. All aspects relevant to the FPGA-based design and integration of this system into the overall trigger system of the experiment were discussed here. As with all trigger components the NNT has to fulfil strict requirements regarding latency and throughput coupled with a high demand for IO resources in order to receive the data necessary for the estimation. These requirements led to the investigation of selecting a suitable platform to host the system. The most suitable platform is represented by the UT3 and 4, which are custom FPGA-boards designed for the Belle II trigger system. While they are not equipped with the most modern generation of FPGAs, they are fulfilling the requirements for IOs and are equipped with an IP core library maintained by the experiment's collaboration. Due to the strict time schedule, it represented the best choice to allow early operation and evaluation of the NNT in the experiment. The considered aspects for integration were explored beyond the selection of a suitable platform. This includes an integration strategy that allows the complete coverage of the CDC without exceeding the number of available UT3s. In addition, every interface protocol to be supported was investigated and FPGA modules supporting these were developed. The most complex one is hereby represented by the module handling the TSF as it cannot abstract from the CDC and is thus influenced by drift times. To compensate for this, a persistor module was developed and discussed. This module was designed in a flexible way since it has to consider several potentially changing parameters.

This chapter also featured the presentation of the entire processing architecture that was developed to realize the NNT. The presented solution for the implementation of the neural network is hereby based on the concepts introduced in section 4.4 with mostly using the low-latency options discussed there. Preprocessing is meanwhile consisting of custom algorithms that are taking both the geometry and characteristics of the CDC into account. The presentation of the preprocessing architecture is based on the architectures common in Xilinx's FPGAs. Each processing stage hereby followed a flexible design by introducing variable parameters to be defined during design time as well as the possibilities for parallel execution. Evaluating the entire resulting architecture the bottleneck bounding the performance is located at the implementation of the neural network, while the preprocessing is fulfilling all latency and throughput requirements without requiring excessive amounts of resources. The present bottleneck is mostly a result of the limited

availability of DSPs on the considered FPGAs. While the latency requirement is being satisfied by every developed architecture, throughput is always below the maximum rate of tracks to be received during high luminosity operation when using a UT3. This is however improved by using the UT4 which is capable of processing up to two tracks every system clock cycle. As these throughputs are only required in later stages of the experiment, the presented setups of the NNT are already sufficient for the current status using the low-throughput version on the UT3.

The developed realization and architecture of the NNT is supported by several auxiliary tools and mechanisms. This includes support for the configuration of design time parameters for all internal processing that is performed by a semi-automated framework. This framework uses an abstract configuration file for the neural network, which is provided after the training phase. In addition, it includes mechanisms to easily reload different trained neural networks and validate the architecture by the generation of simulation scripts.

Since the NNT is assuming important responsibilities as it is used to decide whether to read out the detector, it needs to be monitored for correct behaviour during operation. These tasks are performed by the Slow Control and Data Quality Monitoring that were both discussed. Solutions for these mechanisms based on the UT3 were presented with DQM being realized by using the B2L and SL being covered by regular register accesses over the VME bus.

All of the aspects and components that were developed for the NNT are finally combined to form different setups that were integrated into the CDCTRG and active during the operation of the experiment. The most important one is hereby the operational setup that is discussed in section 5.4.2.2. This discussion includes a performance analysis using data taken during collision operation with higher luminosity during phase 3 of the experiment. The analysis hereby at first shows that the provided design used for the NNT is correctly receiving all required data achieving coverage of the entire CDC. In addition to this, an analysis of the arrival of trigger signals at the GRL was presented. This analysis shows that the output signals of the NNT are arriving within the latency budget set for the L1 trigger system. Besides the fulfilment of the operational requirements, an analysis of the quality of the estimation is provided for a selected collision run. The generated correlation plots and histograms show that this setup is already capable of providing the desired functionality when compared to ideal processing modelled in software. This setup is capable of estimating the z-Vertex with a resolution of around 40cm. As a result, the NNT can be used with suppressing cuts set to +/- 40cm. This is meanwhile achieved without the usage of a network specifically trained with experimental data but rather trained with data generated from the simulation of the experiment. It is expected that the resolution will significantly improve with newly trained networks, forming the outlook into the future operation of the NNT.

6. The Hough-based 3D Track Estimation

6.1. Upgraded Estimation of 3D-Track Parameters

The basis for track finding within the CDCTRG is represented by the detection and estimation of particle tracks passing through the detector in the two dimensions omega and phi. Estimating these parameters is the responsibility of the 2DS [98], which combines all active axial TS across the different layers of the CDC. By combining a subset of suitable TS, the 2DS estimates a particle track that is the matching them as close as possible. This track hereby has the smallest distance between the location of the track and the TS across the layers. This estimation represents the basic track trigger signals within the entire trigger system and is used at the GDL as part of the readout decision rule set. Besides its role in supplying the GDL, it is used for the subsequent estimation of additional track parameters such as both theta and the z-Vertex. Both of the systems dedicated to generating these parameters, the 3DS and NNT, are in principle combining the estimated 2D track parameters with additional stereo TS and the event time to derive desired parameters.

While the 2DS is performing as specified according to its concept, there is still room for improvement by using enhanced derivatives of the core algorithm. One such approach is the Hough-based 3D-Track Finding. This approach extends the original concept by adding stereo TS to the hough-based track finding. In addition to this, the generated hough map itself is extended by being weighted with the probability of track parameters matching an observed TS. The original design was meanwhile using a binary representation, solely describing whether parameters are matching an observed segment. The design of an FPGA-based solution using this approach is the core of this chapter. Both algorithmic and detector specific aspects of the approach were investigated and developed at the Max-Planck Institute (MPI) Munich which is for example described in Ref. [Ska19]. In summary, the investigations conducted there show that the accuracy of the overall track estimation is greatly improved by using this approach.

6.1.1. Functional Description and Processing of the Proposed Approach

The S3D is algorithmically in many ways similar to the present 2DS. It has, however, some differences which make it more complex for its adoption on FPGAs. The data flow can hereby be separated into multiple sequentially processed functional components. The first component in this flow is the creation of the 3D hough map itself. This is followed up by a threshold filter that and an optional 3D clustering which is coupled with an estimation of the center-of-gravity. The last processing step is represented by the selection of the suitable hits matching the estimated 3D-track. All of these components have to be

revised internally when compared to the original 2DS, however many of the implementation concepts are shared and can thus be reused.

Similar to the data flow of the S3D, the first step of 2D track finding consists of generating a 2D hough map in this case solely using aTSF. The new processing element to be considered at this stage is the inclusion of the remaining sTSF. With the help of their inherent orientation, these TS form the basis for extending the hough map towards the third dimension by adding the calculation of theta-planes. Each theta-plane is represented by a separate new 2D hough map. The complete 3D hough map is then generated by calculating a 2D hough map for each separate theta-plane.

Each hough map consists of a set of hough cells. Each of these cells is hereby assuming a distinct value which is representing the accumulation of the conditional probability for a track parameter set belonging to an observed TS that is currently active in the drift chamber. The conditional probability is hereby calculated by following Bayes-theorem as described in Ref. [Ska19]. These probabilities are then normalized for each cell of the map by taking both the mean value and the standard deviation of all related TS into account. This normalized probability is then represented either binary, that indicates surpassing of a predefined threshold, or by a configurable bit size. In the subsequent generation of the hough map, this probability is used as a weight that is essentially representing the contribution of a respective TS towards being represented by a certain track. The option of using non-binary weights represents one of the major differences to the 2DS, in which such an option is not available. The idea here is to have a higher resolution for representing the contribution of a TS for the track finding. This added resolution, is in turn, allowing more precise estimations of the track parameters. The bit width of the weights is one of the available design-time optimization parameters. The trade-off between higher and lower bit-widths is in the precision and resource consumption, as higher bit widths require additional hardware resources.

After the generation of the hough map, a filter is applied to it in order to reduce both the data to be stored and the space to be searched in the later stages of the processing chain. A threshold is hereby applied to all of the cells. Each cell is checked for whether it is surpassing a constant value defined during design-time to not be suppressed. At this stage, the reference implementation of this approach is commencing the processing by finding clusters of neighbouring active hough cells. A simplified alternative to this is to just find the maximum cell and choose it to represent the best estimation for the track. The reason for considering a simplified alternative is the added processing complexity of clustering especially compared to the 2DS. This is due to the third dimension that has to be taken into account. The reference implementation is hereby using DBSCAN [95]. The actual track candidate is then at the end determined by calculating the center-of-gravity within this cluster.

6.2. System Requirements

As with all components of the CDCTRG, the S3D has to fulfil the requirements set for the L1 trigger system. These are defined in terms of latency, throughput and compatibility characteristics to be provided for successful integration. These requirements are

discussed for the S3D within this section.

Connectivity

From the trigger system point of view, the S3D takes on the same role as the 2DS. The method uses the found TS and tries to reconstruct a particle track most suitable to match all of the observed segments. The big difference between both approaches lies in the estimation of the additional three-dimensional track parameter. This cannot be performed by only using segments from axial SLs, rather it is required to also take stereo SLs into consideration. In order to support this, additional communication channels have to be established that are connecting the S3D to all of the sTSF. This also means that the selected FPGA platform has to provide additional GT lanes for receiving this information. Each sTSF is hereby sending data over four GTH lanes just like the axial Track Segment Finder (aTSF). Since there are four SL with stereo orientation, an additional 16 GTH lanes must be supported in total. This can, however, be reduced by using the same approach as described in section 5.2.1 in which the complete space of the CDC was partitioned into quadrants, for which separate FPGA-boards are responsible. This is reducing the demanded GT lanes by half.

While the additional connections to sTSF are mandatory in order to be operational, there are some options with regard to the required ports for the outgoing data and the inclusion of ETF. This additionally depends on the chosen integration method with regard to the NNT as it can be integrated together with 3DS on one FPGA platform. In the case of a joint integration, GTH lanes can be saved between both systems when compared to a separated integration. In the case that the S3D and NNT are implemented separately, further lanes are necessary for the communication between both as the NNT requires the track estimations. The total number of lanes required is depending on the number of 3D-tracks to be sent. For this purpose, the matching TSs have to be sent in addition to the three track parameters. The total amount of data to be sent for one data clock cycle is at 225 Bit as shown in equation 6.1. As each lane is capable of sending up to 170 Bit, this results in at least two GTH lanes necessary to implement this connection for just for one track. Each additional track to be sent hereby requires at least one additional GTH lane. Thus it is highly desirable to pursue a joint integration in order to reduce both the latency and required number of GT lanes.

$$\begin{aligned} S3D_{SingleTrack} &= TS_{Data} + TrackParameters + Status_{Data} \\ &= 9 \cdot 21 + 3 \cdot 8 + 12 = 225Bit \end{aligned} \tag{6.1}$$

An interesting alternative to the joint integration is the relocation of the NNT's preprocessing to the FPGA hosting the S3D. In that case, only the inputs used for the MLP have to be sent across the two boards. As a side effect, the resource consumption at the NNT is reduced as it only has to implement the network itself. This solution would reduce the demand for GT lanes to just one for every possible configuration of supported tracks.

The NNT is the only data sink of the S3D in case that the current 2DS is kept operational in parallel. However, if it were to be replaced entirely, the S3D will be required to additionally send the track information to the 3DS and the GRL. The same considerations as

done before in the case of the NNT meanwhile apply to the connection of the 3DS as they have the same principle interface.

The total number of GTH lanes to be provided is therefore at least 28 lanes with regard to equation 6.3 for each quadrant. This already exceeds the maximum number of available lanes on the UT3, which is limited to 24 lanes. Thus, this platform already falls short of being a suitable host for the S3D. However, the newer UT4 is supporting up to 32 GTH lanes by using the newer FPGA architecture of the Ultrascale generation.

Assuming that the UT4 is used as a host, four GTH lanes are remaining available in the case that one track per data clock cycle is to be sent. The remaining lanes can then be used to send more tracks. Since the 3DS can natively handle up to four tracks, these additional lanes can be assigned for transferring the full number of tracks towards it.

$$\begin{aligned} TotalInputGTH &= axialGTH + stereoGTH = 5 \cdot 2 + 4 \cdot 2 = 18 \\ TotalGTH3D &= NNTGTH + 3DSGTH = 4 + 4 = 8 \\ TotalOutput &= GRL + B2Link = 2 \end{aligned} \tag{6.2}$$

$$\begin{aligned} TotalGTH &= TotalInputGTH + TotalOutputGTH \\ &= 18 + 8 + 2 = 28 \end{aligned} \tag{6.3}$$

As shown in section 5.2.3.2, receiving data from the TSF must be handled by using a persistor module that takes into account the operational characteristics of the CDC. As the S3D is receiving data from each of the TSF, a separate persistor must be instantiated for each SL. This will lead to higher resource demand compared to both the NNT and 2DS, which are the other components currently using persistors. While this is complicating things for the S3D it is advantageous for the NNT since it will receive already matched TS and does not have to use persistors anymore.

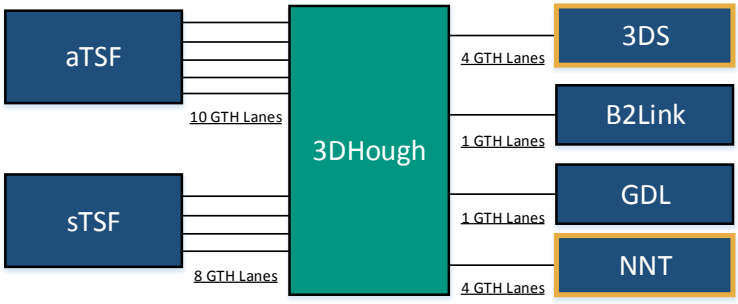


Figure 6.1.: Block diagram of the S3D describing all required input and output interfaces with the number of respective required GTH lanes. Interfaces depending on the integration method are highlighted orange.

Considering both DQM and SC basically the same approaches as discussed in section 5.3 can be reused by using both B2L and VME. The increase in the amount of input data must also be taken into account here since it is used for monitoring correct inputs. The overall required interfaces are shown in a block diagram in figure 6.1.

Latency

As with all components, the maximum latency is determined by the CDCTRG by the deadline of the GRL/GDL. As the S3D allows joint integration with the NNT, the available latency budget also depends on the selected integration strategy. In case that the 2DS is to be replaced, the latency of the 3DS has to be considered as well. The maximum allowed latency is thus determined by its two receiving components 3DS and NNT, which have to be able to complete their tasks within the total latency of the trigger system. The easiest approach to estimating the available latency budget for the S3D is hereby to apply the same requirements as they are present for 2DS. As described in section 5.4.2.2, in the current setup around 251.2 ns of processing time is allocated to 2DS which is resulting in the trigger system that is just barely within the set deadline. In case of a separated integration, the same latency budget can be used for S3D resulting in a system fulfilling the requirements. If, however, the S3D is integrated together with the NNT, a higher latency budget is available for the combined solution. In this case, the latency budget consists of both the current processing latency of NNT and 2DS together with the time necessary for sending the data between the two boards. Both latency budgets are shown in equations 6.4 and 6.5. Especially the transmission delay makes a huge difference and is allowing more flexibility for scheduling of the operations.

$$Latency_{S3D} = Latency_{2DS} = 251.2ns \tag{6.4}$$

$$\begin{aligned} Latency_{S3D+NNT} &= Latency_{2DS} + Latency_{NNT} + Latency_{GTH} \\ &= 251 + 300 + 300 = 851.2ns \end{aligned} \tag{6.5}$$

Throughput

Similar to the 2DS, a specified throughput for the found tracks is expected from S3D to achieve optimal operation. However, as with the NNT, there are some degrees of freedom in the number of tracks to be found in parallel at each data clock cycle. The version of the 2DS that is operational at this stage of the experiment, is sending up to four estimated tracks per board and data clock cycle. The limitation is set by the maximum achievable data rate achievable by using the GTH lanes of the UT3. The implementation itself is capable of finding and estimating up to six tracks in parallel for each data clock cycle. While the NNT in its first fully operational version can only process one track for each data clock cycle, its implementation on the more powerful UT4 is capable of processing up to two tracks, with the assumption that the smallest available FPGA is used. In a CDCTRG in which the S3D is solely used to improve the NNT, the maximum number of tracks to be estimated is determined by its implementation. In case that the 2DS is replaced, at least

four tracks shall be estimated which is matching the current maximum throughput.

Flexibility

Demand for flexible implementation of the S3D is mostly the result of using weights as the content of hough cells which are derived during the training phase using a data set. As the data set used for training can be changing over time, for example, by switching from a simulation data set to data gathered during physics operation, the weights have to be adjustable over time. Overall this situation is similar to the weights used for the MLP of the NNT. It is thus the best approach to again established a semi-automated toolset in order to generate a firmware configuration quickly and effortlessly.

Summary

The requirements of the featured S3D are in general very similar to the requirements set for the 2DS. Since the S3D is basically an alternative or replacing approach it is integrated into CDCTRG at the same position with regards to the data flow. Meanwhile, the baseline for both the maximum latency and minimum throughput in standalone operation is taken from the current 2DS and set as a target for the S3D.

The main differences with regard to integration are the upgraded FPGA-platform that is planned to be used, the UT4, and the inclusion of all of the TSF information, while the 2DS is only relying on aTSF. A rough estimation of the required total amount of GTH is basically excluding the usage of the older UT3 platform and makes usage of the UT4 or an alternative platform necessary.

The S3D has also to fulfil hard requirements on both latency and throughput. Since the component is located in the middle of the CDCTRG data processing chain, its influence on subsequent stages has to be considered. Overall, the latency of the 2DS must be matched. This was already at the limit of what was feasible in the original design but was also based on older Virtex-6 FPGAs. In the optimal case, the throughput achieved by the 2DS is also matched. However, there are degrees of freedom in the number of tracks processed in parallel, while more processed tracks are further increasing the efficiency.

6.3. Realization of the S3D

Having established the requirements to be fulfilled as well as the algorithmic fundamentals of the track finding approach, the focus is put on the realization of the FPGA-based S3D. At first, all aspects regarding the integration into the CDCTRG are discussed, especially the options for joint integration with the NNT. Following the integration considerations, the focus is shifted over towards the implementation of the logic.

6.3.1. Integration into the Trigger System

There are three basic approaches for integrating the S3D into the CDCTRG. As it is basically an extension of the current 2DS, adding an additional track parameter, the S3D can

be used to replace the previous 2DS entirely. Even though that might seem the only logical conclusion as it is reducing the hardware effort, there are several arguments against this approach. Since the 2DS is essential for the entire trigger system to be operational, it must first be proven that any alternative realization can achieve comparable or better performance in terms of tracking before replacement efforts can commence. This has to be conducted thoroughly to prevent the hindering of the experiment.

Alternatively, it is possible to operate the S3D in parallel to the 2DS in order to facilitate smooth operation and transition. The big disadvantage here is the need for additional hardware, especially with regard to the E-Hut. New crates and space have to be allocated within the E-Hut accompanied by the need for a new cabling plan.

A much better solution is available in case the S3D is integrated into the CDCTRG together with the NNT on the same FPGA board. Integrating both components on one board would save cost, area and simplify the cabling. In addition, it would provide advantages regarding the latency. A joint integration of both components has the most advantages, the question herein is whether the available resources of the FPGA are sufficient for realization. The options are additionally shown on an architectural level for all options in the figures 6.2, 6.3, 6.4, except joint implementation without 2DS.

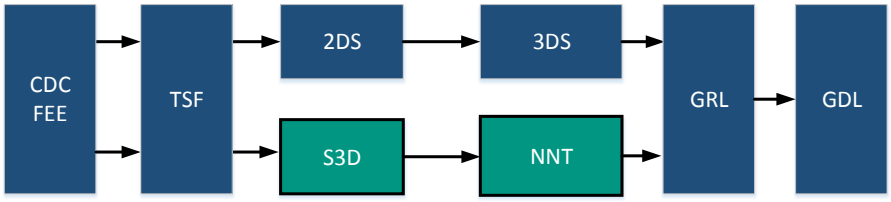


Figure 6.2.: Architecture for separate integration into the CDCTRG without replacing 2DS.

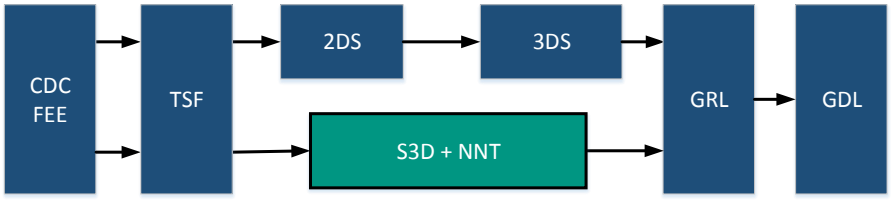


Figure 6.3.: Architecture for joint integration into the CDCTRG without replacing 2DS.

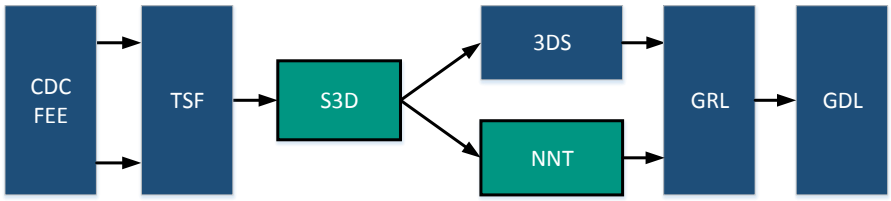


Figure 6.4.: Architecture for separate integration into the CDCTRG replacing 2DS.

6.3.2. FPGA Architecture of the S3D

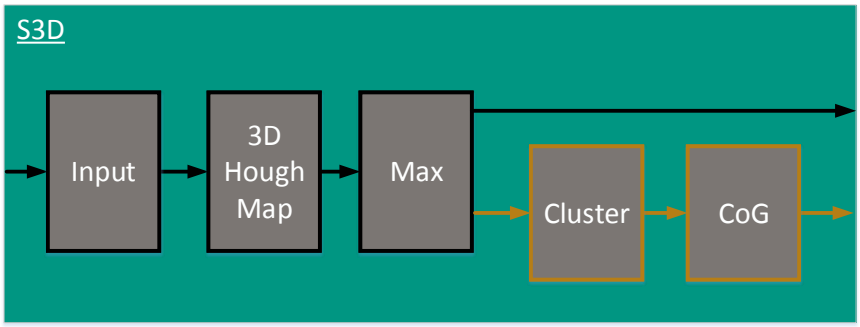


Figure 6.5.: Processing architecture of the S3D. Orange boxes are representing the optimal path in terms of most precise track estimation. It is a refinement step, which can be bypassed at the cost of worse estimations.

The hardware architecture of the S3D can be divided into four separate components according to the tasks to be performed. As with the NNT, input data handling is required to support the TSF. After the reception of TSs, the first stage of processing is responsible for the creation of the hough map by using the available pool of observed TS. The input to this stage depends on the used configuration of the TSF. As such at least 10 TS are processed at each data clock cycle, other possible configurations are 15 or 20 TSs. The following processing steps are then applying a set of operations onto the generated hough map. At first, a threshold is applied to suppress low-probability tracks. Afterwards a set amount of maxima is found within the map. These maxima are already viable results of the S3D, that can be used by the following components. However, the optimal path as defined in the reference implementation is calculating a center-of-gravity using a cluster around each of the found maxima. As such the architecture can be configured to either stop at the maximum finding or proceed to calculating the centre-of-gravity. An illustration of the entire architecture is shown in figure 6.5.

6.3.2.1. Generation of the 3D Hough Map

The first component of the architecture is the handling of the input data received by the TSF. In principle, it has to fulfil the same tasks as was the case for the NNT. Besides complying with the communication protocol, the main elements are the persistors, which accumulate the TS observed within a defined time window. One solution is to reuse the implementation discussed in section 5.2.3.2. The S3D is however by nature different in its algorithmic implementation, which allows for an alternative solution. The core idea behind this alternative is to implement the persistors directly within the generation of the Hough map instead of creating the pools of TS beforehand. As a result of this approach, the persistors implementation is taking part after the input handling.

The generation of the hough map consists of filling its individual cells, which are representing the possible track parameters, with the weights of the currently observed TS. There are in principle two approaches to this, in one of these the hough map is filled by using the current pool of TS. In the alternative approach, partial hough maps that are only containing the information of one TS are generated at first independently of each other. These partial maps are summed up afterwards to calculate the complete map. Within this implementation, the former approach is chosen. The reasoning behind this choice is that partial hough maps are either unnecessarily memory intensive, as several maps have to be buffered before finally being summed up, or in case they are zero-suppressed can only be summed up with higher computational effort, especially with regard to its latency.

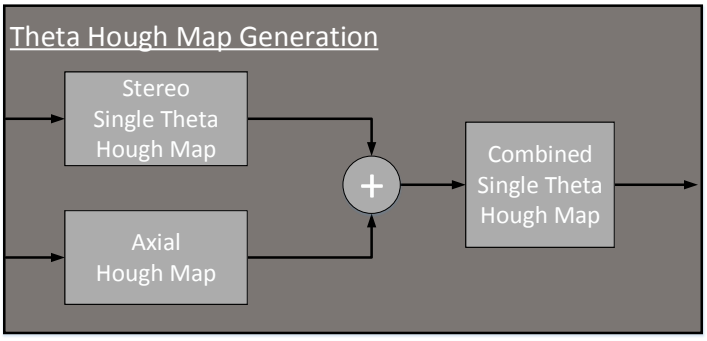


Figure 6.6.: Architectural view on the combination of axial and stereo Hough maps. The axial Hough map is added to each theta-Hough map separately in parallel.

The chosen basic principle for creating the Hough map is to mark every cell, which is associated with a currently active TS. With that relationship known, the corresponding weights are subsequently loaded and summed up across all of the cells. The combination of all individual cells is then forming the three dimensional Hough map that is reflecting the current state of the experiment from the CDC's point of view. This can in principle be implemented by using one adder coupled with a memory that stores the relationships and weights. In order to meet both throughput and latency requirements, parallelism is

facilitated as much as possible. This is always accompanied by the decomposition of the entire task and allocation to separate modules.

The first module to be defined has the task of generating a 2D-Hough map. The reason behind this is to separate the processing of both the different SL orientations and the theta slices of the complete map. As those are each handled differently, it is the better solution to instantiate different modules instead of implementing a common one. Inputs to these modules are the respective TS to be processed, at this point stereo and axial wires are separated and processed on parallel data paths. Afterwards, both partial hough maps are added to generate one theta plane of the entire map. This architecture is shown in figure 6.6.

Internally, input data is additionally separated according to the separate SLs, the hough map is hereby generated for each layer in parallel using only the TS present within that layer. This reduces the amount of TS to be processed significantly, thus lowering the latency for generating the map. However, for each SL a separate map is buffered before being merged together. The architecture of this separated processing is shown in figure 6.7. The same separation can be applied to processing the different priority positions that a TS can assume, thus first priority hits are processed in parallel to second priority hits. The core of generating each partial hough map consists of processing each individual cell. Which will be the point of focus in the following section.

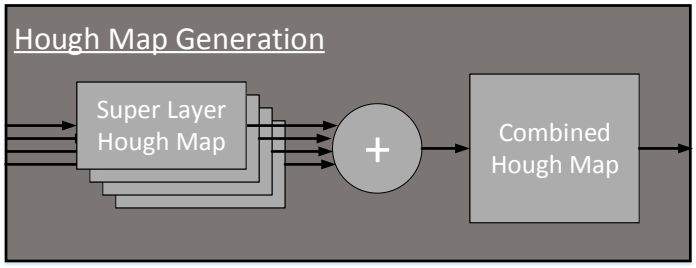


Figure 6.7.: Architectural view on the generation of a Hough map. It is performed separately for axial orientation and each theta.

Calculation of a Single Track Probability Represented by Hough Cells

Each cell of the hough map can be processed the same way. The first stage is hereby to check whether a currently active TS may have been caused by the track representing the cell under consideration. In case that TS and track are related to each other, the weight, representing the probability is to be used. Doing this for all TS and summing up the weights concludes the processing of the hough cell. Meanwhile, these operations can be calculated independently for each cell of the map. Maximum parallelism is employed here in order to minimize latency, thus each individual cell is processed in parallel. As the operations are always the same, independent of the specific cell a single module is developed to carry out the calculations while being design flexibly to load weights effortlessly.

The option for sequential processing is however provided by the implementation of this module as well, in case FPGA resources are not sufficient and time budget is available.

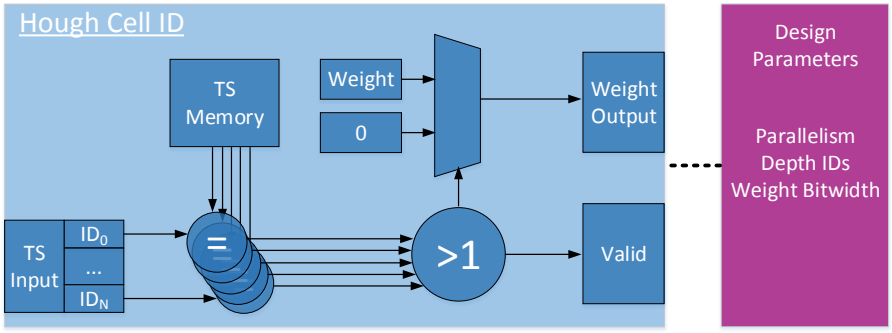


Figure 6.8.: Architecture of the module responsible for determining the cell that is related to a currently active TS.

The module is receiving the IDs of all TS from the respective SL at each data clock cycle. In order to determine which TSs are related to a respective cell within the hough map, all IDs contributing to one cell are stored within a TS memory. The contents of these cells are defined during design time as the relationship is part of the algorithm's training process. All incoming IDs are then compared with the contents of this memory. In case one or more IDs are matched, the hough cell is sending a valid signal indicating that it is to be considered for further processing. In parallel to this, the respective weight is loaded and send along with the valid signal. Weights are hereby stored in a separate memory that is again filled during design time. The architecture of this module is shown in figure 6.8.

To synthesize the module for the FPGA, the size of both memories must be determined in advance at design time. The size must be chosen so that the calculation of the map stays within the resource budget. In case that not all of the possible TSs fit into the memory, a good trade-off can be found by not using TS with a low probability contribution to its respective track. This way subsequent tracking accuracy is reduced the slightest. The maximum number of TS stored for the developed prototype was defined by using a provided sample data set. With regard to this data set, a memory depth of up to five IDs is already sufficient to provide storage to fully store all IDs for all of the cells.

The complete hough cell is then processed by instantiating several single ID processing units as illustrated in figure 6.9. The determined weight for each found TS is then summed up and passed along to the next processing stage at which the cell's weight is combined with all the remaining cells processed in parallel. Parameters that are set during design time are the bit width of the weights, parallel comparisons at each clock cycles and the depth of the used memory.

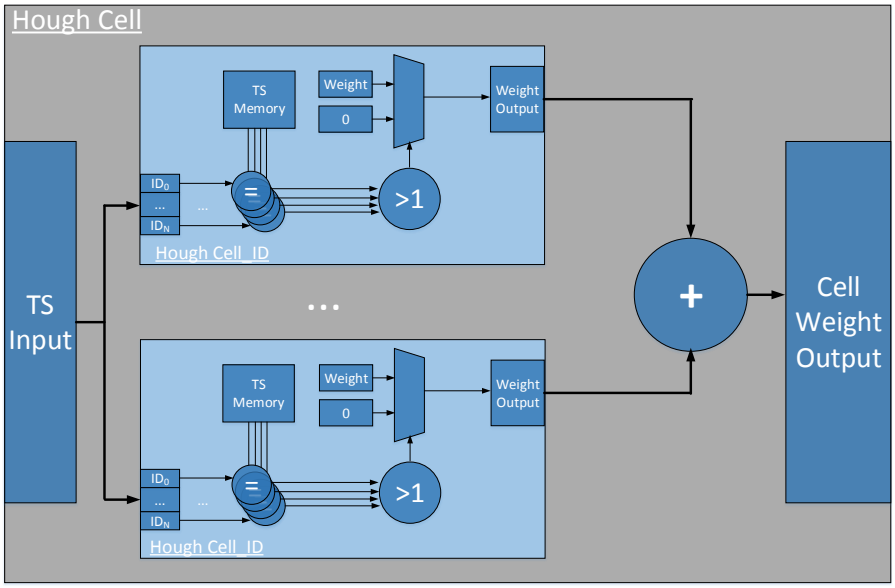


Figure 6.9.: Architecture developed for the calculation of a cell's accumulated weight depending on the found TS IDs.

Consideration of TS Persistence within the Cell Processing

As described before, the persistence of TSs can be integrated into the creation of the Hough map. The basic idea is hereby to store the result generated at the cell processing of one TS for a fixed time interval instead of a single clock cycle. At each successive clock cycle, this stored result is reused and updated with the currently generated result for the newly arriving TS. Buffered intermediate results are hereby invalidated after the expiration of the defined time interval. Compared to the original approach that was implemented for the NNT, this alternative has several advantages. Firstly, there is no need for the inclusion of a separate persistor module. This is already a significant improvement, as the persistor is one of the bottlenecks of the NNT in terms of resources. Since the S3D is using not only receiving a subset of the TSFs but all nine, the impact is even more significant compared to all other components. The other advantage is that TSs that were already processed in previous clock cycles are not processed again. Instead, their previously calculated weight is used again. As a result, however, the processing unit is including an additional memory together with a clock counter that represents the time window of validity.

Evaluation of Resources, Latency and Throughput

The presented approach for generating hough maps was synthesized for the UT4 and UT4+, that is Ultrascale and Ultrascale+. The UT4 option are hereby distinguished between the prototyping option, UT4 base hosting a VU80, and the expected final platform

UT4 max, hosting a VU125. Results are shown in table 6.1 for a single cell of the map and in table 6.2 for the entire hough map. The results were achieved with the configuration listed in table 6.3. In total it is possible to operate the Hough map generation with a clock frequency of 200 MHz at the UT4 and 255 MHz at the UT4+ with a latency of one clock cycle. This means that the required input throughput frequency of the TSF at $31.75 MHz$ can be easily matched. The substantial difference between both clock frequencies can hereby be used to stretch cell processing across several clock cycles in the case that overall resource utilization is high. The resource consumption for processing a single cell is very small, however, it has to be used in large numbers number, as a large amount of Hough cells have to be processed in total. This can be seen in the high overall resource demand for the generation of the entire hough map. The result of this module is the entire 3D hough map. Since it considers all of the incoming TS only one instance of this module is to be implemented, as such the results are acceptable.

Category	UT4	UT4+
Slice LUTs total	36	36
FF total	8	8
Frequency in MHz	200	245
Latency in clock cycles	1	1

Table 6.1.: Synthesis results for the calculation of one Hough cell's content.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	161 715	158 480	156 330
Slices LUTs percentage	36 %	22 %	26 %
FF total	10 853	10 636	10 262
FF percentage	1 %	1 %	1 %
Frequency in MHz	200	200	245
Latency in clock cycles	1	1	1

Table 6.2.: Synthesis results for the entire 3D Hough map.

Parameter	Depth	Parallelism	Bit width weights
Value	5 IDs	full	3 Bit

Table 6.3.: Configuration used to generate the synthesis results.

6.3.2.2. Finding the most Suitable Track Candidates

The first stage of internal processing is concluded with the creation of the Hough map. The next task to be performed is the determination of the track parameters most closely

matching the observed event. For this, at first, the number of possible candidates is reduced by using a threshold filtering. This is implemented for every single cell separately and performed in parallel, again with the goal to keep latency small. Although the resulting map is substantially reduced after this filtering stage, it is still representing a large space that is to be investigated. The used approach to find suitable track candidates within this space is described in the following. The idea behind this approach is to partition the entire map into a number of subsets to be searched for a local maximum probability cell. The reason for the partitioning of the map is to keep the latency small and facilitate parallelism by searching smaller subsets of the map and enabling parallel search. As there its no preferred way to partition the map, it is separated into subsets all having the same size. The number of partitions to be used is meanwhile a parameter to be determined, however, the most straightforward way is to set it to the number of track candidates to be found. For the prototype investigation, it will be assumed that four candidates are to be found, as it recreates the current implementation of the 2DS. Using this approach, there is a chance that multiple maxima are present within each of the resulting quadrants. The task of choosing the maximum to be used is again a design decision to be determined depending on the required quality of the results. For this investigation, it is performed in a way that is reflecting the 2DS. This means that the track candidate to be considered to be used in each quadrant is the one with the highest probability ordered by its impulse. To minimize latency, the search for a candidate is performed by separate modules for each quadrant. These modules can be operated in parallel, significantly reducing the total latency. The resulting architecture is depicted in figure 6.10.

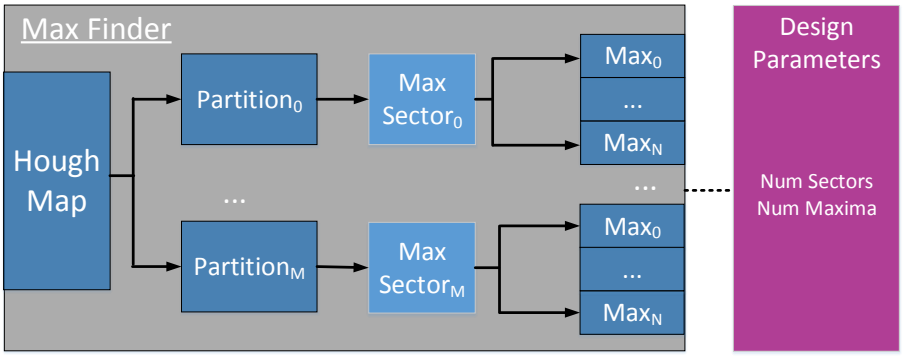


Figure 6.10.: Architecture of the track candidate finding using a partitioned Hough map.

The module for finding suitable track candidates was synthesised for the same platforms as before. The results are summarized in terms of resources, latency and clock frequency in table 6.4 for one quadrant and in table 6.5 for the entire map. The latency can be set as desired, however, the best trade-off between all figures of merit was found at four clock cycles. This is additionally the lowest achievable latency while keeping the maximum frequency at over 200 MHz, thus preventing this module of being the bottleneck. The

achieved resource utilization is, in general, reasonable and much smaller than the generation of the Hough map itself.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	6 809	7 068	7 222
FF total	10 494	9 892	9 970
Frequency in MHz	200	200	245
Latency in clock cycles	1	1	1

Table 6.4.: Synthesis results for one quadrant of the 3D Hough map.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	24 513	25 448	26 002
Slices LUTs percentage	6 %	4 %	4 %
FF total	23 320	21 983	22 157
FF percentage	3 %	2 %	2 %
Frequency in MHz	200	200	245
Latency in clock cycles	1	1	1

Table 6.5.: Synthesis results for the entire 3D Hough map.

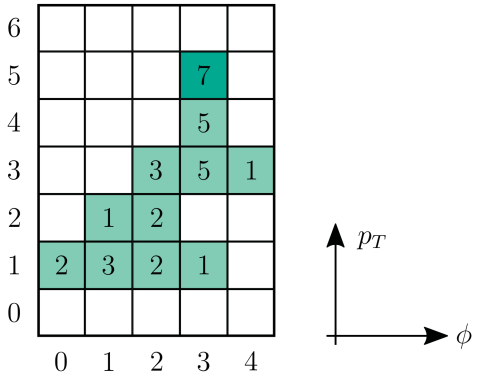
6.3.2.3. Refinement of the Track Candidate

In principle, the already found maxima could be used as a lower-accuracy estimation of the desired track parameters. Further refinement of the found candidate is however possible by using the information about neighbouring active cells in the hough map as there is still some probability that the correct track is to be found in those cells. Such refinement of the estimation is the topic of this section in which the centre-of-gravity is calculated for a connected cluster formed around the found maximum cell. It includes two general approaches the unconstrained and area-constraint clustering. With the later having to abide by a restricted maximum cluster area.

Unconstrained Clustering Approach

The unconstrained approach does not impose any restrictions on the maximum size of the found clusters. Algorithmically, the approach taken for this is resembling a region growing in which active cells of the map are at first detected and combined to multiple successive neighbouring cells. These combined cells shall be called strips and are formed along one of the dimensions for example phi. After all, strips are formed, the region is expanded towards another dimension in a subsequent, that is either pt or theta. The advantage of this approach is the high degree of parallelization for combining cells since rows of cells can be processed independently. The presented approach to unconstrained

clustering is hereby based on the master thesis Ref. [Hoc18].



Row	StartCol	EndCol
5	3	3
4	3	3
3	2	4
2	1	2
1	0	3

Figure 6.11.: Schematic description of the approach used in the unconstrained clustering [Hoc18].

As it has no specific constraint this approach is finding all of the clusters while providing a high degree of parallelism, which is desired in order to achieve low latency. However many cells might be merged without need as they are not part of the cluster chosen for the representation of the track. These additional operations are herby potentially wasting processing resources. As the goal is not only the realization of the S3D but also a unified integration with NNT, a more resource-efficient solution is presented in the following. Instead of considering all of the cells in the Hough map, the approach taken alternatively is to only start from the already found maximum weighted cell and expand the cluster from this cell by merging the row of the maximum with its neighbouring rows in one dimension. This merging is continued until no neighbouring cell rows with active overlap can be found. After the merging in the second dimension, the process is expanded to next. In the case of the S3D, the order in which dimensions are processed is phi, pt concluding with theta. This approach is shown in figure 6.11 for a 2D-Hough map that shows an example cluster to be formed. Starting from row five in which the maximum is located, the cluster is expanded towards lower pt, as this is the direction at which rows of active cells are overlapping. The subsequent expansion into the third dimension is then illustrated in figure 6.12.

This approach reduces resource consumption by two measures. First, the cluster is only formed around the maximum cell. Then it is expanded in a row-wise manner in which a maximum of two rows can be merged in parallel. This, however, is representing the downside of this approach as it makes latency indeterministic since the number of rows to be merged is varies with the currently processed hough map. This drawback is, in conclusion, the reason for the investigation of area-constrained approaches that have a fixed latency.

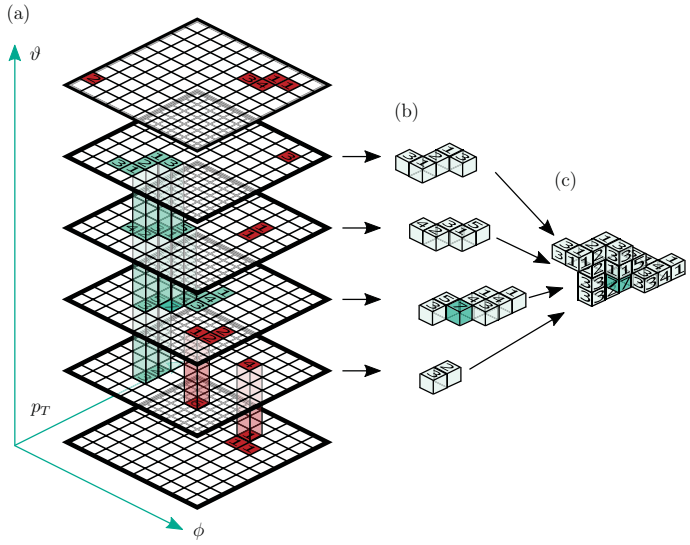


Figure 6.12.: Graphical description of the 3D clustering approach employed without using area-constraints. Weights are written into the separate cells, with dark green indicating the maximum cell, light green cluster members that are to be found and red cells are showing cells that are not part of the desired cluster [Hoc18].

Evaluation

The implementation of the unconstrained clustering was performed using the HLS tools by Xilinx. Several options were hereby enabled such as array partitioning, array reshaping and pipelining, which led to the generation of an architecture with parallel processing. The results using this approach for the UT4 with the commonly used options are shown in table 6.6. Looking at the results it is apparent that this approach cannot be used with all of these FPGAs as it both exceeds the resources budgets of the LUTs and the latency, which is in addition variable depending on the size of the cluster that is to be found. At this point, there are two straightforward approaches to addressing these shortcomings. For once this implementation is covering the entire Hough map, which will be reduced when using multiple FPGA boards that are covering the different quadrants. While this might solve the resource problem, latency has still to be addressed. The reason for the high and variable latency is hereby due to the variable cluster sizes. By constraining the maximum area that can be assumed by a cluster, this problem can be addressed as it is done in the area-constrained approach that is presented in the following.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	1 498 072	1 509 158	1 463 018
Slices LUTs percentage	338 %	211 %	243 %
FF	382 125	390 149	374 864
FF percentage	45 %	26 %	31 %
Frequency in MHz	200	200	218
Latency in clock cycles	103 to 3961	10 to 3961	103 to 3961

Table 6.6.: Synthesis results for the unconstrained clustering approach.

Area-Constrained Clustering Approach

The biggest drawback of the first clustering approach is its high and variable latency in the worst case due to the possibility of a cluster spanning across a large area. An alternative approach, to this is to constrain the maximum size a cluster can assume. The idea here is to define an area around the maximum weighted cell in which cells are considered for merging into a cluster. This fixes the number of cells to be considered and prevents excessive latency. The presented approach to area-constrained clustering is hereby based on the master thesis Ref. [Hua19].

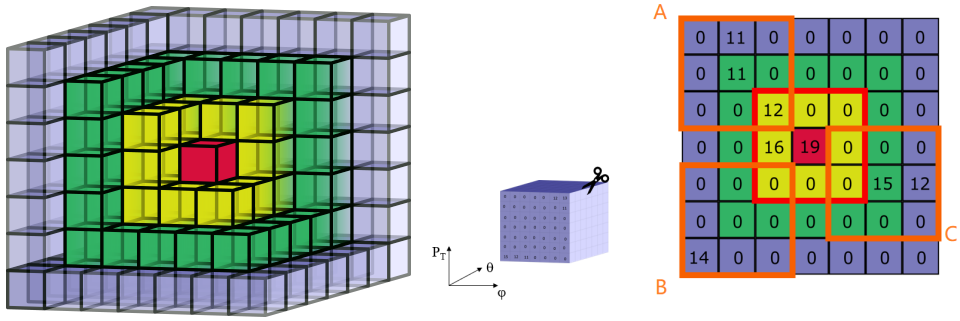


Figure 6.13.: Illustration of the area-constrained clustering approach. Starting from the maximum weighted cell, shown in red. Surrounding areas are formed and checked for active cells as shown in (a). The progression of cluster expansion is indicated by the different colours yellow, green then blue. An example for this is shown in two dimensions in (b), in which only the cluster A is merged with the maximum [Hua19].

The first design question to be answered for this approach is how to set the parameters of the area to be searched. This choice is representing a trade-off between accuracy and resources/latency. The answer to this is meanwhile depending on the size of clusters that lead to desired tracks. In the case that most of them are exceeding the defined area

to be searched, accuracy might be significantly reduced. The approach to this is to first choose a reasonable size that will result in a feasible implementation which will then be used to represent a base solution to the problem. Additional mechanisms are meanwhile investigated to increase the accuracy. Using the provided simulated data sets in which clustering was performed by using DBSCAN, most of the clusters to be considered are sufficiently covered by a cluster area of (phi,pt,theta)=(7,7,7).

The approach to finding the cluster is illustrated in figure 6.13. Here, at first all of the neighbouring cells around the maximum are checked for their weight, which spans a (phi,pt,theta)=(3,3,3) cube around the maximum. The idea is now to expand this approach to all the active neighbours within this area in order to generate a cluster in an area representing a cube of (phi,pt,theta)=(5,5,5) and then (phi,pt,theta)=(7,7,7) around the maximum. From an implementation point of view, each neighbouring cell can be checked in parallel throughout all processing stages. However, the progression in area can only be performed sequentially since the previous active neighbours must always be known to advance. This is generating a good mixture of parallel processing for achieving low latency and sequential processing for the reduction resource consumption.

This approach was implemented for the available options based on the UT4. The results are shown in table 6.7 for one quadrant and table 6.8 for the entire hough map. It is immediately observable that the new approach is faring better than the unconstrained clustering. Both resource and latency are this time well within the available budget of the FPGA and CDCTRG even for parallel processing of all quadrants. In addition, the latency is now constant due to the constraint size of clusters.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	42 345	39 430	38 054
Slices LUTs percentage	10 %	6%	6%
FF total	114 764	105 424	110 998
FF percentage	13 %	7 %	9 %

Table 6.7.: Synthesis results of the area-constrained clustering approach for one quadrant.

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	153 055	143 920	136 616
Slices LUTs percentage	35 %	20 %	23 %
FF total	233 475	214 477	223 786
FF percentage	27 %	15 %	19 %

Table 6.8.: Synthesis results of the area-constrained clustering approach for the entire Hough map.

Category	UT4 base	UT4 max	UT4+
Frequency in MHz	200	200	225
Latency in clock cycles	6	6	6
Throughput in million clusters	33	33	33

Table 6.9.: Performance results for the area-constrained clustering.

In addition to the implementation characteristics, the algorithm was evaluated in terms of its accuracy. For this, simulated tracks were used to create sample clusters on which the new area-constraint clustering approach and an implementation resembling the exact algorithm were performed. Afterwards, both results were compared with each other. The results are hereby shown in table 6.10 for two selected datasets. While the estimation generated by the area-constrained approach is equal to the reference implementation's result for dataset 993, there are deviations when investigating dataset 973. The reason for the deviation for dataset 973 is due to the shape of the processed hough map in which the cluster to be found is spanning across cells that are outside of the area-constrained clustering's scope. A graphical representation is shown for both investigated datasets in figure 6.14. It shows that the cluster to be processed in dataset 973 is having a long trail of active cells that leads to a wide-spread cluster, not being covered by the discussed approach.

Dataset /Category	Estimation	Reference	Deviation
Dataset 993	(11.49,38.80,2.48)	(11.49,38.80,2.48)	(0,0,0)
Dataset 973	(5.35,31.97,1.46)	(5.90,31.28,1.47)	(-0.54,0.69,0.01)

Table 6.10.: Comparison of the different achieved estimations by the presented area-constrained clustering and the reference algorithm. The values represent the Hough map coordinate in as ()phi,pt,theta)

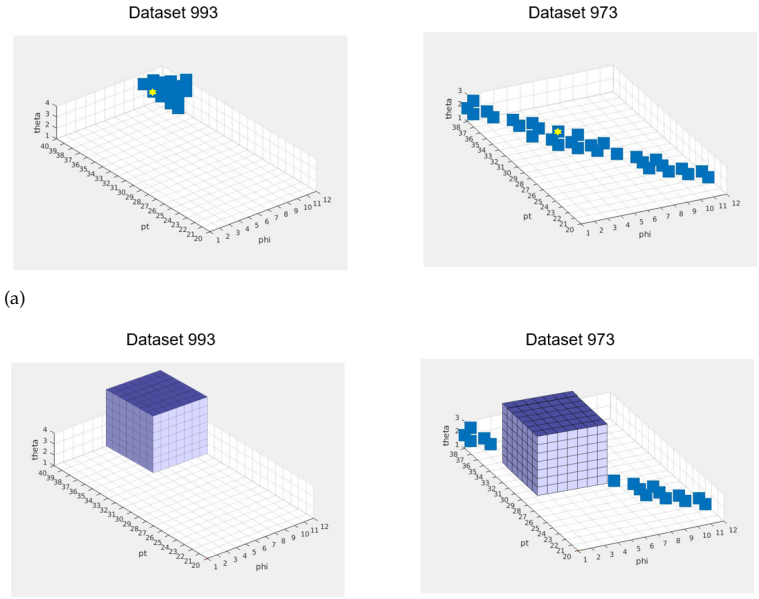


Figure 6.14.: Illustration of the used data samples with blue areas showing active cells and the maximum cell being highlighted by a star as shown in (a). The area covered by the constrained clustering is meanwhile shown by an overlapping cube as shown in (b) [Hua19].

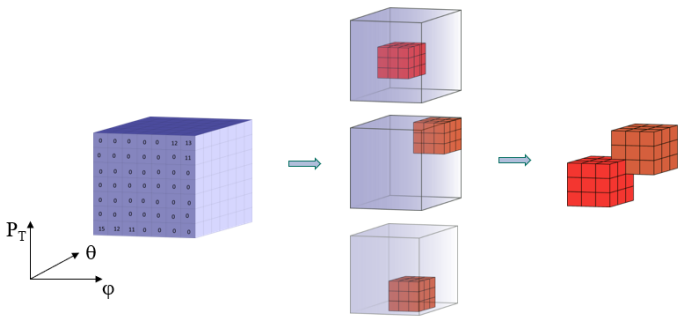
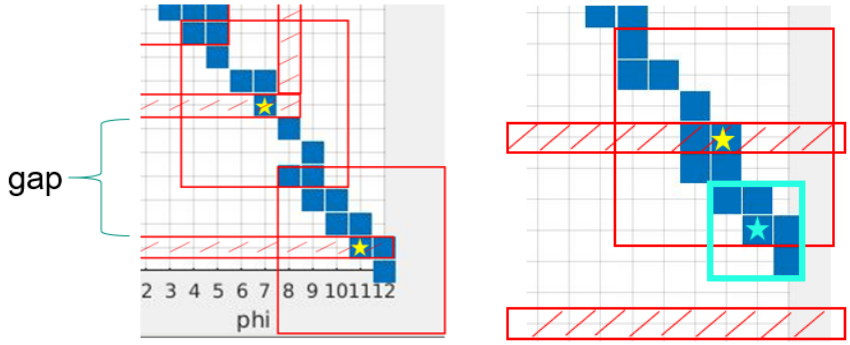


Figure 6.15.: Illustration of the global area-constrained clustering approach, in which additional assisting clusters are formed outside of the reach as defined by the maximum weighted cell. The assisting cluster is shown in brown, while the primary cluster is highlighted red [Hua19].

Global Area-Constrained Clustering Approach with Patching

The initial (phi,pt,theta)=(7,7,7) version of the area-constrained clustering can already be implemented while fulfilling all requirements. However, it has room for improvement in terms of the achieved accuracy as was shown in the respective investigation. Two additions will be introduced that are addressing this shortcoming of the initial approach. The first idea for improvement is to extend the area within the hough map that is searched. This is done by defining additional cells that will serve as additional centres around which clustering is performed. The performed clustering method itself is hereby similar to the initial idea around the maximum cell. The benefit is here that an additional (phi,pt,theta)=(7,7,7) area is investigated in addition to the original one. Extension of the original cluster's reach is then achieved by merging with the new additional cluster. The additional area to be merged is hereby located towards the direction of the main cluster's growth in terms of the total weight of the cells towards the direction. The centre of the additional cluster is then placed towards this direction with a constant distance to the maximum cell, in one dimension. For the developed prototype a distance of five in pt and phi was chosen. This value was chosen since it showed the best characteristics when evaluating the clustering using the provided test data set, as it allowed a good expansion of the original cluster. In addition to this, special cases are occurring around the boundaries of the hough map. These special cases are addressed by introducing an additional mechanism, the patching of clusters. For this, patches are added at a specific location which ate are smaller areas of cells in the map that are to be searched, for example, (phi,pt,theta)=(3,3,3). Using these patches allows covering smaller areas that are often found in between the main cluster and the boundaries of the Hough map. Both of the introduced additions require the implementation of additional new logic operations to be performed. The question is hereby whether the increase in accuracy is worth the added cost towards the resources. The general approach is additionally illustrated in figure 6.15 in which assisting cluster areas are added. An example of both added mechanisms, that is including patching, is meanwhile shown in figure 6.16.

The comparison of the track parameter estimations using the global area-constrained approach without patching is shown in table 6.12. It shows the result for dataset 973 in which the estimation was previously deviating from the correct solution. With the added mechanisms, the clustering is now capable of exactly replicating the estimation of the reference implementation. The subsequent characteristics when implementing it for FPGAs, are meanwhile shown in table 6.11. First of all, both latency and resources are still within the defined limits, with the latency not decreasing due to parallel processing of the supporting cluster. The resources demand is however significantly increased. The decision whether to use these optimizations is now depending on the overall resource consumption of the S3D together with the generation of the hough map. In addition, the most precise option, the global area-constrained clustering with patching, has to be weighed against a less accurate solution that is requiring less resource and is capable of achieving unified integration with the NNT.



(a) (b)

Figure 6.16.: Definition of an assisting merge cluster is shown in an example in (a) with the distance to the maximum cell being set to 5, as defined by the gap. Patching is meanwhile shown in (b) in which a smaller area is defined at the boundaries of the Hough map [Hua19].

Category	UT4 base	UT4 max	UT4+
Slice LUTs total	241 762	232 627	225 323
Slices LUTs percentage	56 %	35 %	38%
FF total	609 868	590 870	579 627
FF percentage	70 %	41 %	48 %

Table 6.11.: Synthesis results for the global area-constrained clustering approach with patching for the entire Hough map.

Dataset /Category	Estimation	Reference	Deviation
Dataset 973	(5.35,31.97,1.46)	(5.35,31.97,1.46)	(0.0,0.0,0.0)

Table 6.12.: Comparison of the different achieved estimations by the global area-constrained clustering and the reference algorithm. The values represent the Hough map coordinate as (phi,pt,theta)

6.3.3. Evaluation of the Complete System

Category	UT4	UT4 max	UT4+
Slice LUTs total	186 228	183 928	182 332
Slices LUTs percentage	42 %	13 %	30 %
FF	34 173	32 619	32 419
FF percentage	4 %	2 %	3 %
Frequency in MHz	200	200	231
Latency in clock cycles	15	15	15

Table 6.13.: Implementation results for the track parameter estimation based on the maximum cell.

With the design and implementation of all required sub-components established, the entire S3D can now be evaluated for its suitability to be used during operation. Three different setups are considered for this. They differ in their resource consumption and achievable accuracy. The first setup is calculating only the cell within the hough map with the maximum weight for four quadrants and then sending the associated parameters. The achieved operational characteristics of this setup are hereby shown in table 6.13. In general, this solution is fulfilling all of the requirements in terms of resources, throughput, and latency. However, none of the refinement options based on the center-of-gravity are enabled which will result in reduced accuracy. Its impact on the overall trigger system is meanwhile to be investigated, as it was currently only evaluated compared to the ideal algorithm. The second setup adds a clustering and centre-of-gravity estimation for the refinement of the parameters. Its characteristics are shown in table 6.14. Here, both latency and throughput are fulfilling the individual requirements. The resource consumption, on the other hand, is close to reaching the limit with around 70% utilization. Such a high ratio is typically rather unlikely to meet timing constraints set for the internal signals. However, the final version of the UT4 will be based on the larger VU125. When considering this FPGA, the resource utilization is much more feasible. The last evaluated setup is based on the area-constrained clustering with the addition of further refinement by using the global clustering and patching. The achieved resource characteristics are hereby shown in table 6.15. While this setup is achieving the best accuracy across all options, it has a high resource utilization even surpassing the number of available resources on the VU80. Since it improves accuracy only for a certain subset of all of the occurring clusters, it is currently not worth the additional cost in order to be considered for operation.

Category	UT4	UT4 max	UT4+
Slice LUTs total	339 283	327 848	318 948
Slices LUTs percentage	76 %	46 %	53 %
FF	267 648	247 096	256 205
FF percentage	30 %	17 %	21 %
Frequency in MHz	200	200	245
Latency in clock cycles	15	15	15

Table 6.14.: Implementation results for the track parameter estimation based on the area-constrained clustering.

Category	UT4	UT4 max	UT4+
Slice LUTs total	427 990	416 555	407 655
Slices LUTs percentage	96 %	58 %	68 %
FF	644 041	623 489	612 046
FF percentage	72 %	44 %	51 %
Frequency in MHz	200	200	231
Latency in clock cycles	15	15	15

Table 6.15.: Implementation results for the track parameter estimation based on the global area-constrained clustering with patching.

6.3.4. Design Flow

The development flow for implementing the S3D on an FPGA must be easily adaptable in order to support the already existing software framework used for simulating its software-based counterpart. This framework is implemented in python, which should be interfaced with FPGA development tools as tightly as possible. In the proposed design flow the already established reference software implementation is also used for the validation of the hardware modules. In addition to providing the reference algorithms to be matched by the hardware, it is responsible for the generation of the weights to be used and thus the definition of the hough map loaded into hardware. This hough map is hereby based on the test data set used to define the probabilities.

To address the necessity for coupling of the hardware tools with the python-based reference implementation the design flow built around the usage of the Vivado HLS tools. Using these tools makes integration of the heterogeneous toolsets easier due to its HW/SW Co-Design capabilities. In addition to the capability to interface HDL-based modules to software-based implementations, the toolset is already integrated into an environment coupled to the HW/SW co-simulation engine provided by Xilinx. The same test data used for training and prototyping of the algorithm can then be used directly for creating a

testbench without the need of translating to a representation used for FPGA development such as based on VHDL.

The strategy for implementing the actual functionality is hereby twofold, representing the development history. HLS is used for the implementation of the hough map definition, as the tools were able to efficiently implement this aspect. In addition to this, it is basically the only part that has to be interfaced with external tools with regard to configuration parameters. The clustering algorithms, on the other hand, proved to be barely realizable on the FPGA and are thus implemented on RTL-level design in VHDL in order to manually tune and optimize the implementation. The clustering is then included as an external IP core into the overall processing architecture.

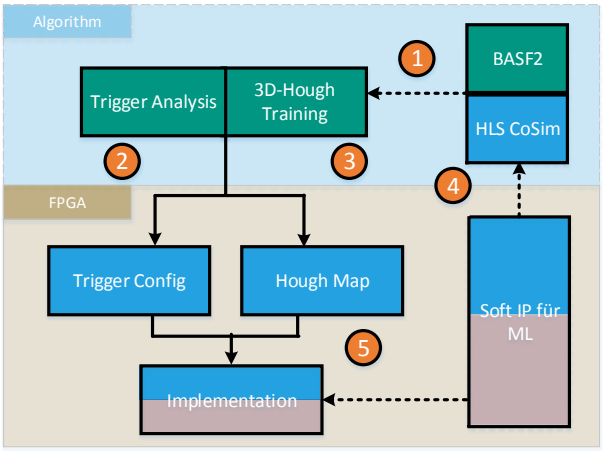


Figure 6.17.: Design flow used for generating S3D firmware.

The resulting design flow is shown in figure 6.17 and follows the reference flow introduced in section 4.3. Again the development is divided into algorithm and FPGA engineering. The algorithm development is coupled in the reference implementation software to the simulation of the experiment BASF2, with which it is possible to generate tracks for given experiment configurations and data sets. A more accurate estimation of the real or hardware-level performance of the algorithm is achieved by coupling to the HDL-based simulation of the FPGA architecture. At this level, all information relevant for the implementation of the S3D on the FPGA is incorporated.

Extraction of the Architecture

All parameters that are depending on the learning process of the S3D algorithm are stored in the popularly used object structures pickle [94]. It stores the configuration of the hough map in terms of resolution in cells used for each dimension together with the probability of occurrence P_{Hough} for each TS associated with a cell. Within the used design flow

this file is used to instantiate the respective FPGA-based modules that are responsible for the hough map generation on the FPGA. From an implementation point of view, a new tool was developed that extracts the parameters relevant for the hardware implementation and writes these into C++ header files. These header files can then be used for the configuration of the FPGA-based modules with the help of Vivado HLS tools. Using this approach allows for quick and easy adaptation of the architecture for each time that a new pickle file is generated. This allows the implementation of newly trained maps in an automated way, in order to facilitate a quick transition to new firmware versions.

6.4. Configurations for Operation at Belle II

As discussed in section 6.3.1 there are several ways to integrate the S3D into the CDCTRG. The main criterion for deciding which of the options is selected in the end is the feasibility according to the resources available on the selected FPGA. Due to the connectivity and other requirements, the UT4 was already established as the targeted platform to be used. As a result, the targeted FPGAs are of the Ultrascale series. Two choices can be used, that are the VU80 and VU125 as they are both used for the trigger system. This section examines how the NNT could be integrated on the same FPGA.

To provide a realistic estimation of the feasibility of the realization on the target platform, all required modules need to be considered. This includes all of the service and communication modules typically required for operation in the CDCTRG. At present, there is no representative basic setup available for UT4 since most of the components necessary for communication, such as the GTH transceivers, do not have reference implementations. These modules will contribute to an increase in the overall resource utilization and will allow for a more accurate estimation of the achieved timings. As these components are not available at the moment, this investigation will focus entirely on the functional components. The influence on resource consumption due to the missing components will be approximated by taking into account the results of the NNT.

The joint implementation of S3D and NNT allows multiple configurations. Both have several design-time parameters, especially with regards to the scheduling of operations. This is a result of the increased latency budget that is gained by avoiding the otherwise required communication between both components. With regard to the S3D only one instance is required as it is capable of generating several track estimations in parallel. Here especially the clustering component can profit significantly from a higher latency budget when considering its rather high resource consumption. The NNT can also be rescheduled using the additional latency. However, the biggest benefit here is in the usage of parallel instances which can increase the throughput in terms of estimated tracks. The throughput is the one major shortcoming of the current implementation. The optimal configuration is hereby assumed to be at four tracks per system clock cycle, which reflects the current CDCTRG.

However, this investigation is not focusing on the available rescheduling opportunities. The integration effort will rather follow a strategy in which the lowest possible latency is targeted and that is allowing early prototyping. Efforts targeting this optimization option is part of the future work and the move from the prototypical implementation towards

a productive system. As the FPGA to be used was not known beforehand during the creation of the results and the later decision to use the VU125 was largely influenced by the results presented here, the following FPGAs will be considered VU80, VU95, VU125 and VU160. All of these were available for the UT4. Even though the more powerful Ultrascale+ series was available and compatible with the UT4, it is not considered in detail within this investigation. An FPGA of this series would have been the best choice for an integrated NNT and S3D, as it includes plenty of DSPs. However, the advanced manufacturing and decision process underlying the UT4 prevented it from being considered without delaying the production timelines.

UT4 FPGA	Slice LUT s	DSPs	max Networks
VU80	70%	50%	1
VU95	50%	35%	1
VU125	50%	50%	2
VU160	40%	60%/ 80%	3/4
VU190	30%	67%	4
VU5P	50%	35%	4

Table 6.16.: Evaluation of resource utilization for the possible FPGAs targeting an integrated solution hosting both S3D and NNT.

The resource results for the relevant FPGAs are shown in table 6.16 using the low-latency configurations for both S3D and NNT that were shown before. The basic FPGA VU80 is hereby already capable of hosting both one S3D and one instance of the NNT in theory. However, the utilization is already quite high with regard to the Slices. Taking into account that high utilization is influencing the timing, it will probably not be possible to achieve timing closure. Especially when considering that additional required modules are missing at this stage of development. However it shows that DSP utilization is low enough to host at least one instance of the NNT with the overall bottleneck being the utilization of Slices.

Using the VU95 is already alleviating the resource bottleneck considerably. The achieved results show that an integrated solution is within reach of being feasible, with the assumption that resource utilization is between 50 to 60%, will be sufficient to achieve timing closure. When considering several instances of the NNT, the bottleneck is now within the DSP utilization. Considering this only a maximum of two networks is feasible, resulting in 70% utilization. However, this is already a quite high ratio and a heterogeneous architecture for the neural network might be required to achieve timing closure.

The VU125 FPGA provides a massive increase for both Slices and DSP resources. Considering an integrated solution, both are utilized at about 50% when using two parallel NNT. The spare resources might hereby even be sufficient to host additional parallel instances when using heterogeneous architectures. The other optimization strategy here is to exploit the increase in latency to facilitate the reuse of DSPs with a higher degree of multiplexing.

In addition, the results for the VU160, VU190, and VU5P, of the Ultrascale+ series, are shown. The VU5P is the best here, but cannot be used due to the manufacturing cycles. However, it was the state-of-the-art FPGA at that time and provides an outlook into the future of these kinds of trigger systems when using the newest available hardware. Both the VU160 and VU190 provide better characteristics than previous FPGAs, however, the improvements are rather marginal. In addition, they have a large price increase compared to the VU125, which was the most cost-efficient FPGA in terms of processing element per money unit.

With the analysis carried out here, the first indications are shown that combined NNT and S3D solutions are possible with the available FPGAs. On that smallest available FPGA the VU80, an implementation will probably not meet the timing requirements due to the high resource utilization of the Slices. However, this could be addressed by rescheduling internal processing as the investigated architecture is configured to be low latency and does not consider the additional latency budget. With the usage of the VU125, it is however already possible to integrate multiple parallel NNT instances together with the S3D, all while a sufficient number of spare resources are available to safely assume that feasibility can be achieved.

Considering all available options for the UT4, the VU125 is the most suitable FPGA. It is capable of hosting both methods even allowing multiple instances of the NNT. The number of estimated tracks can be increased with additional effort put into rescheduling and balancing of the resource. In addition, it is the most cost-efficient solution. Due to this, the decision was made to use this FPGA for the final productive version of the UT4 that is to be used for the upgrade of the CDCTRG.

6.5. Summary

This chapter presented the design and prototypical implementation of a Hough-based 3D-Track Finding that is aiming at extending the current CDCTRG to improve its capabilities. All considerations were performed with having an integration into the CDCTRG in mind. However, due to the current production time windows, it was not possible to implement such a system on the final hardware platform. The same problem is meanwhile impacting the availability of supporting infrastructure such as the GT IP cores. As such the presented design is only prototypical and representing the current availability of the required resources.

The investigation of the S3D starts with an evaluation of the requirements that are imposed on the system. Here especially the increased amount of required GTs necessitates the usage of newer FPGAs compared to the currently used UT3. Since all of the SLs of the CDC are required to be processed by the S3D, the UT3 used for the NNT cannot provide a sufficient number of GTs. Its succeeding platform the UT4, on the other hand, provides the required number of ports. This property is making it suitable for the implementation of the S3D. Using the more powerful FPGAs available for the UT4 is enabling new possibilities for the integration into the CDCTRG. There are two approaches to integration as a result. The system can be either integrated together with the NNT on one single platform

or separately. The former solution is currently preferred due to its manifold advantages and the current prototypical implementation points towards a highly likely feasibility.

The subsequent section of this chapter is focusing on the FPGA-based realization of the S3D. As the system is without the entire knowledge necessary for final integration, all of the processing modules are designed to be as flexible as possible with the goal of allowing a custom configuration as soon as all the details are available. The entire processing chain is meanwhile including optional processing options that can be used to increase the precision of the implementation at the cost of resources and latency. The options are hereby aiming at refinement of the track parameters representing the currently observed track. The simplest and most resource-efficient solution is the finding of the maximum-weighted Hough cell. However, clustering can improve the estimation additionally. For this, two approaches to clustering were investigated. The unconstrained and the area-constrained clustering. While the former is achieving the best results in terms of the estimation's accuracy, it is not feasible for implementation as both latency and resources are exceeded. The area-constrained clustering is addressing these shortcomings by providing a solution with much lower resources and a fixed lower latency while achieving reasonable accuracy.

The configuration of the complete architecture is meanwhile supported by a Vivado HLS-based framework that has been developed. It is allowing to easily adjust the Hough map parameters and weights of the architecture. Besides the advantages for the implementation, this tooling approach is also used for validation by using HW/SW Co-Simulation.

The developed architecture was subsequently implemented on the basis of the FPGAs supported by the UT4. The achieved results show developed architecture based on a maximum-weighted track estimation can be easily implemented, even on the smallest FPGA, the VU80. Additional refinement based on the area-constrained clustering, without any additional features such as patching is meanwhile as well supported by this FPGA. Enabling these additional optimizations, however, will exceed the resources of the VU80. The targeted biggest supported FPGA, the VU125, is on the other hand capable of hosting the algorithm with all optimizations being enabled. This is already showing the viability of the developed architecture to be used for the realization of the S3D in the future operation of the experiment. These results were however generated without having a unified integration with the NNT in mind. When implementing both systems on the same platform, the VU80 is barely capable of fitting the requested resources. With regard to timing closure, it is hereby highly unlikely that such a configuration will be feasible. The VU125, however, is capable of hosting the S3D together with up to two instances of the NNT. These results were subsequently highly influential towards the decision to mainly use this FPGA for the UT4.

In general, the presented discussion provided an architecture that is realizing the S3D together with ingratiation concepts towards operational readiness. It investigated the available trade-offs between the accuracy of the estimation and resource efficiency as well as low-latency operation in terms of multiple clustering solutions. A good trade-off between both characteristics was found by combining the generation of the hough map with an area-constraint clustering approach. This combination is capable of being integrated together with the NNT on one UT4 that is hosting the VU125 FPGA. As soon as all of the supporting infrastructure is available for the UT4, the investigation can commence to determine the final configuration to be used for the experiment.

7. The Track Segment Finder based on State Machine

The functional principles of the initial TSF that was designed for the CDCTRG were introduced in section 2.2.1.2. The concepts described there were implemented in the firmware that was used during all of the early tests with cosmic rays and first collisions. In subsequent analyses, the efficiency of the detection of TS was investigated and evaluated. It revealed that some parts are having room for improvement for example in its handling of noisy events. Data showed that some events were experiencing a high amount of active TS due to the CDC's susceptibility to noise, which occupied valuable outgoing bandwidth. At other times additionally lower than expected efficiency for finding active TS was observed. It is to note here that even though the TSF is being used during the experiment's operation, it is is still under development. While the problem of high output data rates was addressed by implementing the suppression of adjacent active TS, a new implementation of the TSF logic was proposed in order to solve the remaining problems. This new implementation is based around restructuring its purely combinatorial processing chain toward the usage of state machines. Since the TSF is generating the inputs for all further components of the CDCTRG and thus the NNT, the overall physics performance is increased by improving TSF. The design and implementation of the revision based on state machines is the focus of this chapter in the thesis. The work is based on the master thesis Ref. [Ung18] and expanded by additional optimization such as the selective usage of BRAM that was also presented in Ref. [Ung20].

7.1. Analysis of the Initial Track Segment Finder

The new implementation of the TSF is mostly relying on the reuse of the already established architecture that was developed for the original TSF. Only the logic within the detection of TSs is addressed in this revision. All remaining aspects regarding the integration, interfaces and platform selection remain unchanged. The dissemination of these is the focus of this section.

7.1.1. Integration into the Trigger System

The detection of an active TS can in principle be carried out independently for each possible candidate. The maximum number of candidates that can be examined in parallel is therefore theoretically unlimited. Dependencies for processing are only introduced when additional functionality is required, for example, the suppression of neighbouring candidates. The maximum number of TS to be processed on one FPGA board is limited only

by the throughput of its input sources, the internal structure of the detector readout and the maximum allowed latency. Due to the outline of the detector's readout, it is possible to realize every individual SL separately. For this purpose, exactly one UT3 is allocated for each SL and is dedicated to processing data sent by the corresponding merger units.

One of the keys to implementing such a readout scheme is the used hardware platform and especially the number of provided transceivers in order to receive data from all mergers of a nSL. As the requirements are different across different SL it is sufficient to analyse the SL with the highest processing demand, which is SL8. In addition to this, it is necessary to use the GTH ports in order to achieve the data rates that are necessary to transmit all of the information generated by the high amount of wires processed by each merger unit.

The decision to partition the system this way is determining the number of required GTH lanes per board. Data sent by each merger unit hereby contains information about its status, the wire activity, the timing and drifts within every trigger cell formed by respective wires. Meanwhile, each SL is read out by a variable number of merger units. The reason for this is that each individual SL is containing a different total number of wires to be read out. For example, five merger boards are used for SL0, while 12 merger units are being used for the outermost SL8. Each of these merger units does not send the complete SL rather it is responsible for processing a specified area of the CDC that it is responsible for. The partitioning of a layer into areas processed by one merger unit is illustrated in figure 7.1.

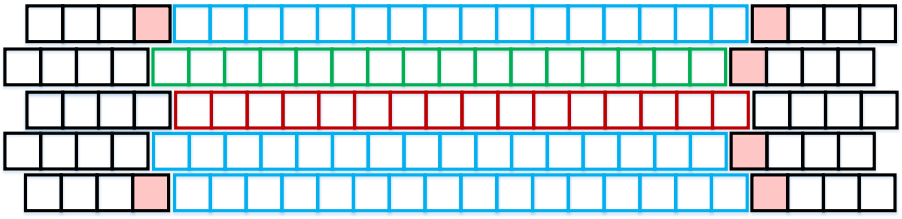


Figure 7.1.: Partitioning of the CDC into segments that are processed by one merger unit. All wire cells that are processed by one merger are coloured. Primary priority wire cells are coloured red, while secondary priority is indicated by green cells. Cells coloured in blue are processed by the merger as well, but has no special positioning information. Neighbouring cells that are coloured black, are processed by different merger units. Here special cells, that are part of neighbouring mergers but necessary to detect a TS are marked in rose.

One of the results of this partitioning is the introduction of artificial boundaries between mergers. These boundaries have to be considered the algorithm of detecting active TS employed on each TSF as active segments can be detected across different merger units. For the TSF, this means that it must process TS candidates at the merger boundaries in a special way. For segments that are located directly at the boundaries of a merger, addi-

tional wire information derived from the adjacent area must be considered. In addition to this, the integration of the TSF is influenced by the output side of the employed readout scheme. Considering the readout scheme described in section 5.2.1 the output data to be sent towards 2DS, NNT and 3DS are distributed across four different hardware boards that are responsible for the different quadrants of the detector.

7.1.2. FPGA Architecture of the Original TSF

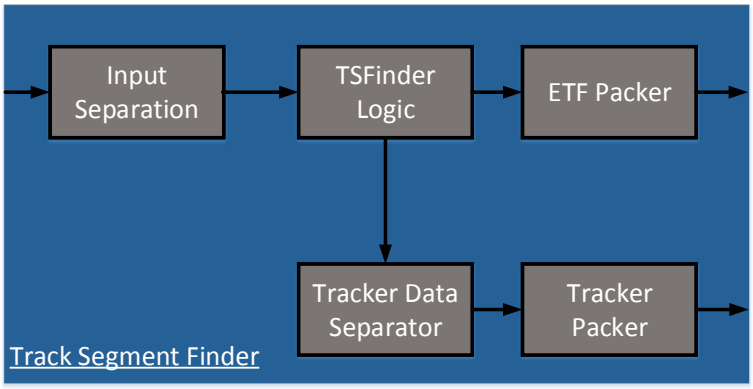


Figure 7.2.: Internal architecture of the original TSF design. The main component to be addressed is the TSFinder Logic module.

The internal processing architecture of the TSF can be divided into several sub-components as shown in figure 7.2 [52, 53]. The core of the entire processing chain is hereby represented by the TSFm module. The task to be fulfilled by this module is to find active TS within a defined area of the respective SL. In principle the area processed by one instance of this module could be used to cover the entire SL, however, the processing is internally distributed on the level of granularity of single merger units, as they have to be processed separately. As a result, the module is instantiated for each supported merger unit separately. Since the number of merger units present in each SL is known in advance, both the interfaces of the module and the number of required TSFm instances are known beforehand and thus statically defined in advance. As a result of this, each merger unit is responsible for processing an area of the CDC that consists of 16 potential TS candidates. This number defines the number of instances to be used for each TSF, for example, SL0 is consisting of 80 TS candidates in total which requires 5 TSFm instances. Meanwhile each TSFm module receives 256 Bit of detector data that is consisting of the hitmap and the priority timing of the specific area of the CDC. In addition to this 18 Bits are received from the neighbouring mergers, which are used to form TS at the boundaries of the designated area.

The combination of this boundary information with the hitmap to be processed is performed by the Input Separation module. The TS information found by each TSFm is functionally divided into two categories. These are the data needed for needed for the track estimation and data needed for the estimation of the event time. The difference between both is mainly in the inclusion of wire timing. Both categories are handled by separated data flows after the TSFm. Data intended for tracking is additionally split by the Tracker Data Separator into packets bound for the different boards that are responsible for the different quadrants of the detector. Formatting and sending tracking data in packets is hereby the task of the Tracker Packer, while data bound for the ETF is packed separately. With regard to the design of the TSFsm, the TSFm module is of particular importance as it represents the core logic. It is thus discussed in more detail in the following. The packers are meanwhile of lower importance as they are basically reused and remain untouched.

7.1.3. TSFm Module

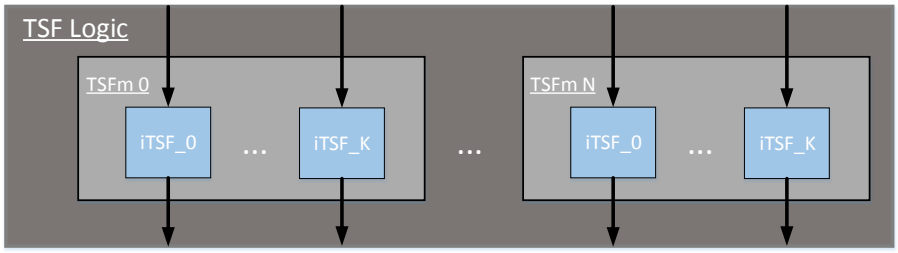


Figure 7.3.: Internal architecture of the TSF logic, consisting of several TSFm instances that are responsible to separate each merger unit to be processed and iTSF instances, which are checking each TS candidate.

Each instance of the TSFm module has the task of determining which of the TS are active at a certain point in time on the granularity of single merger units. For this, each potential TS is examined separately and in parallel. The actual check for activity is carried out inside another internal module, the iTSF. For each segment, this module is checking whether a sufficient number of wires are active according to a pre-defined geometric shape. The resulting overall architecture of the TSF logic is shown in figure 7.3. Each iTSF instance receives all the data associated with the potential TS that is to be checked to determine activity. This data includes the secondary priority information as well as the fastest and priority timing. Based on this information two separate outputs streams are generated. One of these is representing the information required to determine the event time. It contains the fastest time across all of the present wires within a certain TS with a resolution of 9 bit. The other stream is generated to be used by the tracking mechanisms of the CDCTRG. This data stream consists of the 9 bit priority timing, the LR information, and the priority wire type. Both outputs streams differ in that protocol at which they are

send injected into the remaining CDCTRG. Information about the event time is hereby only sent once while tracking information is updated with the occurrence of new wire information from the CDC. These updates are mostly due to the drift time, which can lead to more refined LR information due to the arrival of additional wire hits. For this, it is possible that wire hits will not arrive in time to be sent in the same clock cycle as the other wires as it rather arrives later on.

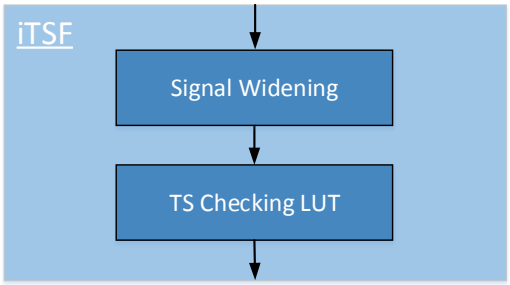


Figure 7.4.: Internal architecture of the iTSF module.

The internal structure of the iTSF module is shown in figure 7.4. At first, incoming CDC data is sent to a signal widening module. This module stores the data for a pre-defined time interval of up to 16 data clock cycles. This mechanism is employed to compensate for different arrival times of separate wire information. The accumulated wire information is then passed on to a LUT, which is responsible for checking whether all criteria are fulfilled in order for the TS to be assigned the active status. The contents of this LUT are defined and filled during design time by estimating the relationship between a track passing through the TS and the internal wires. This estimation is performed by simulating particle tracks passing through the TS. The content can vary over the course of the experiment, as such, they are designed to be easily configurable during design time by using VHDL-based generics. Additionally, the content is depending on the current condition of the actual wires in the CDC. In case that a wire is operating with below its intended performance, the LUT can be trained to reflect this ultimately improving the checking for activity.

7.2. State Machine Approach

After establishing the basics of the initial TSF implementation, the focus is now shifted over towards the state machine based revision, the TSFsm, that is covered within this section.

7.2.1. Functional Description

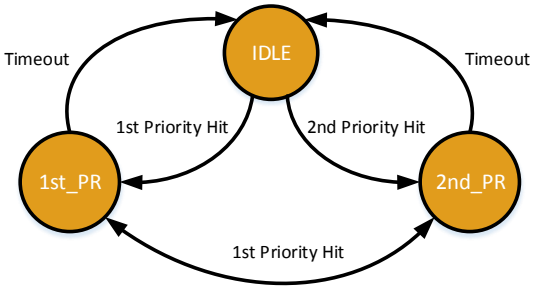


Figure 7.5.: Control flow of the TSFsm represented as a state machine [Ung18].

To improve the original TSF a new version based on state machines [41] was proposed. This new revision is intended to solve the present problems of lacking maintainability and sometimes lower than expected TS detection efficiency. The new approach can be divided into the actual state machine with its transitions and the processing directives within each state. The state machine is hereby shown in figure 7.5. It consists of three states, starting from the idle state, in which no active TS is found within the observed area of the CDC. Depending on whether a first or a second priority hit is observed, the state is changed to indicate activity. Due to the properties of the CDC wire hits can be arriving across several clock cycles. Even when currently being in active states an additional check thus is performed to incorporate recently arriving TSs. In this case, these TSs are added to the detection processing. However, the state is not kept indefinitely. A predefined time interval describes the time for which the active states are maintained. If the internal activity time counter exceeds a pre-defined value, the state machine is reset back to the idle state. For the overall logic of the TSF this means that the previously found TS is no longer active. Subsequent hits are then assumed to be caused by another later occurring event. A special case here is the distinction between the priorities. In case a secondary priority hit was observed at first and a first priority hit afterwards, the overall state is going to change to first priority, which includes a different set of individual operations to be performed as shown in the figure 7.6. The difference between both states is in the protocol used for transmitting the output. First priority hits, will always trigger activity signalling for the respective TS towards further processing steps. Secondary priority hits are meanwhile only sent when at least three additional hits are found in separate wire layers of the TS.

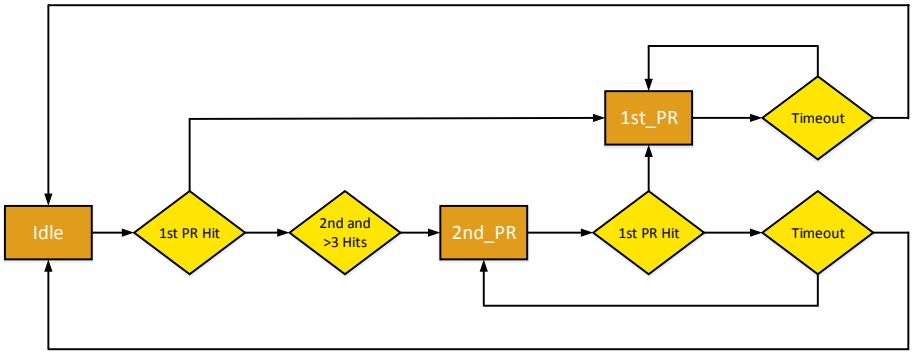


Figure 7.6.: Control flow and branches within the TSFsm processing [Ung18].

7.2.2. Suppression of Neighbouring Active Track Segments

The decision to determine activity separately for each TS, allowed for full parallel processing. However, it introduces an important disadvantage in the presence of high noise within the CDC. The TSF may be quickly reaching its maximum outgoing bandwidth, due to a high amount of TS being active all at once. This is even worsened by the problems of the used UT3 platform, as it fails to achieve its maximum specified bandwidth. As a result of that a saturation of the outgoing data rate was observed several times during the operation of the experiment with collisions. This however had a negative effect on subsequent track finding, as potentially important TS were not send and received.

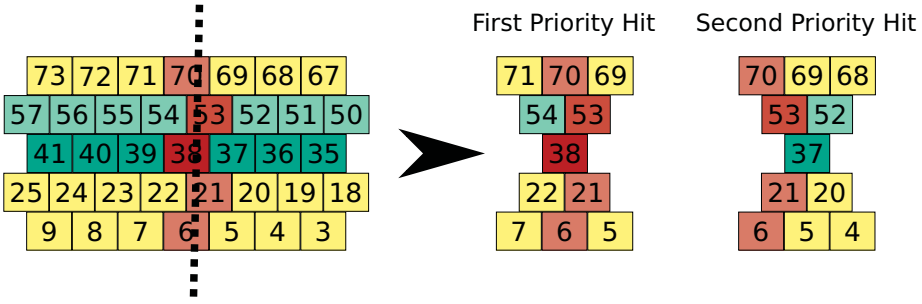


Figure 7.7.: Example for the suppression of neighbouring active TSs [Ung18].

An efficient mechanism that is addressing this problem is the suppression of any potentially unnecessary TS. The main mechanism employed for this is the suppression of neighbouring active segments that are having different priority statuses. An example of this is

illustrated in figure 7.7. In this, a particle's track is estimated by a dotted line. Because the wire cells of neighbouring TS candidates are overlapping, two TS are active at the same time, indicated by the priority wires IDs 38 and 37. Following the general rule for determining activity, both TSs are assigned active state since both are possessing at least four wires that are active in separate layers. However, it is sufficient to send only the TS with priority wire 38, while the adjacent segment can be discarded since it is only of secondary priority. The neighbourhood suppression mechanism is recognizing and solving such situations.

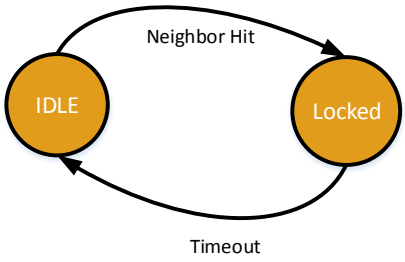


Figure 7.8.: State machine graph showing the neighbour suppression [Ung18].

The implementation of the suppression must still contain a waiting time similar to that of the segments themselves in order to compensate for the different drift times. For this purpose, a state machine was defined that is shown in figure 7.8. The timeout was hereby determined stimulative and set to 16. This additional state machine is included in the FPGA-based processing module responsible of processing TS.

7.2.3. Architecture of the TSFsm

The new TSFsm will only replace the original internal logic for determining the activity of TSs. The remaining components will be retained for the time being, whereby some joint revision with the packer might allow for additional optimization potential. The new architecture is shown in figure 7.9. To make the transition as simple as possible and to be interchangeable with the old TSF, the original interfaces are retained. In the same way, the internal division of processing an SL is retained, as such, each TS is going to be processed by its own iTSFsm instance in parallel. In contrast to the original implementation, however, the previously external additional modules for generating the output are integrated into the state machine as well. Even though this part of the overall processing coupled with the TSF logic, this approach avoids the necessity of implementing another additional module. Each iTSFsm module is meanwhile clocked with the data clock frequency and is receiving hit information for each clock cycle.

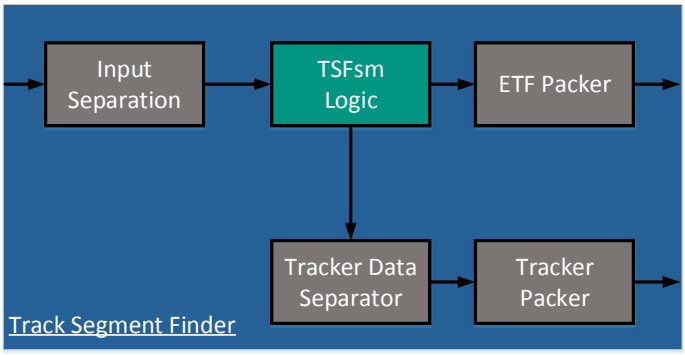


Figure 7.9.: Overall architecture of the revised TSF using the state machine approach.

The new TSFsm will also be equipped with a signal widening to be able to compensate for the different arrival times of the wires. For this, a module is provided which is able to support different time windows. The interfaces of the newly implemented modules are shown in figure A.5 as part of the appendix.

7.2.4. Realization of Left/Right Look Up Tables

The original LR-LUT of the TSF is using a dedicated BRAM for processing each track candidate. For each possible combination of active wires inside of a segment, the LR characteristic is calculated in advance during design time by using a simulation of the detector. The required amount of BRAM resources is then depending on the number of total wires within a segment. For the SL1-8, each segment is made up out of 10 wires. This number is then corresponding to the required address width to be supported by a BRAM. The implementation is treated differently for SL0 as each of its segments is consisting out of 15 wires. As a result, more memory resources are required for SL0. The resource requirements for both types of configurations in terms of BRAMs are shown in table 7.2.4. It can be seen that an LR-LUT used for SL0 is requiring significantly more resources than the other SL. This is one of the reasons for SL0 traditionally being the hardest to implement on a UT3.

Category	SL0	SL1-8
BRAM	16	2

Table 7.1.: Resource consumption of one LR-LUT for all both types of configuration in terms of required BRAMs.

Since such a BRAM has to be instantiated for each TS, the resource consumption of the TSF scales with the number of merger units to be supported. Especially the SL0 is infer-

ring a high resource demand. It is currently implementable, however, it is not possible to use the two main debugging interfaces Chipscope and B2L. The bottleneck for this is at the number of BRAMs required to implement their inherent buffering structures as it exceeds the resource budget. In addition to this, high demand for BRAM is often resulting in timing violations in the implementation. Both of these issues are motiving the investigation of optimization strategies.

Selective Usage of Dual-Port BRAM

The method used within the design of the TSFsm that is addressing the high demand for BRAM is the selective usage of dual-port BRAMs. By configuring BRAM for dual port operations the issue of high resource demand can be resolved. Since the LR-LUT to be stored has the same content for each segment, regardless of its position within the CDC, it can be shared across multiple modules processing the TSs. Using multi-port implementation is then allowing to share the same BRAM.

However, using the dual-port BRAM in parallel means that the LR-LUT has to always remain the same for all of the segments. However, there are use cases in which different LR-LUTs are required. The most prominent example is a segment in which a wire is defective and cannot be used anymore. To compensate such a defect, a specialized LR-LUT can be generated that takes this wire into account. These special cases are then preventing the usage of shared BRAMs, however, they represent a rare exception. In order to provide, an optimized overall solution, with keeping the resources as low as possible, both implementation variants are supported.

For the implementation of selective BRAM usage, all segments are divided into groups of two. These pairs of TS can then potentially share the same BRAM. Additionally, a list of segments with defective wires is defined for each TSF separately at design time. When subsequently instantiating the iTSF, this list is evaluated in order to decide whether a pair of segments can share a BRAM or have to be realized separately in case of the presence of special cases. Assuming that only a small number of segments can be containing defective wires, resource consumption is still significantly reduced compared to the original design.

The specific configuration for using this heterogeneous BRAM architecture is determined at design time. Defective wires are typically detected after a more detailed analysis of the current status of the detector and its efficiency. Subsequently, specialized LUTs are trained, which might then be used from that point on. These only need to be adjusted afterwards in case another defective wire is found in the same merger area. Overall, new LUTs will be rarely generated and loaded, thus they will be in use for a long period of time. A configuration at design time is as a result already sufficient with no advantage by using runtime adaptability.

7.3. Evaluation

7.3.1. Characterization

The new TSFsm that is presented in this thesis was synthesized and implemented on the basis of the UT3 that is hosting an XC6VLX550T FPGA. This is the bigger FPGA compared to the UT3 used for the NNT. For this characterization, TSFs were implemented and evaluated for all of the SLs. The characteristics are additionally compared with the original TSF. Achieved results are listed in table 7.3.1 representing the resource consumption. The resources are hereby divided into three main categories that are LUTs, registers, and BRAM. The LR-LUTs are implemented with BRAMs and are therefore in high demand. This is especially true for SL0, in which the amount of observed TS is the largest due to their inherent pyramid shape.

In all of the presented variants, it can be immediately seen that the TSFsm is achieving lower resource consumption for the LUTs across all layers. The demand for BRAMs is hereby reduced due to the mechanism presented in section 7.2.4. The combination of savings for both resource types leads to more SLs being able to support the two additional interfaces B2L and Chipscope. Supporting these interfaces allows for more sophisticated analysis and debugging.

Category	TSF 0	TSFsm 0	TSF 1	TSFsm 1	TSF 2	TSFsm 2
LUTs %	26	21	30	21	32	22
Registers %	18	20	20	15	20	14
BRAMs	91	46	32	16	29	15
Category	TSF 3	TSFsm 3	TSF 4	TSFsm 4	TSF 5	TSFsm 5
LUTs %	43	31	45	34	51	36
Registers %	28	21	29	23	33	24
BRAMs %	35	18	26	13	26	13
Category	TSF 6	TSFsm 6	TSF 7	TSFsm 7	TSF 8	TSFsm 8
LUTs %	55	39	59	44	63	49
Registers %	36	26	41	30	41	30
BRAMs %	24	12	30	15	27	14

Table 7.2.: Synthesis results achieved for all SLs using the newly developed TSFsm logic.

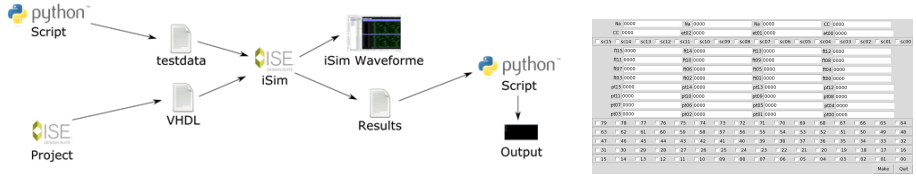
7.3.2. Methodology for Testing

The validation of the TSFsm is very extensive due to its large amount of possible test inputs. In addition to the amount of data to be tested, there are 9 different TSF instances to be tested individually. Each TSF is meanwhile connected to a varying number of merger units for which data has to be collected over multiple clock cycles to imitate the operational behaviour. Considering the overall amount of different test cases to be covered, a

large amount of test data is required to achieve high test coverage with confidence. The problem here is that only a few samples of test data were provided, which only cover a small number of test cases.

To address this shortcoming, two additional test mechanisms are introduced to facilitate more comprehensive testing. The first is a selective test environment that allows specifying all parameters under which a merger is producing its data that is sent to TS. This is used to recreate certain situations as they might occur in the experiment and test them individually. It was for example used to test the inclusion of the neighbour suppression and the correct processing of a merger's corners.

The other test mechanism that was developed aims at achieving large scale testing by covering as many test cases as possible and showing that the TSF delivers the correct results with high statistics. Assuming a uniform distribution for the active wires throughout the CDC, the distribution of found TS should have a uniform distribution as well. For this purpose, a test with randomly generated merger unit data was created. This test used to emulate the input data for the TSF. Both of these test modes are shown together in figure 7.10 in a flow diagram. An additional screenshot of the graphical interface that was developed for the generation of specific test cases is shown here.



(a) (b)

Figure 7.10.: Offline testing flow used for the TSFsm in (a) and the graphical interface used to define special test cased in (b) [Ung18].

7.3.3. Validation

Validation of the TSFsm was performed using both offline tests based on simulation and online tests in which the TSFsm was integrated inside the CDCTRG. The first online tests hereby could only be performed using a merger play setup since the detector itself was in a maintenance cycle in which it was prepared for usage for phase 3 of the experiment. This means that early tests were using predefined input patterns that are considered to cover the most critical functional paths of the CDCTRG. The result is that wire activity as it would be produced in the presence of noise is not taken account. This is mostly an issue for testing of the neighbour suppression as it is designed to reduce noise.

For comparison to the original design, both variants, the original TSF and the new implementation were tested. Since this is difficult with cosmic ray operation as the input data is not known in advance, a merger play test setup with a predefined event is preferable. This

event was generated via TSIM and used to validate the original TSF. This event includes multiple active TS which IDs are known in advance.

The results of this test are shown in figure 7.10 as part of the Appendix. The detection of an active TS is hereby represented by a peak in the signal diagram for the respective bit value. Both variants are able to recognize the active TS, showing the correct operation of the new implementation. However, in the original variant, a second output is generated in which the LR information is updated.

While representing the best approach to test a trigger component with consideration of all integration aspects, the merger play test is rather unsuitable for achieving high test coverage due to the limitation of using only selected single events. The random number based test is more suitable for this purpose. Using this test, both the new TSFsm and the original design were evaluated regarding their abilities to detect both first and second priority TS. Since the implementation across all TSFsm is functionally based on the TSFm module, which is responsible for covering one merger unit, it is sufficient to test only this module to achieve representative results. Figure 7.11 hereby shows the results for using random input data, with the y-Axis showing the percentage of found TS for a certain type of priority and the x-Axis showing the respective TS that was found in this area. For the proof of correct functionality, a uniform distribution is expected across all of the possible TSs. This characteristic can be observed in both variants and is indicating correct functional behaviour. An additional comparison in absolute numbers is shown in table A.7.

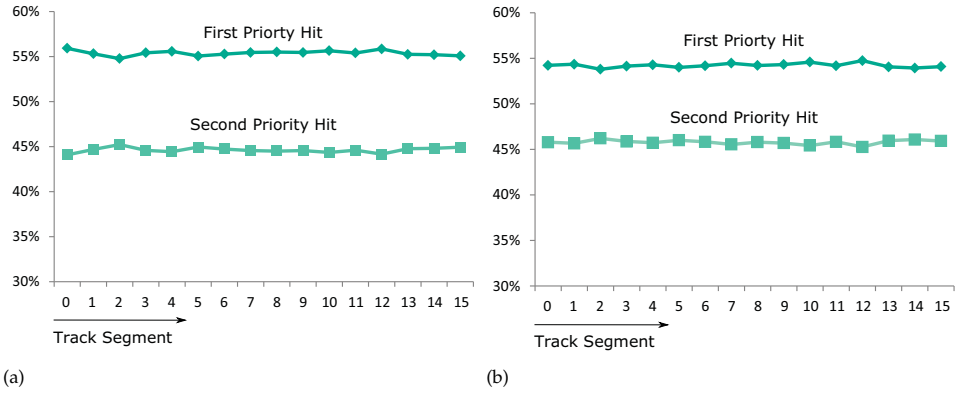


Figure 7.11.: Histogram of found TS using random event generation for both the original TSF (a) and new TSFsm (b) [Ung18].

In addition to the general functionality of the TSFsm, tests were conducted to show the correctness of the neighbour suppression logic. The same random test input generation approach was used for this, with different levels of occupation within the CDC. The results of these tests are shown in figure 7.12. Here, the y-Axis is showing the total number of found TS for different occupations in the CDC that are represented as percentages on the x-Axis. With low occupation, most of the TS that are active and not suppressed. This increases with higher occupation as more TSs are suppressed in order to save the output

bandwidth. In addition to showing general correctness regarding the intended functionality, it also shows that the new TSFsm logic coupled with the neighbour suppression is generating fewer TSs which is consistent with the expected results.

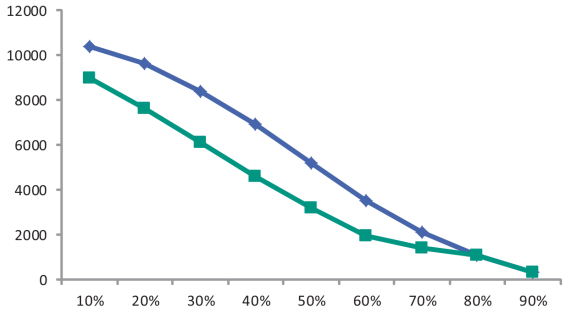


Figure 7.12.: Validation of TSFsm neighbour suppression logic [Ung18]. The blue line is showing the number of found TS for the original design while green is representing the TSFsm.

7.3.4. Results from Tests within the CDCTRG

In addition to a purely simulative validation using the presented test methods and a reduced operational validation using the merger play setup, the TSFsm was evaluated while being integrated into the CDCTRG with a connected CDC. An excerpt from the observed behaviour is discussed in this section. The efficiency is hereby used as the main metric for the evaluation. It is describing the ratio of events of a specific type for example "-fff", which means more than two 2D tracks were found [83], that are expected to be found by analysing the data and events actually found by the CDCTRG.

The achieved efficiencies are hereby shown in figure 7.13 for several runs. It is additionally partitioned into phase 2 and phase 3, with the difference being that the TSFsm was in use during phase 3 while the original TSF was being used during phase 2. Immediately it is observable that efficiency is never reaching 100% showing that some inefficiency is always present. The exact reasons for this are currently still under investigation. The benefit of using the TSFsm is shown when comparing it to its predecessor. Across all investigated runs, the original TSF was achieving an about 10% lower efficiency. This shows that TSFsm is not only contributing to finding TS more efficiently but events themselves, which are even more important.

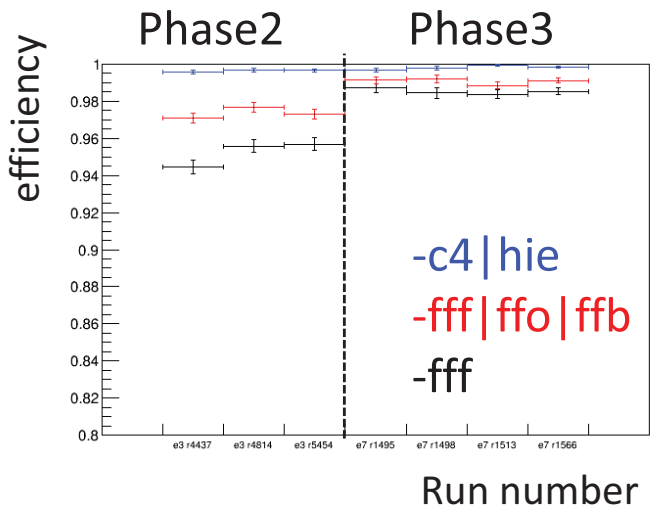


Figure 7.13.: Comparison between the efficiencies for hadronB skim during phase2 using the old TSF and phase3 using the new TSFsm [58].

7.4. Summary

The TSF is the first stage of data processing of the CDCTRG, all other more complex trigger components such as the NNT are based on its capability to efficiently and reliably provide TS. In the early phases of the experiment, a first version of the TSF logic was integrated. This version was continuously updated and at some point lost maintainability in addition to the loss of detector efficiency. Since this efficiency is critical for the entire trigger, optimizations are necessary that include a structural revision of the core logic. The proposed approach for this update of the TSF is to convert the mostly combinatorial design of the original realization to a state machine-based design. Using the original TSF as the starting point the TSFsm was developed. The new design is based on the architectural patterns for state machines as it is recommended by Xilinx, the FPGA's manufacturer [118].

The resulting implementation itself is consistently achieving lower resources demand compared to the original. Further optimization of the resources is achieved by addressing the implementation strategy for look-up tables that are used to determine the LR information of a TS. The followed strategy is hereby based on using BRAMs with selective dual port configuration instead of the previous single port solution that was used for all LUTs. In order to take into account special situations that might be occurring within the CDC, e.g. defective wires, an option was added that allows instantiation as a single port at design time to load special LUTs which can achieve better performance.

Additionally, tools were developed to assist with the usage of the new TSF. These tools enable the generation of specific test patterns, which can be used to emulate certain events

in order to check the logic in detail for correctness. At the same time, TS suppression based on neighbours was added to the core logic to achieve more efficient usage of the limited output bandwidth. The new implementation was validated with the help of randomly generated data, the merger play test and early experiment operation. In all variants the correct function of the approach was proven, even showing that track finding efficiency of the subsequent 2DS can be improved.

8. The Online Cluster Analysis

8.1. Online Cluster Analysis for Rescuing Slow Hadrons

The online cluster analysis represents the application case within this thesis for machine learning-based online data reduction using hardware realization. It is aiming at showing the viability of such algorithms to be used in real-time on FPGAs to efficiently identify particles of a certain type by using the experiment's pixel detector. The following sections are discussing its role within the experiment and the functional principles.

8.1.1. Experimental Context

The main method for online data reduction of PXD data is the RoI approach presented in section 2.3.1. While it is highly efficient, it has the drawback that at least three layers of the SVD have to be reached in order for a particle to be considered. Most particles are reaching the required number of layers, however, a fraction of the possible decays can result in particles not reaching the required number of layers due to low momentum. However such particles can still be of high importance to the physics experiment, the most important example for this are slow pions. To retain detector data generated by these particles, an alternative mechanism to the RoI is required. Architecturally the idea is to add an additional particle identification mechanism in parallel to the current one. This additional mechanism shall be capable of correctly identifying such particles using PXD data only.

In Belle II, the SVD is used for this kind of data reduction. The extrapolation is based on track data generated from several successive layers of the detector. Here the probability of the random background throughout these layers of the detector is very small, making the SVD a good candidate to be the basis for the definition of RoIs. An approach based on the SVD, however, has the disadvantage that a track has to be found in order to extrapolate it to the PXD. This is only possible if a sufficient number of layers of the detector have been reached by a particle. The necessary number of layers is determined by the tracking algorithm of the SVD [16] [96]. Particles that do not hit enough layers are meanwhile suppressed.

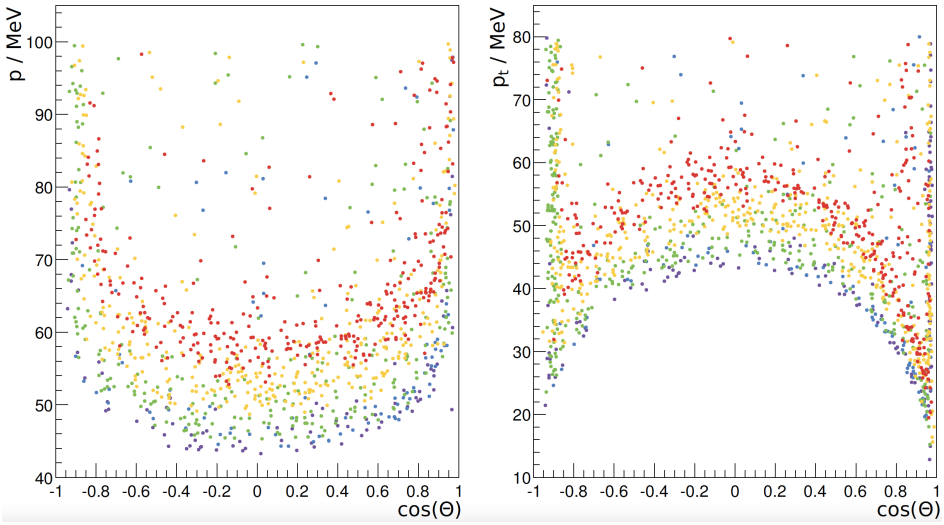


Figure 8.1.: Range of slow pions in the form of SVD layers reached depending on pt [93].

A candidate for particles that are potentially suppressed, but are important for the experiment, are pions. Their reach within the SVD is shown in figure 8.1. Here a momentum of 60 *MeV* may already not be sufficient to reach enough layers of the SVD for tracking. As a result pions with or below this momentum cannot be used for the RoI approach.

The question arises as to how important these particles are for the actual experiment and thus also whether one can live with their loss. Considering the important B->D*->pi D^0 decay they appear quite often as shown in the red superimposed histogram figure 8.2, which is a result of conducted studies in Ref. [93]. In addition, entries are plotted with respect to their momentum. Since D* and D^0 have only a small mass difference, the resulting pion can only inhibit a low momentum. However, these will not reach the first three SVD layers, thus losing corresponding data from the PXD. As a result, the probability that such a decay will subsequently be correctly reconstructed is decreasing significantly.

Several possible algorithms were investigated for the identification of low momentum particles. The NeuroBayes algorithm is hereby providing the best performance in terms of achievable classification, even when being confronted with the most significant parts of the anticipated background, QED and Toushek, which make up an estimated 70% of the total background events. A characterization of the capability of identifying particles using a trained NeuroBayes network is presented in figure 8.3. It shows the distribution of the network's output which is representing the probability of a cluster of pixels located at the PXD either being related to signal, desired particles, or background. An output value of 1 represents the highest estimated probability of a cluster being signal while -1 is representing the highest estimated probability of it belonging to background events. The real assignment of data to one of the two groups is known beforehand in this case and indicated by the colouring of its bin. Red bins are representing signal, while background

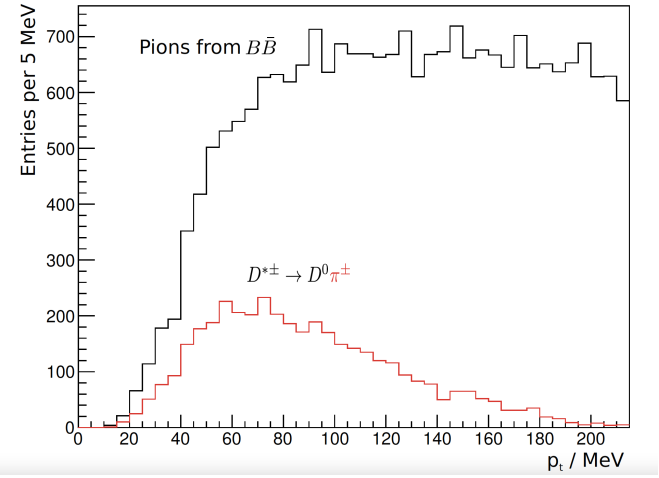


Figure 8.2.: Contribution of pions in D* decays [93].

is highlighted in black. By studying the plot it is obvious that the algorithm is capable of separating both classes from each other, as the majority of estimations for both classes are trending towards the opposite side of the output x-Axis. Additionally, both classes are mostly correctly classified, with only a few outliers being present.

This plot is however just generally showing that the algorithm is performing well. A more quantified view is provided in figure 8.4. Here, the signal efficiency and background efficiency are facing each other for a used data sample. To have more insight into the nature of the classified particles, they are plotted separately for different momenta, which are colour coded. Especially low momentum particles are of interest as these are the ones that will be suppressed by the RoI approach. For the target background rejection ratio of 90%, a signal efficiency of 95% can be achieved even for the group of particles possessing the lowest momentum. This is showing the effectiveness of the algorithm.

8.1.2. Functional Description of the OCA

For an effective design of a data reduction system, it is first necessary to study the footprints of the probable particles within the pixel detector. When a particle passes the PXD, it is often not only effecting a single pixel but is rather interacting with several neighbouring pixels. This results in a cluster of active pixels caused by the same particle. These clusters form the basis for all processing within the OCA. Data that is essential to identifying the observed particle can be classified into the three groups of charge, spatial and detector-related information. Spatial information describes, for example, the size of the produced cluster in two dimensions. Meanwhile, information about a particle's momentum is derived by making use of the DEPFET sensor's capability to record the deposited charge with a fine resolution. These two classes form the most significant information nec-

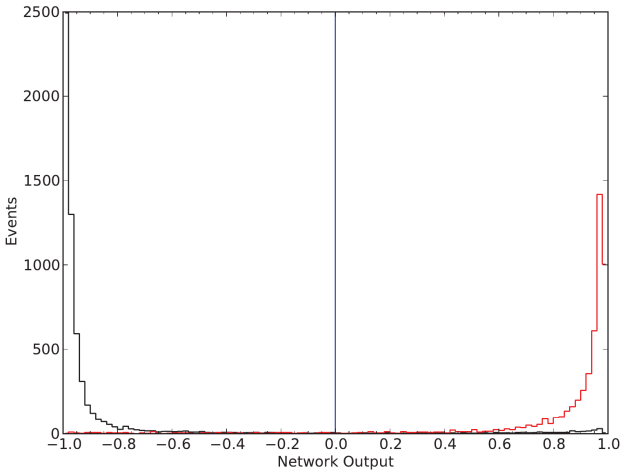


Figure 8.3.: Distribution of the cluster classification based on the NeuroBayes algorithm [93].

essary for particle identification. In addition to them, the position of the cluster relative to the detector is taken into account in the form of the respective ladder, at which the cluster is recorded. The significance of considering the deposited charge is apparent when studying the expected background events. These events are predicted to only deposit a small amount of charge in pixel sensors compared to signal events. Especially the targeted pions with a momentum of lower than 100 MeV are expected to generate much higher charge values in the sensors. The difference is so significant that the first attempts of realizing an OCA were solely based on applying a threshold on the recorded charge. Background particles are additionally expected to produce pixel clusters with significantly different shapes. Preliminary studies showed that for example, Touschek background will lead to long drawn-out clusters with many members, while clusters produced by pions are having fewer members and more uniform shapes. Based on these considerations, 9 separate characteristics of a cluster are used for the particle identification. These are corresponding to the inputs for the neural network that is used later on for the classification. An overview of the characteristics is provided in table 8.1.

Based on the considerations about pixel clusters, the general functional approach of the OCA is to calculate these characteristics out of pixel detector data and then pass them to a neural network that was trained for the identification of particle types. The network was trained as part of Ref. [93]. The definition of physics signals is represented by slow pions but could be extended such as described in Ref. [116]. Targeted slow pions are assumed to have small momentum such that they are most likely not reaching the outer layers of the SVD, thus potentially being suppressed by the RoI.

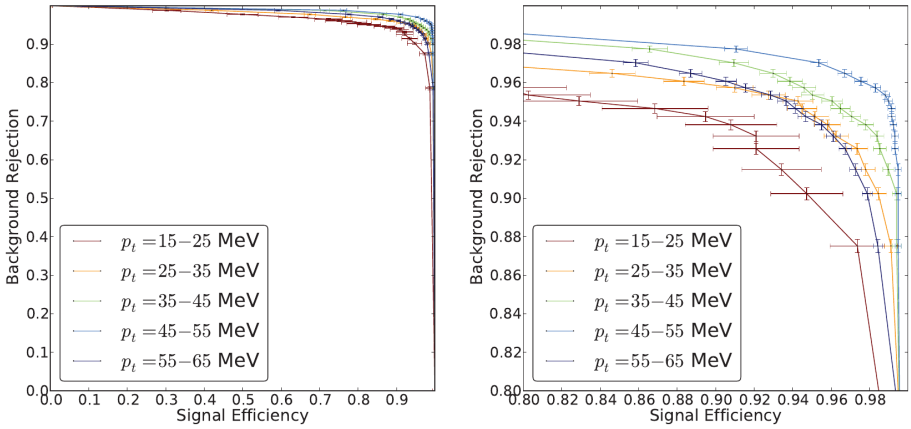


Figure 8.4.: Efficiency-Rejection of the NeuroBayes algorithm for classification of particles using PXD data. It is plotted for different momenta, with low momenta particles having the highest significance for the experiment [93].

Input name	Description
Total Charge	Summed charge of all pixels in cluster
Max Charge	Maximum charge across all pixels in the cluster
Min Charge	Minimum charge across all pixels in the cluster
Variance Charge	Variance of the charge for all pixels
Members	Number of pixels in the cluster
Length r	Maximum spread of the cluster along rho
Length z	Maximum spread of the cluster along z
Ladder	Position of the pixel cluster

Table 8.1.: Characteristics of pixel clusters that are used for the OCA.

8.1.3. Requirements

Being an instrument that is used to reduce the outgoing data rate online, the requirements for the OCA are set by the parameter of the DAQ that was designed to be used together with the experiment's VXD. In this section, the focus is put on the major characteristics to be achieved represented by the throughput and required background suppression.

The minimum throughput to be achieved by the OCA is dictated by the maximum amount of data to be read out from the PXD for one event. This value is determined according to equation 8.1. Besides the amount of data required to transport the information of one pixel, both the maximum occupancy and the total amount of pixels in the detector major

contributing factors. The occupancy is hereby describing the fraction of pixels active for a given event. The maximum occupancy is meanwhile estimated by simulation due to the lack of real experiment data. With even the most conservative estimation, that is likely overestimating the real fraction, the maximum occupancy is excepted to be at around 3%.

$$\begin{aligned} DataRateMax_{PXD} &= PixelCount_{PXD} \cdot OccupancyMax_{PXD} \cdot Bytes_{pixel} \\ &= 8 \cdot 10^6 \cdot 0.03 \cdot 4 \frac{Byte}{Event} \\ &= 937.5 \frac{kByte}{Event} \end{aligned} \tag{8.1}$$

The maximum allowed data rate for transporting PXD data through the DAQ is per design set to a fixed amount of $100 \frac{kByte}{Event}$. In nation with the maximum occurring data rate for an event, the reduction factor to be achieved can be determined by using equation 8.2. Thus a reduction factor of around 10% has to be achieved to not exceed the available bandwidth.

$$\begin{aligned} DataRateReductionPXD &= \frac{DataRateMax_{PXD}}{DataRateDAQ_{PXD}} \\ &= \frac{100}{937.5} = 0.11 \end{aligned} \tag{8.2}$$

8.2. Realization of the OCA based on the NeuroBayes Algorithm

8.2.1. Integration into the DAQ of the PXD

Compared to the previous systems such as the NNT, there are much fewer degrees of freedom in terms of integration possibilities into the DAQ with the PXD as it was an already established system when the OCA was proposed. As such there are no opportunities in terms of using new hardware, all integration efforts have rather be made in accordance to the present system. Most of the considerations presented here are based on the publication Ref. [Bae15] by this thesis's author.

Selection of the Platform

Considering the DAQ's architecture there are only two viable locations at which the OCA can be integrated. These are at the DHH or the ONSEN. The former is located earlier in the data processing chain and mainly tasked with combining pixel data from several different ladders. Additionally, it hosts the clustering algorithms that combine neighbouring active pixels. These clusters are hereby representing the input data basis for the OCA. Even though these tasks are requiring significant portions of the underlying FPGA's resources, around 50% of the logic resources are still available to be used [69]. The ONSEN, on the other hand, is located directly after the DHH and is thus much later in the processing chain. While representing a viable host for the OCA, the chances for integration at this point are very slim, due to the high occupation of logic resources by its vital functions.

Around 90% of the logic resources are already in use, basically rendering integration of the OCA impossible. For this reason, the DHH is chosen as the targeted hosting platform. An architectural view of an OCA integration into the DHH is shown in figure 8.5.

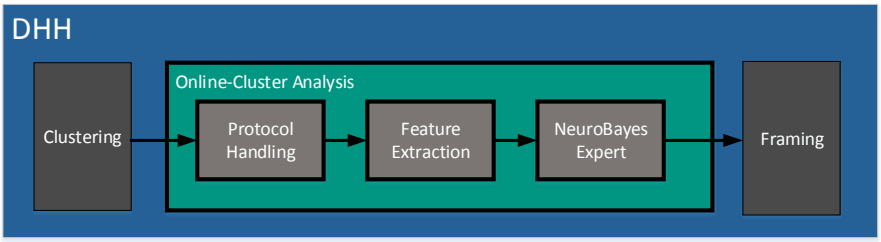


Figure 8.5.: Overall processing architecture used for the OCA. The modules Clustering and Framing are expected to be provided at the DHH.

Handling of Interfaces

The input values for the OCA are arriving in the form of pre-calculated clusters consisting out of neighbouring active pixels. On the DHH these are passed on in their own data format and protocol as is defined within the DAQ of the VXD. This data has to be both unpacked and subsequently processed in order to be usable for the OCA.

The clustering engine of the DHH is providing up to two pixels that are part of a cluster at each clock cycle. In the case that larger clusters are transmitted, several clock cycles are required. As mentioned previously clusters that are belonging to signal events have a rather small footprint. To limit the maximum latency and resource consumption, the maximum cluster size supported by the OCA is already defined in advance and set to 16 pixels. This number is derived from studies of simulated events in the PXD. These showed that clusters produced by signal have at most 16 members, with even more members pointing towards background events. This means that a maximum amount of 8 clocks are required to accept a single cluster. At the same time, pixels belonging to different clusters can arrive at the same clock cycle. These are treated separately, with a new cluster being buffered and the old one being forwarded towards further processing. The calculation of the most input values is computationally simple. The charge-based values are mostly relying on comparisons or accumulation of the individual pixel values. Spatial information is meanwhile indicated in each pixel of the cluster, as dedicated flags are used to represent the increase in the direction of phi or z. Both charge and spatial data can be processed at each clock cycle, for example, a partial sum is calculated even though not all pixels were received.

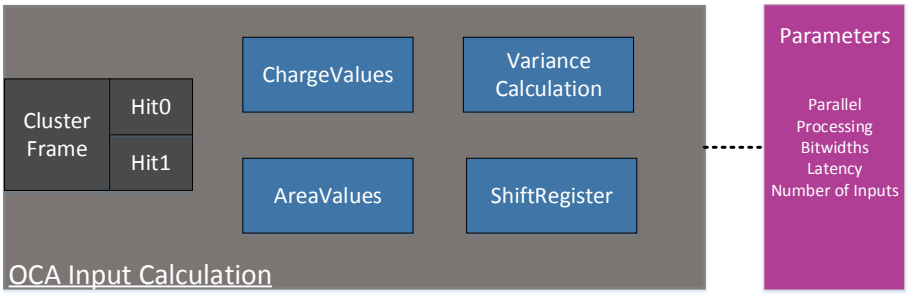


Figure 8.6.: Architecture of the calculation of inputs for the OCA using cluster frames.

The computationally most complex value to be calculated is the variance of the charge. However, by constraining the maximum supported size for a cluster, it can be greatly simplified by implementing the required division as a multiplication with a constant. Since the variance can only be calculated after the arrival of the entire cluster, it is at the end after all members arrived. This results in the architecture shown in figure 8.6 for the input calculation.

8.2.2. Architecture of the OCA on FPGA

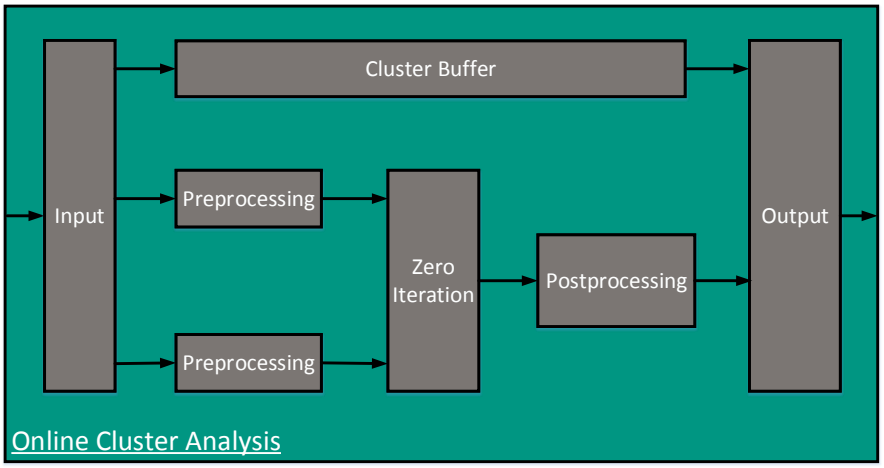


Figure 8.7.: Complete FPGA architecture for the NeuroBayes algorithm based on the zero iteration.

The developed architecture used for the realization of the OCA is shown in figure 8.7. First, the incoming clusters are buffered and decoded within. The now decoded values are then used to generate the defined input values to be used for the prediction algorithm. In this case, nine parallel data streams are generated each representing the respective input signal.

Parallel to these data streams, the received cluster data is written untouched into a cluster buffer. These raw clusters are delayed by the same number of clock cycles as the latency required for the processing of the algorithm. Since there is no clock domain crossing and maximum throughput requires to accept a cluster at each clock cycle this buffer is implemented as a shift register. The reason for keeping the cluster data unchanged is that it shall be passed untouched to the next stage. The OCA is only setting a flag that represents the prediction result, which is the classification into background or signal. This flag is added at the output stage, which is additionally implementing the protocol for transferring processed clusters to the next stage.

The algorithm itself consists of three processing stages, which can be interchanged with different components depending on the current use case. The first stage is representing the preprocessing of the algorithm, in which the received input values are transformed into a representation suitable for the solution algorithm. In this version of the algorithm, all of these input values are processed independently of each other. This is indicated by parallel instances in the architectural view. The subsequent zero iteration algorithm is then combining all processed data streams into one final output value. This value is then processed in the final stage, the postprocessing. Here the membership for the cluster is determined and represented as a binary decision. The following sections will focus on the design and characterization of each individual component used for the OCA.

8.2.2.1. Preprocessing

The preprocessing of the input data for the NeuroBayes algorithm is realized either by using a two-dimensional correlation function or a Cumulative distribution function (CDF). For the use case of the OCA, using a CDF is already sufficiently accurate for predicting clusters. In addition to this, it is less complex to implement as it requires less processing operations. As a result, only the CDF is considered in this design in detail but could be swapped with another method in future operation. The principal internal processing architecture of the module that was designed for realizing the CDF is shown in figure 8.8. In principle, it can be divided into finding the suitable bin followed up by an optional interpolation to increase the accuracy. The design characteristics of the interpolation hereby strongly depend on the used bit width or maximum possible value of the input value to be processed.

The design is hereby differentiated into two separate cases. In the first case, the assumed bit width is less than 6 bit. In the context of the OCA, this is the case for all of the area-based input values. In this case, all of the individual bins can be directly mapped onto the input values, since the number of bins used within the algorithm is a constant set to 100. Since each input value can be mapped exactly to one bin, no interpolation is necessary. However, as soon as the input value range is exceeding the 100 bins, an additional interpolation has to be applied to addresses that are within a bin's boundaries.

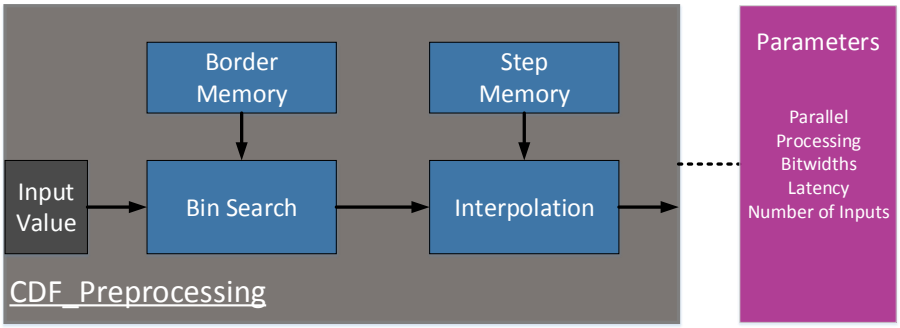


Figure 8.8.: Architecture used for the realization of the CDF within the OCA.

For even bigger input value ranges, the functionality hereby no longer changes, however it has an impact on the choice of implementation used for the interpolation. For smaller bit widths, joint implementation of binning and interpolation using a LUT is representing the most efficient solution in terms of accuracy and resource demand. However, this changes as soon as the bit width is increased since resource demand is not scaling well. In this case, an alternative approach is used. Focusing on the requirements of the OCA, however, shows that the defined input value ranges are all within the limits at which an implementation as a LUT is still reasonable. Since the base algorithm can change over time, due to the changing behaviour of the experiment, an alternative solution for interpolation with large input value ranges was designed.

Architecturally this alternative solution is different from the LUT-based approach. Here, the finding of the correct bin and the interpolation are designed separately from each other as individual modules. Finding a bin consists of two parts as well. It consists of a memory that contains the bin boundaries that are defined during design time for an input value. Subsequently, an additional processing step is necessary to look up the value assigned to this bin.

Based on the found bin, the value can be then interpolated. A linear interpolation is used for this to approximate the value to be used for the solution algorithm. The interpolation is hereby implemented by using a memory that stores a constant offset for each value that lies within the bin's boundaries. The offset is then multiplied with the difference between the starting boundary of the bin and the input value itself.

The architecture used for implementing the CDF is presented in figure 8.8. It represents the most complex case when using interpolation with large input value ranges. The solution for small ranges is mostly the same, however, it is omitting the explicit interpolation that is otherwise implemented in a separate module. The discusses architecture was configured for the use case of the OCA and implemented on the basis of the DHH. The overall results are shown in table 8.2. All of the characteristics are sufficient allowing implementation and fulfilling the set requirements, since the resource demand is rather low while

the frequency is well above the targeted input clock frequency of 200 MHz as 322 MHz are achieved.

Category	LUTs total	Registers	Frequency	Latency
Preprocessing on DHH	1 890	181	322	4

Table 8.2.: Synthesis results for the complete preprocessing of the OCA.

8.2.2.2. Solution Algorithm

The algorithm selected for the OCA zero iteration variant of the NeuroBayes algorithm. Its advantage compared to a full neural network is that it can achieve very good estimation results while using only one single neuron. This significantly reduces the resource demand for the FPGA. This advantage is somewhat bought at the expense of a more complex preprocessing, which introduces additional latency. This additional latency is, however, tolerable for the use case of the OCA.

For the use case of the OCA, low-latency is only a secondary objective. The most important criterion is the achievable throughput. In the considerations from section 4.4.1, an architecture was designed in which neurons are designed to achieve a maximum processing clock frequency using a high degree of pipelining. In this architecture, DSPs are cascaded with the big advantage being that specially dedicated communication channels between them are used to transfer the common data as fast as possible. This architecture is used for the OCA, with the different cluster-based input values representing the cascaded data to be transmitted. The complete architecture is not explicitly shown as it is mostly resembling the one shown in figure 4.4.

A remaining question when using this architecture is the selection of the bit width. Essentially, the question here is whether the zero iteration with the limitations of the DSPs is still capable of achieving sufficiently good estimation results. For this purpose, analyses were carried out to determine the influence of the bit width on the overall result. For this, the algorithm was performed with a set of different constrained bit widths. The results of these were subsequently compared with the reference solution modelled in SW that is using unconstrained data types. It shows the standard deviation of the results between SW and HW for different bit widths. Complete matching can be achieved with a 25 bits width for both weights and inputs. The optimal configuration of $25 \cdot 18$ is able to achieve a deviation of $1.5 \cdot 10^{-}5$ which is negligible. The results of this investigation are shown in more detail in figure A.2 as part of the appendix.

Overall, the zero iteration algorithm behaves quite well when reducing the bit width. Even at very low widths, the deviations are still rather small. The optimal configuration from the hardware's point of view is still at a bit widths of 18 and 25. For this configuration only very small differences at $O(10^6)$ are observed, which are negligible.

In addition to the investigation of the influence of the bit width on the achieved accuracy of the design, it was implemented to determine the resource consumption. The achieved results are listed in table 8.3. The generated results are all very promising, with generally

low resource consumption and a maximum clock frequency of up to 367 MHz, which is well within the targeted frequency.

Category	LUTs total	DSPs total	Registers	Frequency	Latency
OCA on DHH	565	9	367	303	11

Table 8.3.: Synthesis results for the solution algorithm of the OCA.

8.2.2.3. Postprocessing

In the post-processing processing step, the calculated value of the zero iteration is first transformed again using a CDF. The same implementation is hereby used as in section 8.2.2.1. The value is then passed onto the activation function. For the OCA the sigmoid function is used. As with the *tanh* in the case of the NNT, the calculation of the sigmoid function is rather complex and is thus implemented using a LUT. Since the value range is quite large this time, the resource consumption of this implementation would be rather high. To avoid this, the function is stepwise linearised, as discussed in section 4.4.2. The output of the activation function is finally binarized using a configurable threshold filter. The output is either 1 or -1 indicating whether a cluster should be kept or not.

8.2.2.4. Configuration of the OCA for Belle II

The previously individually presented modules are in combination forming the complete OCA design. The results derived from implementing the complete architecture on the basis of the DHH, which is hosting a Virtex-6 VLX130T-2 are listed in table 8.4. Overall all requirements can be fulfilled with the complete architecture. Both LUTs and DSPs are well below the set goals of 30% for LUTs and 50% for the DSPs as they are at around 2% for LUTs and 3% for DSPs. Additionally, the achieved throughput is well above the minimum goal of 200 MHz output rate, as it is reaching up to 303 MHz.

Category	LUTs total	DSPs total	Registers	Frequency	Latency
OCA on DHH	2 455	9	548	303	15

Table 8.4.: Implementation results for the complete OCA.

8.2.3. Design Flow for the OCA

Inspired by the design flow concept described in section 4.3, a semi-automated framework was developed for the OCA. It is meanwhile based on the publication Ref. [Bae16] by this thesis's author. In contrast to the concepts described there, some differences have to be

considered for this use case. As this is the first implementation of the zero iteration algorithm based on FPGAs, no analyses that were investigating the influence of fixed point processing on the estimation performance of the algorithm were available. The other big difference lies in the software environment of the NeuroBayes. For the development, a Python-based implementation of the zero iteration was provided. In addition to the implementation of the algorithm, an environment for verification was provided, which is based on unit tests using already available libraries.

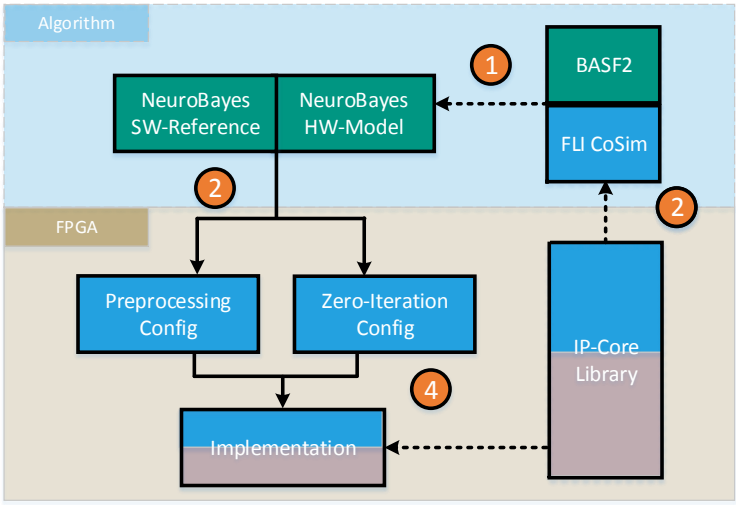


Figure 8.9.: Developed design flow for the generation of OCA firmware.

Since there was no preparatory work on the analysis of fixed-point processing, a design space exploration is included in the developed framework. The aim of this is to investigate the influence of different bit width configurations as was shown in section 8.2.2.2. Design exploration typically suffers from very long processing times due to the massive amount of free parameters to be considered in the hardware-based design. However, considering the time at which the OCA has to be operational, a combinatorial exploration can be used. In addition, in case a new network is to be created, there will always be long time windows until it can be loaded into the operational environment. In addition, the parameter space to be investigated is rather small due to the efficiency of the zero iteration. Due to these reasons, the analysis of fixed-point processing is implemented in the framework as a combinatorial exploration in which all possible configurations are tested. When transitioning the general NeuroBayes design to other use cases, a redesign of the design space exploration might be necessary to avoid excessive design times. The overall design flow used for the OCA is shown in figure 8.9.

Realization

The core of the framework is the integration of the Python-based implementation of the zero iteration algorithm together with the HDL-focused implementation. Several frameworks are meanwhile already available that allow a description of such HW-level models in python. They are even capable of allowing conversion of the model directly into an HDL description. However, not all language constructs are supported which makes their usability rather limited [24]. The recommended description of functionality according to the user guides of such frameworks are basically representing the same level of abstraction as Verilog or VHDL. However, the advantages of HDLs such as the support of IP core libraries and tool directives for optimization, are not supported.

For this reason, such frameworks are not used in the design flow for the generation of the firmware. The approach pursued here is to continue the description of processing modules in an HDL thus relying on tools and libraries for optimization. The behaviour of the module is meanwhile simulated in python as a model. This python model is then validated against reference software and HDL implementation using the existing unit test infrastructure. The parameters of this model are then extracted and used to configure the HDL module, allowing to use less computationally intensive design space exploration. The validation is meanwhile performed with HW/SW co-simulation in an automated way. Only for the parameter selection, intervention is still necessary, since compromises must be made between the different performance and resource characteristics achieved by the investigated configurations.

Native python does not support variable bit widths, which is required in order to model processing on the FPGA. However, this functionality can be added by using external libraries. The best solution for this is provided by the myHDL library. While it aims at creating HDL descriptions within python, it also provides types with constrained bit widths that can be used in general programs. The use of this library also has the advantage that existing unit tests can be reused for validation. The validation of the HDL modules against their python models was meanwhile realized with the help of the tool Modelsim [73]. This tool provides several possibilities for coupling external software to an HDL simulation. One such possibility is the Foreign Language Interface (FLI) [76]. It has the advantage of already being supported by myHDL, so that additional effort is kept to a minimum for the implementation.

Using models of FPGA processing can in parallel be used to make rough estimations about the resource demand at an early design stage. This ability is implemented by calculating the characteristics of the available HDL-based modules in advance for several configuration parameters and then providing the results to the framework. The estimated characteristics are then scaled according to the currently investigated parameters. The same approach is also possible for metrics such as latency and throughput. Even more precise estimates can be generated by using the HLS-based estimation similar to the design flow presented in section 6.3.4.

8.3. NeuroBayes Demonstrator

The realization of the OCA on the DHH as described in section 8.2 is designed for the requirements dictated by the Belle II experiment. The throughput was hereby designed to keep up with the maximum throughput achieved by the clustering as it represented the strictest requirement to be fulfilled. To meet these requirements it is already sufficient to implement a single instance of the zero iteration based processing architecture. Meanwhile, one single instance required only a few of the available resources that are provided by the hosting FPGA. Additionally, most resources of the FPGA were already occupied by the clustering. For the developed implementation and architecture, however, an open question that is remaining is to what extent the throughput is scalable, which is particularly important in systems with higher throughput requirements. A demonstrator was developed to answer these questions. This demonstrator is presented and discussed throughout this section.

Selection of Communication Interfaces

Since the goal of the demonstrator is to achieve the highest possible throughput, a communication interface is required with which as much data as possible can be transferred to and from the FPGA. The PCIe is hereby particularly suitable for this task by providing high data rates and being widely supported in regular computing systems such as desktop computers. Most modern high-performance GPUs are equipped with this interface. Since the high-performance computing market is of interest to FPGAs, most of their modern development platforms are equipped with a PCIe interface to be connected as a co-processor. This widespread adoption of PCIe is making it easier to find a suitable platform.

Selection Hardware Platform

Category/Platform	DHH	Demonstrator
Interface	AXI Stream	PCIe
Maximum incoming data rate	200	3.6
Maximum outgoing data rate	200	1
FPGA	Virtex-6	Virtex-7
DSPs	480	3600

Table 8.5.: Comparison of characteristics between the demonstration platform and the DHH that is hosting the OCA.

The VC709 development platform from Xilinx was selected as the hosting FPGA platform. At the time of development, it had the most advanced FPGA available, a Virtex-7, which is a generation newer than the FPGA used at the DHH. In addition, this platform is supporting the PCIe 3.0 standard, which at that time was the version with the highest achievable data throughput. More modern platforms nowadays are meanwhile already relying on PCIe 4.0. The FPGA also provides a large number of DSP resources. Since these

make up the main part of the algorithm's processing, the platform is suitable for exploring a high degree of parallelism. Table 8.3 compares the properties of both the DHH and the demonstrator. The FPGA-based PCIe infrastructure is meanwhile based on Ref. [San14b].

System Architecture

The overall system architecture is based on a standard desktop computer that is equipped with a PCIe slot in which the VC709 is inserted. Both the desktop PC and FPGA need additional functionality in order to establish high throughput communication. On the host side, a PCIe driver is required to access the FPGA. At the same time, an infrastructure based around a PCIe IP core is required on the FPGA to take care of the PCIe protocol. In order to achieve the highest possible throughput, communication must be carried out via Direct Memory Access (DMA), which needs to use a dedicated controller on the FPGA. The theoretical maximum achievable throughput can hereby only be achieved if the data transfers are carried out with maximum pipelining.

On the host side, a Linux operating system is used to allow easy development of the device driver. On the start-up of the system, it will find and bind the correct PCIe device based on the device code. The main functionality is hereby in setting up and initiating the DMA transfers. In addition, the driver handles interrupts sent by the FPGA, which will be used to indicate the conclusion of estimations. To initiate the data transfers, the driver is transmitting both the start and end addresses of the data to be fetched from the main memory via memory-mapped IO. In addition, the address range is transferred at which the FPGA is allowed to write its results.

The driver is used by an application on the host side that loads simulated detector data into memory and initiates data transfers to the FPGA. This application also contains a visualization showing the achieved results and throughput. It provides two separate operating modes. Either a data set is manually defined and transmitted to the FPGA, with the result being read out afterwards for evaluation. This is of particular interest for demonstrating correctness together with specific test data samples. The other mode is used for demonstrating the throughputs that can be achieved. In this mode, cluster data is read from a predefined file and written into the designated locations in the main memory. This is concluding with the initiation of the transfer to the FPGA. The operation is then performed in a loop so that data is transmitted continuously until an abort is requested explicitly. In parallel to this transfer loop, the output of the FPGA is continuously evaluated with the throughput being measured either for a specific time interval or by the total accumulation of estimations over time.

On the FPGA side, the architecture can be divided into three sections. The first section is consisting of all the components required for PCIe communication. A custom-designed infrastructure meanwhile controls all the DMA operations for reading and writing the data. It includes an interrupt control unit that supports the usage of MSIX. Following this, data is interpreted as cluster data and distributed among several parallel instances of the OCA.

The realization of the OCA is configured to host the maximum amount of parallel instances at which the output throughput is still increasing. Since the output data rate is representing the bottleneck for the demonstrator, the amount of instances is set in a way

that it is saturated. When using only binary decisions describing the classification, the maximum degree of parallelism is equal to the number of bits that can be transferred per clock cycle. For the selected host platform, this is at to 128 bit, so that 128 instances are instanced in parallel. This amount of instances of the OCA are then occupying around 40% of the available resources. The overall system architecture is meanwhile shown in figure 8.10.

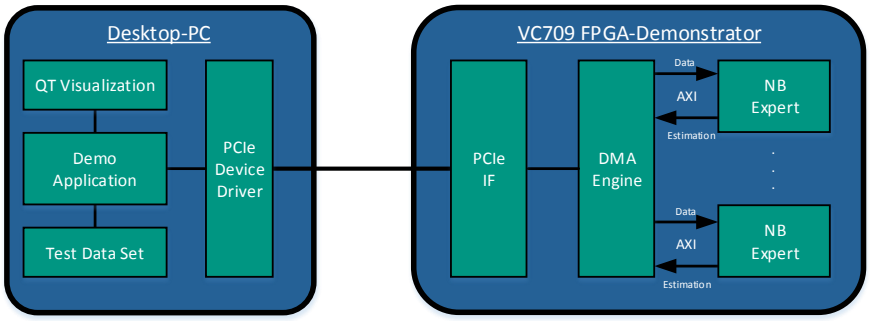


Figure 8.10.: System architecture of the throughput demonstrator based on the VC709.

8.4. Summary

For the identification of slow hadrons from the collisions of the Belle II experiment, a online pixel cluster analysis based on the NeuroBayes algorithm was developed for operation on FPGAs. When integrated this approach within the DAQ of the PXD at the so-called DHH, it is capable of achieving the requested functionality while fulfilling all operational requirements in terms of resource demand and throughput.

The presented solution of the NeuroBayes algorithm on FPGAsrepresents the first realization of this algorithm based on this technology. The design choices and alternatives are meanwhile mainly dependent on the internally used bit widths of the input values as well as the options of used preprocessing. For the OCA application case, a throughput focused architecture was developed. A latency optimized alternative is architecturally feasible using the alternative realization of neurons discussed in section 4.4.The developed architecture is flexibly configurable in order to be adjusted due to changing operational conditions. This is supported by a developed semi-automated design framework that maps a description of the algorithm to a corresponding architecture. Properties such as resource demand, throughput, latency and above all the influence of bit widths were investigated and discussed. The framework is meanwhile based on Python in order to allow integration with the provided unit tests. The influence of bit widths meanwhile conducted using modelling of hardware processing with the help of the myHDL library [24]. This is supported by HW/SW co-simulation for validation using the FLI of Modelsim [76]. Overall,

the OCA can be implemented with the architecture and implementation presented here within the resource budget and in compliance with all requirements of the experiment. The problem for future use is that the algorithm is proprietary and might thus stay in its prototype form.

9. Conclusion and Future Work

9.1. Conclusion

Modern particle accelerator experiments achieve extreme data rates for which the transmission infrastructure necessary for the complete storage of the generated data would be too expensive to be feasible. To address this problem, typically mechanisms for triggered data readout are used. These mechanisms select at runtime which data is to be stored for later analysis. Such systems were part of most of the high data rate experiments in the past and allowed a good trade-off between infrastructure cost and efficiency in terms of keeping most of the desired data that is a result of the experiment. One of the experiments that is depending on the usage of an efficient trigger system is Belle II. Preliminary analysis of the experiment's behaviour hereby shows that a large fraction of the expected tracks is going to be caused by unwanted background events instead of collisions, having their point of origin outside of the interaction point with respect to the z-Axis that is along the beam pipe. Identifying and suppressing such tracks is one of the primary goals of the L1 trigger system of the experiment. While many established algorithms are available to address this problem, the task is made difficult by the currently unknown behaviour of high luminosity operation and the complex structure of the observed background events. The motivation is now to employ algorithms based on machine learning to solve these problems.

The main contribution of this thesis is thus the neural z-Vertex Trigger for the Belle II experiment. This trigger represents the first trigger system based on neural networks that is operating at the L1 of an experiment's trigger system. First and foremost this thesis shows that such a system can be implemented on the FPGAs available at the time of the construction of Belle II while fulfilling all of the operational requirements in terms of latency, integration, and throughput. It is then shown that such a trigger system is capable of estimating the z-Vertex of tracks with reasonable accuracy. It is currently capable of being used with a resolution of around +/- 40 cm.

Along the way of developing this system, several achievements were achieved. At first, a general architecture on the basis of dedicated preprocessing algorithms and general MLPs was investigated and developed. The preprocessing is designed to be highly flexible in terms of parallelism and the internally used bit widths. While the MLP is implemented to be low-latency and resource-efficient to fulfil the set requirements. Several optimization techniques such as layer pipelining and heterogeneous resource usage were developed and discussed, which to produce a solution dedicated to the trigger use case. The correctness of the approach is shown with example collision runs in which DQM data is available and histograms of the z-Vertex were created across all NNT boards. Fulfilment of both latency and throughput is performed experimentally as well. Latency was measured at the

data sink of the trigger system, the GDL, at which trigger signals are arriving just in time to be used for the decision making. Throughput is currently sufficient to keep up with early luminosity runs as no tracks are dropped in all of the considered runs.

The scope of the NNT development was expanded towards higher luminosity operation and the new hardware platform that will be used in later stages. The presented architecture was hereby re-evaluated and configured to be operational on this platform. Results showing resource utilization were generated by developing an early prototype since the real platform is not available. It shows that the architecture can be easily transitioned to the new platform, even increasing the throughput by allowing the usage of multiple instances of the NNT on one FPGA.

From an integration point of view, this work includes all of the aspects required to be addressed in order to reach operational completeness. This started with the investigation of viable hardware-platforms based on FPGAs. The platform was chosen to fulfil the required number of IO ports, support for assisting IP cores and effort for integration into the location at which trigger hardware is placed. To compensate for the behaviour of the central drift chamber, dedicated protocol IP cores were developed in a flexible way to deal with changing requirements. In addition to this, interfaces and solutions for integrating the NNT into both slow control and data quality monitoring were developed and presented.

The NNT itself is already a powerful tool that allows triggering on the z-Vertex with a sufficiently reasonable resolution. However, an even better resolution is requested for later operation. The strategy followed for increasing the resolution is to focus on the preceding processing stages. For this, two optimization strategies were investigated within this thesis. The most powerful and more experimental approach is represented the Hough-based 3D-Track Finding. This method significantly increased both the efficiency and accuracy of the z-Vertex estimation. Within this thesis, an FPGA-based architecture was presented that is realizing the S3D. The presented architecture is capable of providing 3D-track parameters within the set requirements and is currently in a prototyping state since the hardware is currently in production. In addition to a study about the feasibility of integrating this system into the current CDCTRG, an investigation was conducted to explore the possibilities of integrating the S3D with the NNT. These studies showed that the currently planned FPGA VU125 will be capable of hosting both the S3D and two parallel instances of the NNT, which will increase the number of tracks processed at each clock cycle.

A less powerful but important optimization is presented with the redesign of the Track Segment Finder. This component represents the basis for all track trigger operation within the CDCTRG. By using a state-machine-based architecture called TSFsm, several characteristics were improved. Not only is resource consumption reduced across all SL, but the new system is also increasing the efficiency for finding particle tracks. It is meanwhile already in operation in the experiment providing TS reliably during collision operation.

While the main contribution of this thesis is represented by the achievements targeting the Belle II trigger system, additional data reduction approaches based on machine learning were investigated. The Online Cluster Analysis was developed for this purpose, which is currently in a prototyping state. It is based on the proprietary NeuroBayes algorithm and capable of classifying particles, that is slow pions and background, reliably while being

deployed on FPGAs located close to the pixel detector readout. As data rate reduction is most efficient when performed as early as possible in the process of data taking, the OCA is performed online as part of the experiment's DAQ. An architecture fulfilling all of the imposed requirements was developed for this purpose. It is based on throughput optimized processing of neural networks, as this is the tightest requirement to be fulfilled by the system.

9.2. Future Work

The presented systems that are estimating particle track parameters or identifying particle types online and in real-time, are the first operational realizations of this type based on machine learning methods. As they are representing the first such systems and are part of a high energy physics experiment that is currently in its first stages of operation several opportunities for future work are present.

The main opportunity for the extension is at the analysis and observation of the operational setup as presented in section 5.4.2.1. While it is already achieving good estimations for the current state of the experiment, there is still room for improvement. The measured resolution is sufficient to operate the trigger in early stages, however, the requirements will tighten with the increase of luminosity. The planned approach to address this is to train neural networks on the basis of collision data since the current system is still based around simulated collisions. Using such a network is projected to improve the estimation's efficiency significantly, thus the probability of successfully improving the NNT is very high since its architecture is designed to be flexible. The employed network was hereby already updated multiple times, for example, with weight sets trained for cosmic rays or extension of the estimation to +/- 100 cm.

Further optimization based on the presented operational setup can be achieved by using the ETF as intended in the original trigger concept. The ETF is still under development and scheduled for a redesign until then the alternative event time estimation based on the fastest priority times is going to be used. The current setup will have to be reconfigured in order to include data from ETF. Since the ETF is in a development state, the success of this is more difficult to estimate. However, from a theoretical point of view with ideal event times, the improvement to the estimation is significant. Even without this optimization, the NNT should be fine for operation.

One immediate change and opportunity for improvement is the extension of the system to the usage of 15 TS since the current version is only supporting up to 10. Studies based on early collision data showed that certain events will generate more than 10 TS, which will however currently be ignored since they are are not used. Usage of the full 15 TS will impact the NNT as its preprocessing has to be adjusted to reflect the increased amount of data to be processed. At the same time, this is invoking changes in both SC and DQM, as they have to be extended to send the increased size of data. Its impact on the achievable resolution is currently unknown and will have to be investigated in further studies.

Longer term improvements are mostly based around the usage of newer FPGA-platforms such as the UT4. While architecture and integration were being presented within this thesis, the platform itself is currently not available. All of the physical integration tasks and

subsequent evaluations have to be performed as soon as it is available. However, the risk of the NNT not being used on the UT4 is rather low since a prototypical implementation is already available.

The highest projected increase in resolution is achieved by the S3D as presented in section 6. Due to the non-availability of the UT4 and early development stage of the Hough-based processing, it is currently waiting to be integrated into the CDCTRG. However as presented in this thesis, a reasonable integration concept was developed that is projected to be realizable with significant improvements to the overall operation. As this step is highly dependent on the new unknown platform it has both the highest risks as well. However, all of the previously mentioned improvements have a very high probability of success.

Reaching beyond the work presented in this thesis several additional opportunities exist. First of all new modern machine learning algorithms and FPGA-based inference frameworks could be explored. On the other hand, the concept of a neural network trigger can be extended towards a new trigger system that is combining data from multiple detectors instead of only the CDC. This approach is already a work in progress on a conceptual stage, with the TOP detector, for example, providing the event time. On an architectural level, improvements can be achieved by extending the concept of dedicated neural networks targeting certain special conditions. This is based around the usage of the entire on-chip memory or even inclusion of off-Chip memory, which might be within the latency budget for the integrated S3D and NNT solution.

A. Appendix

A.1. Neural z-Vertex Trigger

A.1.1. Belle2Link Data Quality Management Interface

B2Link Data overall			MLP Input INTERN		
MSB	LSB	Name	MSB	LSB	Name
2047	2032	hex"dddd"	1521	1405	Alpha[0,8]
2031	2016	"00000" & B2l Clock	1404	1288	DT[0,8]
2015	2000	NNT Clock Counter	1287	1171	ID[0,8]
1999	1574	unused			
1573	1568	old track found	TSF INTERN		
1567	1567	valid stereo bit	MSB	LSB	Name
1566	1558	2dcc	561	541	TSF1[9]
1557	1552	2dval			...
1551	1551	Enable NNT	372	352	TSF1[0]
1550	1538	MLP Output 1			
1537	1525	MLP Output 2	TSF Selected INTERN		
1524	1522	NetSel	MSB	LSB	Name
1521	1171	MLP Input	1170	1150	TSF Selected_SL8
1170	982	TSF Selected			...
981	772	TSF7	1002	982	TSF Selected_SL0
771	562	TSF5			
561	352	TSF3			
351	142	TSF1	ETF Input INTERN (Board 3D2)		
141	121	TSF0	MSB	LSB	Name
120	100	TSF2	38	30	Clock Counter
99	79	TSF4	29	17	T0
78	58	TSF6	16	16	Valid
57	37	TSF8			
36	30	Omega	ETF Input INTERN (Board 3D0)		
29	23	Phi	MSB	LSB	Name
22	17	old track found	22	14	Clock Counter
16	16	valid stereo bit	13	10	"0000" Padding
15	7	drift threshold	9	1	T0
6	0	unused	0	0	Valid

Figure A.1.: Table of the used B2L-DQM format of the NNT from available since operation in phase 2.

The method used for DQM via the B2L, which is presented in section 5.3.2, used the format shown in table A.1 during the first stages of the NNT's operation. The current

format is found in the internal document management system of the experiment, however this version already provides the ideas and meaning of this format.

A.1.2. Description of the accepted Network Input

The following describes the text file format used for the configuration of the NNT. It is generated by tools at the algorithmic side of the design flow and used by the framework supporting the firmware generation to support the configuration of the architecture.

networkNumber_completeTS
ID0min ID0max ID1min ... ID8max
sectorpattern patternmask
numInputs numHLNeurons numOLNeurons
weight_HLNeuron0_Input0 weight_HLNeuron0_Input1 ... weight_OLNeuronN_InputM
networkNumber_stereo1missing
...
networkNumber_stereo7missing
...
weight_HLNeuron0_Input0 weight_HLNeuron0_Input1 ... weight_OLNeuronN_InputM

A.1.3. Results of NNT operation during experiment 8

The most important results of the NNT were shown in section 5.4.2.2. This attachment is expanding on that by showing the results in more detail, that is distributed across the separate quadrant, which allows to analyze the correctness for each of the four UT3s hosting the NNT.

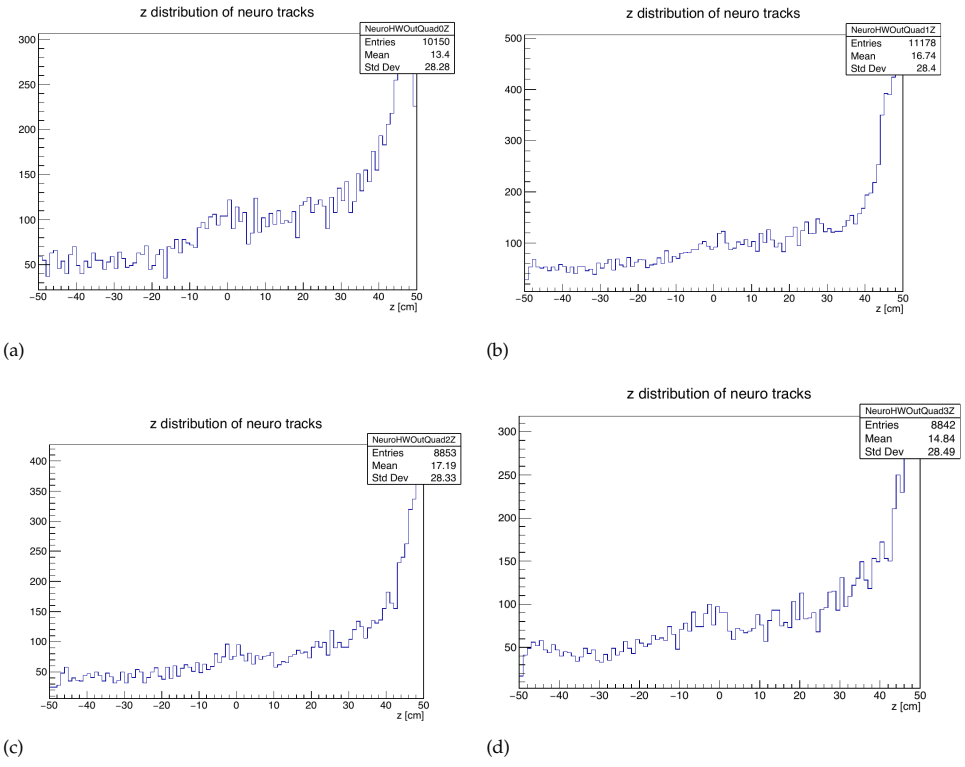


Figure A.2.: Histograms of z-Estimations sent by the NNT for each quadrant are shown, the quadrant is indicated within each plot in the information box. Reasonable distributions are generated for each quadrant.

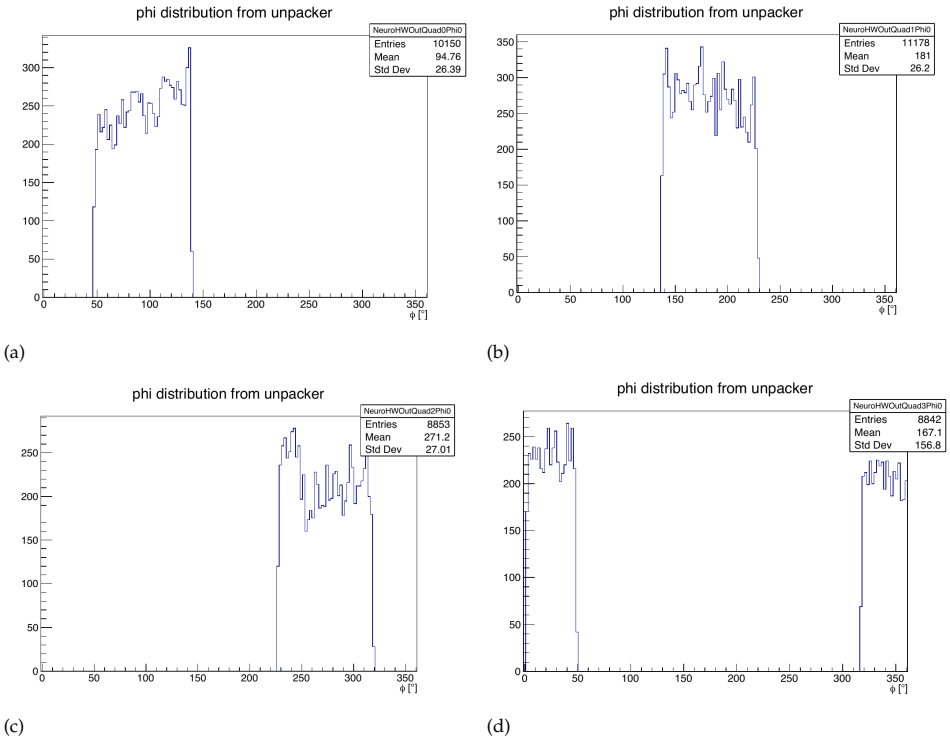
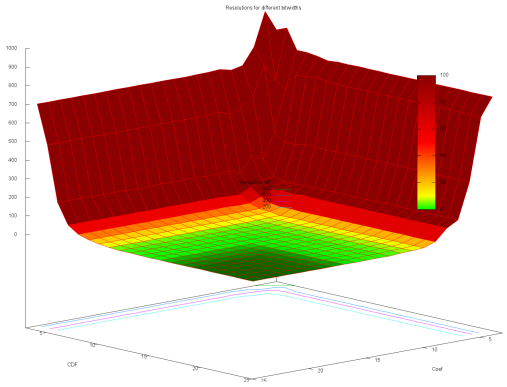
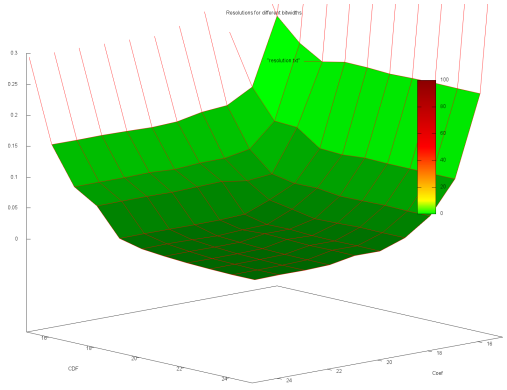


Figure A.3.: Histograms of phi values received at NNT for each quadrant are shown, the quadrant is indicated within each plot in the information box. All quadrants a received correctly.

A.2. OCA bit width analysis



(a)



(b)

Figure A.4.: Comparison of HW and SW implementation of the zero-iteration for different bit widths with the deviation being plotted. Ranges from 3 to 25 bits are shown in (a) while the range 15 to 25 is shown in more detail in (b). The later range represents the operating range supported by the DSPs of the considered FPGA.

A.3. TSFsm

A.3.1. Data format and Interface

	Super Layer 0		Super Layer 1-8	
Input:	FIFO_IN	256 bit	FIFO_IN	256 bit
	EDGE_M_IN	18 bit	Edge_Time_in	12 bit
	EDGE_P_IN	12 bit	Edge_hitA_in	4 bit
	USER_CLK	1 bit	Edge_hitB_in	2 bit
	Edge_fphit_i	**1 bit**	USER_CLK	1 bit
	Edge_fphit_o	**1 bit**	**Edge_fphit_i**	**1 bit**
			Edge_fphit_o	**1 bit**
Output:	TSF_trackerOUT m	variable	TSF_trackerOUT m	variable
	TSF_evtOUT m	variable	TSF_evtOUT m	variable

(a) (b)

	Super Layer 0		Super Layer 1-8	
Input:	CLKIN	1 bit	CLKIN	1 bit
	HITMAP	15 bit	HITMAP	11 bit
	PRIORITYTIME	4 bit	PRIORITYTIME	4 bit
	SECONDPRIORITY	1 bit	SECONDPRIORITY	1 bit
	NeighborHit	**1 bit**	**NeighborHit**	**1 bit**
	positionCounter	5 bit	positionCounter	5 bit
Output:	RorL_2nd	1 bit	RorL_2nd	1 bit
	DOUT	13 bit	DOUT	13 bit
	EOUT	10 bit	EOUT	10 bit

Figure A.5.: Interfaces provided by the new TSFsm. (b) is showing the interface of the new TSF_m and (a) the interface to the state machine module [Ung18].

A.3.2. Configuration of the TSF within the CDCTRG

As described in section 5.2.1, a total of four UT3 boards is used to cover the entire CDC the amount GTH ports required to the supported for full readout. The unique assignment of TS, represented by its ID, to each boards is shown in table A.3.2. In addition, the table contains the total number of TS and supported mergers. With this table it is possible to identify the space covered by each board hosting an instance of the NNT.

SL	#Merger	#TS	Quad 0	Quad 1	Quad 2	Quad 3
TSF0	7	160	0-79	40-119	80-159	120-159, 0-39
TSF1	5	160	0-79	40-119	80-159	120-159, 0-39
TSF2	6	192	0-95	48-143	96-191	144-191, 0-47
TSF3	7	224	0-111	56-167	112-223	168-223, 0-55
TSF4	8	256	0-127	64-191	128-255	192-255, 0-64
TSF5	9	288	0-143	72-215	144-287	216-287, 0-71
TSF6	10	320	0-159	80-239	160-319	240-319, 0-79
TSF7	11	353	0-175	88-263	176-351	264-352, 0-87
TSF8	12	384	0-191	96-287	192-383	288-383, 0-95

Table A.1.: Configuration of the TSF present in the CDCTRG together with the unique assignment of each TS and their quadrant.

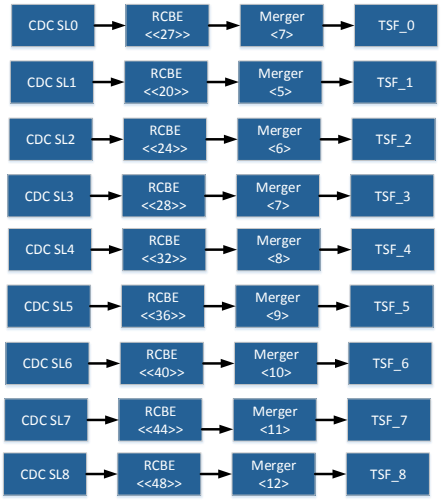


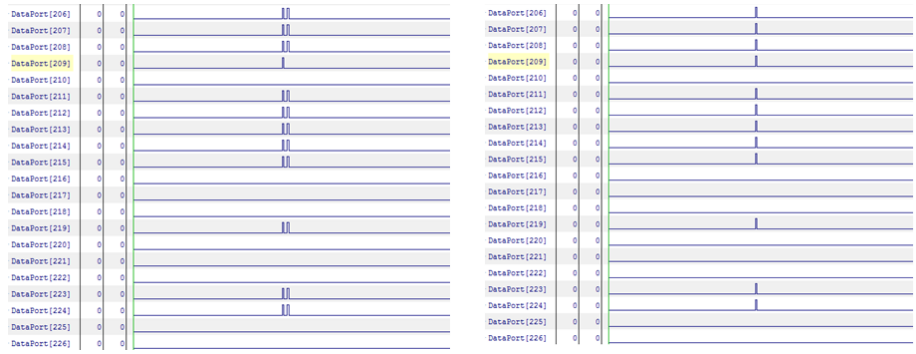
Figure A.6.: Graphical representation of the system architecture connecting the CDC with the entry point into the CDCTRG.

A.3.3. Validation of the TSFsm

Instead of the relative amounts of found TS during testing with random number generated merger inputs, table A.7 is showing the absolute numbers.

	HIT	First Priority	Second Priority	**HIT**	First Priority	Second Priority
Track Segment 0	**61579**	34429	27150	**50403**	27329	23074
Track Segment 1	**61011**	33744	27267	**50369**	27369	23000
Track Segment 2	**60937**	33373	27564	**50400**	27109	23291
Track Segment 3	**60907**	33754	27153	**50438**	27304	23134
Track Segment 4	**60901**	33842	27059	**50402**	27360	23042
Track Segment 5	**60561**	33339	27222	**50275**	27150	23125
Track Segment 6	**60643**	33519	27124	**50196**	27196	23000
Track Segment 7	**61126**	33892	27234	**50304**	27395	22909
Track Segment 8	**60735**	33705	27030	**50270**	27247	23023
Track Segment 9	**60847**	33734	27113	**50201**	27262	22939
Track Segment 10	**60738**	33794	26944	**50248**	27425	22823
Track Segment 11	**60765**	33659	27106	**50472**	27342	23130
Track Segment 12	**60922**	34022	26900	**50340**	27557	22783
Track Segment 13	**60857**	33613	27244	**50521**	27305	23216
Track Segment 14	**60644**	33473	27171	**50335**	27143	23192
Track Segment 15	**61043**	33616	27427	**50329**	27221	23108

Figure A.7.: Comparison of the total number of found hits with random input generation using the TSFsm.



(a) (b)

Figure A.8.: Signal diagrams showing data from both TSF (a) and TSFsm (b) that were captured using Chipscope while performing merger play tests.

List of Figures

List of Tables

Acronyms

ACAP Adaptive Compute Acceleration Platform
ARICH Aerogel Ring Imaging Cherenkov Detector
ASIC Application Specific Integrated Circuit
BPID Barrel Particle Identification Detector
BASF2 Belle Analysis Software Framework 2
BTRGSRV Belle II Trigger Server
BNN Binary Neural Networks
BW bit width
BRAM Block Random Accessible Memory
CDC Central Drift Chamber
CPU Central Processing Unit
CP Charge and Parity
CMS Compact Muon Solenoid
CLB Configurable Logic Block
CNAPS Connected Network of Adapted Processors
CERN European Organization for Nuclear Research
CNN Convolutional Neural Networks
CDF Cumulative distribution function
DAQ Data Aquisition
DHHC Data Handling Controller
DHH Data Handling Hybrid
DHP Data Handling Processor
DQM Data Quality Monitoring
DLA Deep Learning Accelerator
DNN Deep Neural Networks
DSP Digital Signal Processor

DMA Direct Memory Access
DDR Double Data Rate
DCD Drain Current Digitizer
ECLTRG Trigger System of the ECL Detector
EDA Electronic Design Automation
E-Hut Electronics Hut
EPID Endcap Particle Identification Detector
ECL Energy and Cluster Detector
ETF Event Time Finder
EPICS The Experimental Physics and Industrial Control System
FPGA Field Programmable Gate Array
FIFO First-In First-Out Memory
FLI Foreign Language Interface
FINN Framework for Fast Scalable Binarized Neural Network Inference
FEE Frontend Electronics
GT Gigabit Transceivers
GDL Global Decision Logic
GRL Global Reconstruction Logic
GPU Graphics Processing Unit
HDL Hardware Description Language
HL Hidden Layer
HLS4ML High-Level Synthesis for Machine Learning
HBM High Bandwidth Memory
KEK High Energy Accelerator Research Organization
HLS High-Level Synthesis
HLT High Level Trigger
S3D Hough-based 3D-Track Finding
HMC Hybrid Memory Cube
TANH Hyperbolic Tangent
ID Identification Number
IO Input/Output
IC Integrated Circuit
ILA Integrated Logic Analyzer
IP Intellectual Property

JTAG Joint Test Action Group
KLM Kaon and Muon Detector
KEKB KEK-B-factory
KEKCC KEK Computing Cluster
LHC Large Hadron Collider
LR Left Right information
L Level
L1 First Level
LUT Look Up Table
LVDS Low Voltage Differential Signalling
ADC Analog Digital Converter
ML Machine Learning
MPI Max-Planck Institute
MLP Multi Layer Perceptron
MAC Multiply and Accumulate
NSM Network shared memory
NNT neural z-Vertex Trigger
OCA Online Cluster Analysis
ONSEN ONline SElector Node
OL Output Layer
PCIe Peripheral Component Interconnect express
PC Personal Computer
PXD Pixel Detector
PDF Probability Density Function
QED Quantum Electrodynamics
RAM Random Memory Access
RECBE Readout Electronics for the CDC
RX receiving port
ReLU Rectifier Function
RNN Recurrent Neural Networks
RoI Region of Interest
RTL Register Transfer Level
ReBNet Residual Binarized Neural Network
SVD Silicon Vertex Detector

SBC Single Board Computer
SIMD Single Input Multiple Data
SC Slow Control
SW Software
SRAM Static Random Memory Access
TDC Time to Digital Converter
TOP Time of Propagation Detector
2DS 2D-Track Finder
3DS Conventional 3D-Track Finding
TS Track Segment
TSF Track Segment Finder
sTSF stereo Track Segment Finder
TSFsm Track Segment Finder based on state machines
aTSF axial Track Segment Finder
TX transmitting port
TSIM Trigger Simulation
CDCTRG Trigger system of the Central Drift Chamber
TTC Trigger and Timing Control
UT Universal Trigger Board
VCD Value Change Dump
VME Versa Module Eurocard bus
VXD Vertex Detector
VHDL Very High Speed Integrated Circuit Hardware Description Language
VLIW Very Long Instruction Word
xDNN Xilinx Deep Neural Network Processor

Bibliography

[1] ABACO SYSTEM: *GE V7865 Datasheet*.

[2] ABADI, M. et al.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. CoRR, abs/1603.04467, 2016.

[3] ABDELFATTAH, M. S., D. HAN, A. BITAR, R. DICECCO, S. O'CONNELL, N. SHANKER, J. CHU, I. PRINS, J. FENDER, A. C. LING and G. R. CHIU: *DLA: Compiler and FPGA Overlay for Neural Network Inference Acceleration*. CoRR, abs/1807.06434, 2018.

[4] ABE, K. et al.: *Observation of Large CP Violation in the Neutral B Meson System*. Phys. Rev. Lett., 87:091802, Aug 2001.

[5] ABE, T. et al.: *Belle II Technical Design Report*. 2010.

[6] ABREU, P. et al.: *Observation of orbitally excited B mesons*. Physics Letters B, 345(4):598 – 608, 1995.

[7] ADACHI, I. AND BROWDER, T.E. AND KRIZAN, P. AND TANAKA, S. AND USHIRODA, Y.: *Detectors for extreme luminosity: Belle II*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 907:46 – 59, 2018. Advances in Instrumentation and Experimental Methods (Special Issue in Honour of Cai Siegbahn).

[8] ADAPTIVE SOLUTIONS INC.: *CNAPS System Architecture Manual*. Beaverton, Oregon, 3.0 ed., 1993.

[9] ADAPTIVE SOLUTIONS INC.: *CNAPS VME Board Reference Manua*. Beaverton, Oregon, 1.0 ed., 1993.

[10] AJUHA, S., A. CASCADAN, T. COSTA DE PAIVA, S. DAS, R. EUSEBI, V. FINOTTI FERREIRA, K. HAHN, Z. HU, S. JINDARIANI, J. KONIGSBERG, T. T. LIU, J. F. LOW, Y. OKUMURA, J. OLSEN, L. ARRUDA RAMALHO, R. ROSSIN, L. RISTORI, A. AKIRA SHINODA, N. TRAN, M. TROVATO, K. ULMER, M. VAZ, X. WEN, J.-Y. WU, Z. XU, H. YIN and S. ZORZETTI: *A Full Mesh ATCA-based General Purpose Data Processing Board (Pulsar II)*. Techn. Rep. FERMILAB-TM-2650-E, Jun 2017.

[11] ANTCHEVA, I., M. BALLINTIJN, B. BELLENOT, M. BISKUP, R. BRUN, N. BUNCIC, P. CANAL, D. CASADEI, O. COUET, V. FINE, L. FRANCO, G. GANIS, A. GHEATA, D. G. MALINE, M. GOTO, J. IWASZKIEWICZ, A. KRESHUK, D. M. SEGURA, R. MAUNDER, L. MONETA, A. NAUMANN, E. OFFERMANN, V. ONUCHIN, S. PANACEK, F. RADEMAKERS, P. RUSSO and M. TADEL: *ROOT - A C++ framework for petabyte data storage, statistical analysis and visualization*. Computer Physics Communications, 180(12):2499 – 2512, 2009. 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing

architectures.

[12] AUBERT, B. et al.: *Observation of CP Violation in the B^0 Meson System*. Phys. Rev. Lett., 87:091801, Aug 2001.

[13] AUSHEV, T. et al.: *A scintillator based endcap K_L and muon detector for the Belle II experiment*. Nucl. Instrum. Meth., A789:134–142, 2015.

[14] AYDONAT, U. AND O'CONNELL, S. AND CAPALIJA, D. AND LING, A. AND CHIU, G.: *An OpenCL™Deep Learning Accelerator on Arria 10*. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, pp. 55–64, New York, NY, USA, 2017. ACM.

[15] BELLE II COLLABORATION: *Super KEKB and Belle II*, 2019.

[16] BERNLOCHNER, F. AND DESCHAMPS, B. AND DINGFELDER, J. AND MARINAS, C. AND WESSEL, C.: *Online Data Reduction for the Belle II Experiment using DATCON*. EPJ Web Conf., 150:00014, 2017.

[17] BETZ, V. and J. ROSE: *VPR: a new packing, placement and routing tool for FPGA research*. In LUK, W., P. Y. K. CHEUNG and M. GLESNER (eds.): *Field-Programmable Logic and Applications*, pp. 213–222, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[18] BLANCO, R. AND LEYS, R. AND PERIĆ, I.: *Integrated readout electronics for Belle II pixel detector*. Journal of Instrumentation, 13(03):C03001–C03001, mar 2018.

[19] CAI, Y.: *Design of an asymmetric Super-B Factory*. In *Proceedings of EPAC 2016, Edinburgh Scotland*, 2006.

[20] CALAFIURA, P. et al.: *TrackML: A High Energy Physics Particle Tracking Challenge*. In *Proceedings, 14th International Conference on e-Science: Amsterdam, Netherlands, October 29-November 1, 2018*, p. 344, 2018.

[21] CHEON, B.-G. et al.: *Electromagnetic calorimeter trigger at Belle*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 494(1):548 – 554, 2002. Proceedings of the 8th International Conference on Instrumentatio n for Colliding Beam Physics.

[22] COURBARIAUX, M. AND BENGIO, Y.: *BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. CoRR, abs/1602.02830, 2016.

[23] DEAN, J. AND THE GOOGLE BRAIN TEAM: *Recent Advances in Artificial Intelligence and the Implications for Computer System Design*. https://www.hotchips.org/archives/2010s/hc29/, August 2017.

[24] DECALUWE, J.: *MyHDL manual*, 0.11 ed., May 2019.

[25] DENBY, B. AND GARDA, P.: *Fast triggering in high-energy physics experiments using hardware neural networks*. IEEE Transactions on Neural Networks, 14(5):1010 – 1027, 2003.

[26] DUARTE, J. et al.: *Fast inference of deep neural networks in FPGAs for particle physics*. JINST, 13(07):P07027, 2018.

[27] EVANS, L. and P. BRYANT: *LHC Machine*. Journal of Instrumentation, 3(08):S08001,

2008.

[28] FEINDT, M.: *A Neural Bayesian Estimator for Conditional Probability Densities*. ArXiv Physics e-prints arXiv:physics/0402093, Feb. 2004.

[29] FELDBAUER, F.: *Analyse des Zerfalls bei BES-III und Entwicklung der Slow Control fuer das PANDA-Experiment*. PhD thesis, Ruhr-Universitaet Bochum, 2012.

[30] FINKELSTEIN, U. et al.: *GTKWave 3.3 Wave Analyzer User's Guide*. http://gtkwave.sourceforge.net.

[31] FIORINI, M.: *The NA62 Gigatracker: Detector properties and pixel read-out architectures*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 624(2):314 – 316, 2010. New Developments in Radiation Detectors.

[32] GHASEMZADEH, M., M. SAMRAGH and F. KOUSHANFAR: *ResBinNet: Residual Binary Neural Network*. CoRR, abs/1711.01243, 2017.

[33] GIORDANO, R., G. TORTONE, S. PERRELLA, V. IZZO and A. ALOISIO: *Monitoring single event upsets in SRAM-based FPGAs at the SuperKEKB interaction point*. Journal of Instrumentation, 12(07):C07039, 2017.

[34] GRINSTEIN, S. et al.: *Overview of the ATLAS insertable B-layer (IBL) project*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 699:61 – 66, 2013. Proceedings of the 8th International Hiroshima Symposium on the Development and Application of Semiconductor Tracking Detectors.

[35] GUO, K., L. SUI, J. QIU, S. YAO, S. HAN, Y. WANG and H. YANG: *From model to FPGA: Software-hardware co-design for efficient neural network acceleration*. pp. 1–27, 08 2016.

[36] HAN, S., J. KANG, H. MAO, Y. HU, X. LI, Y. LI, D. XIE, H. LUO, S. YAO, Y. WANG, H. YANG and W. J. DALLY: *ESE: Efficient Speech Recognition Engine with Compressed LSTM on FPGA*. CoRR, abs/1612.00694, 2016.

[37] HAN, S., H. MAO and W. J. DALLY: *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. arXiv e-prints, p. arXiv:1510.00149, Oct 2015.

[38] HIGUCHI, T. et al.: *Development of a PCI based data acquisition platform for high intensity accelerator experiments*. eConf, C0303241:TUGT004, 2003.

[39] HIGUCHI, T., M. NAKAO and E. NAKANO: *Radiation tolerance of readout electronics for Belle II*. Journal of Instrumentation, 7(02):C02022, 2012.

[40] HORII, Y.: *TOP Detector for Particle Identification at the Belle II Experiment*. The European Physical Society Conference on High Energy Physics -EPS-HEP2013, 2013.

[41] IWASAKI, Y.: *TSF State Machine idea*. KEK, 2017.

[42] JAIN, A. K., S. A. FAHMY and D. L. MASKELL: *Efficient Overlay Architecture Based on DSP Blocks*. In *Proceedings of the 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, FCCM '15, pp. 25–28, Washington, DC, USA, 2015. IEEE Computer Society.

[43] JAMES, T.: *Level-1 Track Finding with an All-FPGA System at CMS for the HL-LHC*. In *2019 Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019)*, 2019.

[44] JDA SOFTWARE GROUP: *Blue Yonder Webpage*, September 2019.

[45] JEDEC SOLID STATE TECHNOLOGY ASSOCIATION: *JESD235B - HIGH BANDWIDTH MEMORY (HBM) DRAM*.

[46] JIA, Y., E. SHELHAMER, J. DONAHUE, S. KARAYEV, J. LONG, R. B. GIRSHICK, S. GUADARRAMA and T. DARRELL: *Caffe: Convolutional Architecture for Fast Feature Embedding*. CoRR, abs/1408.5093, 2014.

[47] JOHNSON, R. AND STEWART, C.: *JTAG 101 : IEEE 1149.x and Software Debug*. Intel Corp., 2009.

[48] KEMMER, J. and G. LUTZ: *New detector concepts*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 253(3):365 – 377, 1987.

[49] KIESLING, C.: *The Neural z Trigger Project*. Neurotrigger Spring 2019 Project Meeting. "https://confluence.desy.de/display/BI/Trigger+NNT+Meeting+Slides".

[50] KIESLING, C.: *Hardware results on recent Exp./Run 10/05825-35-HLT1-f00xxx*. Presentation at Weekly Trigger-Meeting, December 2019.

[51] KIM, C. H., B. G. CHEON, S. H. KIM, I. S LEE, H. E CHO, Y. UNNO, Y. J. KIM, J. K. AHN, E. J. JANG and S. K. CHOI: *Development of Slow Control Package for the Belle II Calorimeter Trigger System*. 2018.

[52] KIM, K., J. KIM and E. WON: *Status of TSFinder*. Presentation at 21st B2GM, 2015.

[53] KIM, K., J. B. KIM and E. WON: *Status of TSF Recent Cosmic Result*. Presentation at 29th B2GM, 2018.

[54] KIM, S., I. LEE, Y. UNNO and B. CHEON: *Status of the Electromagnetic Calorimeter Trigger system at the Belle II experiment*. Journal of Instrumentation, 12(09):C09004, 2017.

[55] KIM, C.: *TRG Archiver(+SLC)*. Presentation at ECLTRG meeting 10th October, October 2018.

[56] KOEHNE, J.: *Realization of a second level neural network trigger for the H1 experiment at HERA*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 389(1):128 – 133, 1997. New Computing Techniques in Physics Research V.

[57] KOGA, T.: *ECL-CDC fastest priority timing*. Presentation at trigger weekly meeting, June 2019.

[58] KOGA, T.: *TRG operation status*. Presentation at 33rd Belle II General Meeting, June 2019.

[59] KONNO, T.: *The Slow Control and Data Quality Monitoring System for the Belle II Experiment*. IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 62, NO. 3,, JUNE 2015.

[60] KRIZHEVSKY, A., I. SUTSKEVER and G. E. HINTON: *ImageNet Classification with*

Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, 2012. Curran Associates Inc.

[61] LAI, Y. T.: *3D input unsynchronizer study*. CDCTRG Meeting September 2017, September 2017.

[62] LAI, Y. T.: *CDCTRG firmware*. Presentation at Belle II Trigger and DAQ Workshop 2019, August 2019.

[63] LAI, Y. T.: *GRL Status*. Presentation at trigger weekly meeting, March 2019.

[64] LAI, Y. T.: *Status of GRL and short tracking*. Presentation at 33rd B2GM, 2019.

[65] LECUN, Y., Y. BENGIO and G. HINTON: *Deep learning*. Nature, 521(7553):436–444, 5 2015.

[66] LEE, I. S., S. H. KIM, C. H. KIM, H. E. CHO, Y. J. KIM, J. K. AHN, E. J. JANG, S. K. CHOI, Y. UNNO and B. G. CHEON: *Progress on the Electromagnetic Calorimeter Trigger Simulation at the Belle II Experiment*. 2018.

[67] LEE, S., R. ITOH, T. HIGUCHI, M. NAKAO, S. Y. SUZUKI and E. WON: *Belle-II High Level Trigger at SuperKEKB*. Journal of Physics: Conference Series, 396(1):012029, 2012.

[68] LEISERSON, C. E. and J. B. SAXE: *Optimizing Synchronous Systems*. Journal of VLSI and Computer Systems, 1983.

[69] LEVIT, D., I. KONOROV and S. PAUL: *FPGA based data read-out system of the Belle II pixel detector*. pp. 1–2, 05 2014.

[70] LI, C.: *Trigger and Data Acquisition Systems at the Belle II Experiment*. Jul 2015.

[71] LIVENTSEV, D.: *KLM TRIGGER STATUS AND PLAN*. Belle II Trigger/DAQ workshop Presentation.

[72] MACCHIARULO, L., X. GAO, K. NISHIMURA and G. VARNER: *A probability-optimized fast timing trigger for the Belle II time of propagation detector*. IEEE Nuclear Science Symposuim & Medical Imaging Conference, 2010.

[73] MENTOR GRAPHICS CORPORATION: *HDL Simulation - ModelSim PE*, January 2018.

[74] MICRON TECHNOLOGY INC.: *Achieve High-Performance Computing in Three Easy Steps*. Techn. Rep., 2017.

[75] M.NAKAO and S. SUZUKI: *Network shared memory framework for the Belle data acquisition control system*. IEEE Conference on Real-Time Computer Applications in Nuclear Particle and Plasma Physics. 11th IEEE NPSS Real Time Conference. Conference Record, 1999.

[76] MODEL TECHNOLOGY INCORPORATED: *ModelSim Foreign Language Interface Reference*. Model Technology Incorporated, 2002.

[77] MOLL, A.: *The Software Framework of the Belle II Experiment*. 2011 J. Phys.: Conf. Ser. 331 032024, 2011.

[78] MOON, H., K. KIM and J. KIM: *Event Time Finder Status*. Presentation at 29th B2GM,

2015.

[79] NAKAMURA, K. et al.: *Review of Particle Physics*. Journal of Physics G: Nuclear and Particle Physics, 37(7A):075021, 2010.

[80] NAKAO, M.: *Timing distribution for the Belle II data acquistion system*. Journal of Instrumentation, 7(01):C01028, 2012.

[81] NAKAO, M., T. HIGUCHI, R. ITOH and S. Y. SUZUKI: *Data acquisition system for Belle II*. Journal of Instrumentation, 5(12):C12004, 2010.

[82] NAKAO, M., M. YAMAUCHI, S. SUZUKI, R. ITOH and R. FUJII: *Data Acquisition System for the Belle Experiment*. IEEE Transactions on Nuclear Science (Volume: 47 , Issue: 2 , Apr 2000), 2000.

[83] NAKAZAWA, H.: *GDL Configuration (Trigger I/O Bits) for Phase 2*, November 2019. https://confluence.desy.de/pages/viewpage.action?pageId=89667646.

[84] NICOL, C.: *A dataflow processing chip for training deep neural networks*. IEEE Hot Chips 29 Symposium, 2017.

[85] NISHIMURA, K.: *The Time-of-propagation counter for Belle II*. Nucl. Instrum. Meth., A639:177–180, 2011.

[86] NISHIMURA, K., T. BROWDER, H. HOEDLMOSER, B. JACOBSON, J. KENNEDY, M. ROSEN, L. RUCKMAN, G. VARNER, A. WONG and W. YEN: *An imaging time-of-propagation system for charged particle identification at a super B factory*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 623(1):297 – 299, 2010. 1st International Conference on Technology and Instrumentation in Particle Physics.

[87] NORUM, W. E.: *Getting started with EPICS on RTEMS*.

[88] NURVITADHI, E., D. SHEFFIELD, J. SIM, A. MISHRA, G. VENKATESH and D. MARR: *Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC*. pp. 77–84, 12 2016.

[89] PARK, S., Y. KWON, M. NAKAO, S. UEHARA and T. KONNO: *Environmental Monitoring for Belle II*. 21st IEEE Real Time Conference (RT2018), 2018.

[90] POHL, S.: *Data Quality Monitoring*. MPI, 2017. https://confluence.desy.de/display/BI/Trigger+NNT+DQM.

[91] POHL, S.: *Neurotrigger algorithm details*. MPI, 2017. https://confluence.desy.de/pages/viewpage.action?pageId=98076080.

[92] POHL, S.: *Track Reconstruction at the First Level Trigger of the Belle II Experiment*. PhD thesis, Ludwig-Maximilians-Universitaet Muenchen, 2017.

[93] PULVERMACHER, C.: *dE/dx Partile identification and Pixel Detector Data Reduction for the Belle II Experiment*. Master's thesis, Karlsruhe Institut of Technology, 2012.

[94] PYTHON SOFTWARE FOUNDATION: *pickle - Pyhton object derialization*, 2018. https://docs.python.org/3/library/pickle.html.

[95] SANDER, J., M. ESTER, H.-P. KRIEGEL and X. XU: *Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications*. Data Mining and Knowl-

edge Discovery, 2(2):169–194, Jun 1998.

[96] SCHNELL, M.: *Development of an FPGA-based Data Reduction System for the Belle II DEPFET Pixel Detector*. PhD thesis, Bonn U., 2015.

[97] SCHUH, T.: *Entwicklung des CMS-Spurtriggers fuer den Hochluminositaetsbetrieb des Large Hadron Colliders*. Dissertation, Karlsruher Institut fuer Technologie, 2017.

[98] SHENG, T.-A.: *2D Tracker in CDC Trigger*. Presentation at Trigger DAQ Workshop, 2016.

[99] SHENG, T.-A.: *Development of 2D Tracker Firmware in CDC Trigger System*. Presentation at 24th B2GM, 2016.

[100] SHENG, T.-A.: *Output format of the 2D tracker*. NTU, 2017.

[101] SHENG, T.-A.: *B2VCD*, March 2018. https://confluence.desy.de/display/BI/B2VCD.

[102] SHILTSEV, V. D.: *High-energy particle colliders: past 20 years, next 20 years, and beyond*. Physics-Uspekhi, 55(10):965–976, oct 2012.

[103] SIMONYAN, K. and A. ZISSERMAN: *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014.

[104] STOROZHEV, S. and A. DUTHOU: *Timing Closure Exploration Tools with SmartXplorer and PlanAhead Tools*. Xilinx Inc., 2009.

[105] SULLIVAN, G.: *Verilog::VCD*, 05 2018. "https://metacpan.org/release/Verilog-VCD".

[106] SUN, D., Z. LIU, J. ZHAO and H. XU: *Belle2Link: A Global Data Readout and Transmission for Belle II Experiment at KEK*. Physics Procedia, 37:1933 – 1939, 2012. Proceedings of the 2nd International Conference on Technology and Instrumentation in Particle Physics (TIPP 2011).

[107] SYNPLICITY INC.: *Synplify Pro Reference Manual*. Synplicity Inc., 2014.

[108] TANIGUCHI, N.: *Central Drift Chamber for Belle-II*. Journal of Instrumentation, 12(06):C06014, 2017.

[109] TENG, Y.-S., C.-H. WANG, S.-M. LIU, J.-G. SHIU, Y.-T. LAI and C.-S. LIN: *The Status of High-Speed Trigger Multiplexer Module with Aurora Protocol Implemented on Arria II FPGA for the Belle II Cylindrical Drift Chamber Detector*. IEEE Nuclear Science Symposium and Medical Imaging Conference, 2013.

[110] UCHIDA, T.: *Readout Electronics for the Central Drift Chamber of the Belle-II Detector*. EEE Transactions on Nuclear Science (Volume: 62 , Issue: 4 , Aug. 2015), 2015.

[111] UMUROGLU, Y., N. J. FRASER, G. GAMBARDELLA, M. BLOTT, P. H. W. LEONG, M. JAHRE and K. A. VISSERS: *FINN: A Framework for Fast, Scalable Binarized Neural Network Inference*. CoRR, abs/1612.07119, 2016.

[112] USHIRODA, Y., A. MOHAPATRA, H. SAKAMOTO, Y. SAKAI, M. NAKAO, Q. AN and Y. WANG: *Development of the central trigger system for the BELLE detector at the KEK B-factory*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 438(2):460 – 471, 1999.

[113] WANG, J., K. YAN, K. GUO, J. YU, L. SUI, S. YAO, S. HAN and Y. WANG: *Real-Time Pedestrian Detection and Tracking on Customized Hardware*. In *Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia*, ESTIMedia'16, pp. 1–1, New York, NY, USA, 2016. ACM.

[114] WON, E.: *A hardware implementation of artificial neural networks using field programmable gate arrays*. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 581(3):816 – 820, 2007.

[115] WON, E., J. B. KIM and B. R. KO: *Three-dimensional fast tracker for the central drift chamber based level-1 trigger system in the Belle II experiment*. J. Korean Phys. Soc., 72(1):33–37, 2018.

[116] WUNSCH, C.: *Pixel Detector Cluster Rescue for the Belle II Experiment*. Master's thesis, Karlsruhe Institut of Technology, 2015.

[117] XILINX INC.: *Virtex-5 FPGA RocketIO GTP Transceiver User Guide UG196*. Xilinx Inc., v2.1 ed., 2009.

[118] XILINX INC.: *XST User Guide - UG627*. Xilinx Inc., 11.3 ed., 2009.

[119] XILINX INC.: *Memory Interface Solutions UG086*. Xilinx Inc., v3.6 ed., 2010.

[120] XILINX INC.: *LogiCORE IP ChipScope Pro Integrated Logic Analyzer*. Xilinx Inc., v1.04a ed., 2011.

[121] XILINX, INC: *PicoBlaze 8-bit Embedded Microcontroller User Guide - UG129*, 2011.

[122] XILINX INC.: *Virtex-6 FPGA GTH Transceivers User Guide UG371*. Xilinx Inc., v2.2 ed., 2011.

[123] XILINX INC.: *Virtex-6 FPGA GTX Transceivers User Guide UG366*. Xilinx Inc., v2.6 ed., 2011.

[124] XILINX, INC: *ISim User Guide - UG660*, 14.1 ed., 2012.

[125] XILINX INC.: *Virtex-6 FPGA Configurable Logic Block*, UG364 v1.2 ed., 2012.

[126] XILINX INC.: *Virtex-6 FPGA Memory Resources*. Xilinx Inc., v 1.8 ed., 2014.

[127] XILINX INC.: *Virtex-6 FPGA Product Table*. Xilinx Inc., 2014.

[128] XILINX INC.: *UltraScale FPGA Product Tables and Product Selection Guide*. Xilinx Inc., 2016.

[129] XILINX INC.: *UltraScale Architecture GTY Transceivers User Guide UG578*. Xilinx Inc., v1.3 ed., 2017.

[130] XILINX INC.: *7 Series Product Tables and Product Selection Guide*, 2018.

[131] XILINX INC.: *Accelerating DNNs with Xilinx Alveo Accelerator Cards*. White Paper, October 2018.

[132] XILINX INC.: *UltraScale Architecture and Product Data Sheet: Overview*. Xilinx Inc., DS890 v3.5 ed., August 2018.

[133] XILINX INC.: *UltraScale+ FPGA Product Tables and Product Selection Guide*. Xilinx Inc., 2018.

[134] XILINX INC.: *VC709 Evaluation Board for the Virtex-7 FPGA UG887*. Xilinx Inc., 2018.

[135] XILINX INC.: *Versal: The First Adaptive Compute Acceleration Platform (ACAP)*. White Paper, October 2018.

[136] XILINX INC.: *Virtex UltraScale+ product brief*. Xilinx Inc., 2018.

[137] ZENG, H., R. CHEN, C. ZHANG and V. PRASANNA: *A Framework for Generating High Throughput CNN Implementations on FPGAs*. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '18, pp. 117–126, New York, NY, USA, 2018. ACM.

[]

Own Publications

[A+17] ABLYAZIMOV, T. et al.: *Challenges in QCD matter physics –The scientific programme of the Compressed Baryonic Matter experiment at FAIR*. The European Physical Journal A, 53(3):60, Mar 2017.

[A+21a] ABUDINÉN, F. et al.: *First search for direct CP-violating asymmetry in $B^0 \to K^0\pi^0$ decays at Belle II*. arXiv preprint arXiv:2104.14871, 4 2021.

[A+21b] ABUDINÉN, F. et al.: *Measurements of branching fractions and direct CP-violating asymmetries in $B^+ \to K^+\pi^0$ and $\pi^+\pi^0$ decays using 2019 and 2020 Belle II data*. In *arXiv preprint arXiv:2105.04111*, 5 2021.

[A+21c] ABUDINÉN, F. et al.: *Search for $B^+ \to K^+\nu\bar{\nu}$ decays using an inclusive tagging method at Belle II*. arXiv preprint arXiv:2104.12624, 4 2021.

[A+21d] ABUDINÉN, F. et al.: *Study of $B \to D^{(*)}h$ decays using* 62.8 fb^{-1} *of Belle II data*. In *55th Rencontres de Moriond on QCD and High Energy Interactions*, 4 2021.

[Ada15] ADAM, D. AND TVERDYSHEV, S. AND ROLFES, C. AND SANDMANN, T. AND BAEHR, S. AND SANDER, O. AND BECKER, J. AND BAUMGARTEN, U.: *Two architecture approaches for mils systems in mobility domains (automobile, railway and avionik)*. In *International Workshop on MILS: Architecture and Assurance for Secure Systems (MILS 2015), 20.01.2015, Amsterdam*, 2015.

[Bae15] BAEHR, S. AND SANDER, O. AND HECK, M. AND PULVERMACHER, C. AND FEINDT, M. AND BECKER, J.: *Online-analysis of hits in the belle-ii pixeldetector for separation of slow pions from background*. Journal of Physics: Conference Series, 664(9):092001, 2015.

[Bae16] BAEHR, S. AND SANDER, O. AND HECK, M. AND FEINDT, M. AND BECKER, J.: *A framework for porting the neurobayes machine learning algorithm to fpgas*. Journal of Instrumentation, 11(01):C01058, 2016.

[Bae17] BAEHR, S. AND SKAMBRAKS, S. AND NEUHAUS, S. AND KIESLING, C. AND BECKER, J.: *A neural network on FPGAs for the z-vertex track trigger in Belle II*. Journal of Instrumentation, 12(03):C03065, 2017.

[Bae18a] BAEHR, S. AND KEMPF, F. AND BECKER, J.: *Data readout triggering for phase 2 of the belle ii particle detector experiment based on neural networks*. In *Proceedings of the 31th IEEE International System-on-Chip Conference (SOCC), Arlington, VA, September 4-7, 2018*, 2018.

[Bae18b] BAEHR, S. AND KEMPF, F. AND BECKER, J: *Data reduction and readout triggering in particle physics experiments using neural networks on fpgas [in press]*. In *Proceedings of the 18th International Conference on Nanotechnology (IEEE-NANO 2018), Cork, IRL, July 23-26, 2018*, 2018.

[Bae19a] BAEHR, S. AND POEHLER, J. AND UNGER, K. AND HOCHSTUHL, A. AND BECKER, J. AND SKAMBRAKS, S. AND MCCARNEY, S. AND MEGGENDORFER, F. AND KIESLING, C.: *Low latency neural networks using heterogenous resources on fpga for the belle ii trigger*. In *2019 eConf Proceedings of Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019)*, 2019.

[Bae19b] BAEHR, S. AND UNGER, K. AND BECKER, J. AND SKAMBRAKS, S. AND MCCARNEY, S. AND MEGGENDORFER, F. AND KIESLING, C.: *Online estimation of particle track parameters based on neural networks for the belle ii trigger system*. In *Electrontic Conference Proceedings of the Symposium Artificial Intelligence for Science, Industry and Society*, 2019.

[Bap15] BAPP, F. K. AND SANDER, O. AND SANDMANN, T. AND VU DUY, V. AND BAEHR, S. AND BECKER, J.: *Adapting commercial off-the-shelf multicore processors for safety-related automotive systems using online monitoring*. In *SAE 2015 World Congress & Exhibition*. SAE International, apr 2015.

[Bel19] BELLE II COLLABORATION AND BAEHR, S. ET AL.: *Search for an invisibly decaying z' boson at belle ii in $e^+e^- \to \mu^+\mu^-(e^\pm\mu^\mp)$ + missing energy final states*. Physical review letters 124 (14), 141801, 2019.

[Bel20a] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. AND OTHERS: *Measurement of the integrated luminosity of the phase 2 data of the belle II experiment*. Chinese Physics C, 44(2):021001, jan 2020.

[Bel20b] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. AND OTHERS: *Studies of the semileptonic $\bar{b}^0 \to d^{*+}\ell^-\bar{\nu}_\ell$ and $b^- \to d^0\ell^-\bar{\nu}_\ell$ decay processes with 34.6 fb^{-1} of belle ii data*. arXiv:2008.07198, 2020.

[Bel20c] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *A calibration of the Belle II hadronic tag-side reconstruction algorithm with $B \to X\ell\nu$ decays*. arXiv preprint arXiv:2008.06096, 8 2020.

[Bel20d] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Charmless B decay reconstruction in 2019 Belle II data*. arXiv preprint arXiv:2005.13559, 5 2020.

[Bel20e] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Exclusive $b^0 \to \pi^-\ell^+\nu_\ell$ decays with hadronic full event interpretation tagging in 34.6 fb^{-1} of belle ii data*. arXiv:2008.0881, 2020.

[Bel20f] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *First flavor tagging calibration using 2019 belle ii data*. DOI:10.3204/PUBDB-2020-02972, 2020.

[Bel20g] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Measurement of hadronic mass moments $\langle m_X^n \rangle$ in $b \to x_c\ell\nu$ decays at belle ii*. arXiv:2009.04493, 2020.

[Bel20h] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Measurement of the B^0 lifetime using fully reconstructed hadronic decays in the 2019 Belle II dataset*. arXiv preprint arXiv:2005.07507, 5 2020.

[Bel20i] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Mea-*

surement of the branching fraction $\mathcal{B}(\bar{B}^0 \to D^{*+}\ell^-\bar{\nu}_\ell)$ with early Belle II data. arXiv preprint arXiv:2004.09066, 4 2020.

[Bel20j] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Measurements of branching fractions and cp-violating charge asymmetries in charmless b decays reconstructed in 2019–2020 belle ii data*. arXiv:2009.09452, 2020.

[Bel20k] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Rediscovery of $b \to \phi k^{(*)}$ decays and measurement of the longitudinal polarization fraction f_l in $b \to \phi k^*$ decays using the summer 2020 belle ii dataset*. arXiv:2008.03873, 2020.

[Bel20l] BELLE II COLLABORATION AND ABUDINEN, F. AND BAEHR, S. ET AL.: *Search for axionlike particles produced in e^+e^- collisions at belle ii*. Phys. Rev. Lett., 125:161806, Oct 2020.

[K$^+$18] KOU, E. et al.: *The belle ii physics book*. INT-PUB-18-047, 2018.

[Lai18] LAI, Y.-T. AND BAEHR, S. AND CHANG, M. C. AND IWASAKI, Y. AND KIM, J. B. AND KIM, K. T. AND KIESLING, C. AND LU, P. C. AND LIU, S. M. AND MOON, H. K. AND MOON, T. J. AND NAKAZAWA, H. AND SHIU, J. G. AND SHENG, T.-A. AND WANG, C. H. AND WON, E.: *Level-1 track trigger with central drift chamber detector in belle ii experiment*. NSS/MIC, 2018.

[Lai20] LAI, Y.-T. AND AOYAMA, M. AND BAEHR, S. AND CHANG, M. C. AND IWASAKI, Y. AND KIM, J. B. AND KIM, K. T. AND KIESLING, C. AND LU, P. C. AND LIU, S. M. AND MOON, H. K. AND MOON, T. J. AND NAKAZAWA, H. AND SHIU, J. G. AND SHENG, T.-A. AND WANG, C. H. AND WON, E.: *Development of the level-1 track trigger with central drift chamber detector in belle ii experiment and its performance in superkekb 2019 phase 3 operation*. In *INSTR20: Instrumentation for Colliding Beam Physics : JINST*, 2020.

[McC19] MCCARNEY, S. AND BAEHR, S. AND KIESLING, C. AND MEGGENDORFER, F. AND SKAMBRAKS, S. AND VAN TONDER, R.: *Optimizing the first level neural network z-trigger for the drift chamber at the belle ii experiment*. In *2019 Talk at Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2019)*, 2019.

[Meg20] MEGGENDORFER, F. AND SKAMBRAKS, S. AND BAEHR, S. AND UNGER, K. AND BECKER, J. AND KIESLING, C.: *Performance of the z trigger under luminosity conditions: First experience*. In *2020 Talk at Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2020)*, 2020.

[Pfa18] PFAU, J. AND FIGULI, S. P. AND BAEHR, S. AND BECKER, J.: *Reconfigurable fpga-based channelization using polyphase filter banks for quantum computing systems*. In *Applied Reconfigurable Computing - Architectures, Tools, and Applications, Proceedings of the 14th International Symposium, ARC 2018, Santorini, Greece, 2nd - 4th May 2018. Ed.: Nikolaos Voros*, volume 10824 of *Lecture Notes in Computer Science*, pages 615–626. Springer, Cham, 2018.

[Rei15] REINHARDT, D. AND ADAM, D. AND LUBBERS, E. AND AMARNATH, R. AND SCHNEIDER, R. AND GANSEL, S. AND SCHNITZER, S. AND HERBER, C. AND SANDMANN, T. AND MICHEL, H. U. AND KAULE, D. AND OLKUN, D. AND REHM, M. AND HARNISCH, J. AND RICHTER, A. AND BAEHR, S. AND SANDER, O. AND BECKER, J. AND BAUMGARTEN, U. AND THEILING, H.: *Embedded vir-*

tualization approaches for ensuring safety and security within e/e automotive systems. In *Embedded World Conference, Nürnberg, February 24 - 26, 2015*, 2015.

[San14a] SANDER, O. AND AND SANDMANN, T. AND VU DUY, V. AND BAEHR, S. AND BAPP, F. K. AND BECKER, J. AND MICHEL, H. U. AND KAULE, D. AND ADAM, D. AND LUEBBERS, E. AND HAIRBUCHER AND RICHTER, A. AND HERBER, C. AND HERKERSDORF, A.: *Hardware virtualization support for shared resources in mixed-criticality multicore systems*. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.

[San14b] SANDER, O. AND BAEHR, S. AND LUEBBERS, E. AND SANDMANN, T. AND VU DUY, V. AND J. BECKER: *A flexible interface architecture for reconfigurable coprocessors in embedded multicore systems using pcie single-root i/o virtualization*. In *2014 International Conference on Field-Programmable Technology (FPT)*, pages 223–226, Dec 2014.

[San14c] SANDER, O. AND BAPP, F. K. AND SANDMANN, T. AND VU DUY, V. AND BAEHR, S. AND J. BECKER: *Architectural measures against radiation effects in multicore soc for safety critical applications*. In *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 663–666, Aug 2014.

[Ska19] SKAMBRAKS, S. AND BAEHR, S. AND KIESLING, C. AND MCCARNEY, S. AND MEGGENDORFER, F. AND VAN TONDER, R.: *A 3d track finder for the belle ii cdc l1 trigger*. Journal Of Physics: Conference Series, 2019.

[Ska20] SKAMBRAKS, S. AND BAEHR, S. AND UNGER, K. AND BECKER, J. AND MEGGENDORFER, F. AND KIESLING, C.: *A machine learning based 3d track trigger for belle ii*. In *2020 Talk at Connecting the Dots and Workshop on Intelligent Trackers (CTD/WIT 2020)*, 2020.

[Ung20] UNGER, K. AND BAEHR, S. AND IWASAKI, Y. AND KIM, K. T. AND LAI, Y.-T. AND BECKER, J.: *Realization of a state machine based detection for track segments in the trigger system of the belle ii experiment*. Proceedings of Science, TWEPP2019, 2020.

[Vu 14a] VU DUY, V. AND SANDER, O. AND SANDMANN, S. AND BAEHR, S. AND HEIDELBERGER, J. AND BECKER, J.: *Enabling partial reconfiguration for coprocessors in mixed criticality multicore systems using pci express single-root i/o virtualization*. In *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, pages 1–6, Dec 2014.

[Vu 14b] VU DUY, V. AND SANDMANN, T. AND BAEHR, S. AND SANDER, O. AND BECKER, J.: *Virtualization support for fpga-based coprocessors connected via pci express to an intel multicore platform*. In *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, pages 305–310, May 2014.

[Vu 15] VU DUY, V. AND SANDER, O. AND SANDMANN, T. AND HEIDELBERGER, J. AND BAEHR, S. AND BECKER, J.: *On-demand reconfiguration for coprocessors in mixed criticality multicore systems*. In *7th International Workshop on Dependable Many-Core Computing (DMCC 2015), Amsterdam, July 20 – July 24, 2015*, 2015.

[]

Supervised Student Research

[Gao19] GAO, Z.: *Flexible parameterization and time persistence for a 3D-Hough Map on FPGAs*. Bachelor's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2019.

[Hoc18] HOCHSTUHL, A.: *A Hough-based 3D-Track-Finder for the Belle II Particle Collider Experiment*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.

[Hua19] HUANG, J.: *An Integrated Estimation of 3D-Track Parameter for the Belle II Particle Accelerator Experiment*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2019.

[Ma19] MA, Y.: *Investigation and Implementation of Soft Error Mitigation and Detection for Belle II Trigger Modules*. Bachelorarbeit, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2019.

[Ohn15] OHNEZAT, N.: *Development and Evaluation of a flexible architecture for big data applications for embedded systems*. Diploma Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2015.

[Pfa17] PFAU, J.: *Scalable FPGA-based Channelization using Polyphase Filter Banks*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2017.

[Poe18] POEHLER, J.: *Optimierung der neuronalen Netze fuer den z-Vertex Track Trigger in Phase 3 des Belle II Teilchenexperiments*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.

[Reu18] REUTER, T.: *Interfacing and Data Quality Management for the z-vertex Trigger of Belle II*. Bachelor's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.

[Rin18] RING, J.: *Adaptation of the z-Vertex Trigger to VC709 Board and Creation of a Demonstrator of the Belle II CDC DAQ*. Bachelor's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.

[Ste19] STEINHILPER, T.: *Entwurf eines Interfaces als integrierte Hochspannungs-CMOS Schaltung fuer Kfz-Generatorspannungsregler mit programmierbarer Funktion*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2019.

[Ung18] UNGER, K.: *A State Machine based Track Segement Finder for the Belle II Particle Collider Experiment*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.

[Wu16] WU, X.: *Exploration and Development of Neural Networks on FPGAs for the Z-Vertex*

Trigger of Belle II;. Bachelor's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2016.

[Zha18] ZHANG, R.: *Evaluation of Binary Neural Networks on FPGAs for Belle 2 Trigger Systems*. Master's Thesis, Karlsruhe Institute of Technology, Institute for Information Processing Technologies, 2018.