FERMILAB-PUB-21-217-CMS-E-SCD (accepted) DOI:10.1109/TNS.2021.3087100

A reconfigurable neural network ASIC for detector front-end data compression at the HL-LHC

Giuseppe Di Guglielmo, Farah Fahim, Christian Herwig, Manuel Blanco Valentin, Javier Duarte, Cristian Gingu, Philip Harris, James Hirschauer, Martin Kwok, Vladimir Loncar, Yingyi Luo, Llovizna Miranda, Jennifer Ngadiuba, Daniel Noonan, Seda Ogrenci-Memik, Maurizio Pierini, Sioni Summers, Nhan Tran

***Abstract*—Despite advances in the programmable logic capabilities of modern trigger systems, a significant bottleneck remains in the amount of data to be transported from the detector to off-detector logic where trigger decisions are made. We demonstrate that a neural network autoencoder model can be implemented in a radiation tolerant ASIC to perform lossy data compression alleviating the data transmission problem while preserving critical information of the detector energy profile. For our application, we consider the high-granularity calorimeter from the CMS experiment at the CERN Large Hadron Collider. The advantage of the machine learning approach is in the flexibility and configurability of the algorithm. By changing the neural network weights, a unique data compression algorithm can be deployed for each sensor in different detector regions, and changing detector or collider conditions. To meet area, performance, and power constraints, we perform a quantization-aware training to create an optimized neural network hardware implementation. The design is achieved through the use of high-level synthesis tools and the `hls4ml` framework, and was processed through synthesis and physical layout flows based on a LP CMOS 65 nm technology node. The flow anticipates 200 Mrad of ionizing radiation to select gates, and reports a total area of 3.6 mm^2 and consumes 95 mW of power. The simulated energy consumption per inference is 2.4 nJ. This is the first radiation tolerant on-detector ASIC implementation of a neural network that has been designed for particle physics applications.**

***Index Terms*—ASIC, artificial intelligence, autoencoder, LHC, machine learning, SEE mitigation, high-level synthesis, hardware accelerator**

Manuscript received March 26, 2021; (Corresponding e-mail: farah@fnal.gov)

Farah Fahim, Cristian Gingu, Christian Herwig, James Hirschauer, Llovizna Miranda, Nhan Tran are with Fermi National Accelerator Laboratory, Batavia, IL, USA and are supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy (DOE), Office of Science, Office of High Energy Physics

Manuel Blanco Valentin, Farah Fahim, Yingyi Luo, Seda Ogrenci-Memik, Nhan Tran are with Northwestern University, Evanston, IL, USA

Giuseppe Di Guglielmo is with Columbia University, New York, NY, USA

Javier Duarte is with UC San Diego, La Jolla, CA, USA and is supported by the DOE, Office of Science, Office of High Energy Physics Early Career Research program under Award No. DE-SC0021187.

Philip Harris is with Massachusetts Institute of Technology, Cambridge, MA, USA and is supported by a Massachusetts Institute of Technology University grant.

Martin Kwok is with Brown University, Providence, RI, USA

Jennifer Ngadiuba is with California Institute of Technology, Pasedena, CA, USA

Daniel Noonan is with Florida Institute of Technology, Melbourne, FL, USA

Vladimir Loncar, Maurizio Pierini, Sioni Summers are with CERN, Geneva, Switzerland and are supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369).

Vladimir Loncar is with the Institute of Physics Belgrade, Serbia

I. Introduction

BREAKTHROUGHS in the precision and speed of sensing instrumentation are impactful on advances in scientific methodologies and theories. Thus, a common paradigm across many scientific disciplines in physics has been to increase the resolution of the sensing equipment in order to increase either the robustness or the sensitivity of the experiment itself. This demand for increasingly higher sensitivity in experiments, along with advances in the design of state-of-the-art sensing systems, has resulted in rapidly growing big data pipelines such that transmission of acquired data to the processing unit via conventional methods is no longer feasible. Data transmission is commonly much less efficient than data processing. Therefore, placing data compression and processing as close as possible to data creation while maintaining physics performance is a crucial task in modern physics experiments.

At the CERN Large Hadron Collider (LHC) and its high luminosity upgrade (HL-LHC), extreme collision rates present extreme challenges for data processing and transmission at multiple stages in detector readout and trigger systems. As the initial stage in the data chain, the on-detector (front-end) electronics that read out detector sensors must operate with low latency and low power dissipation in a high radiation environment, necessitating the use of application-specific integrated circuits (ASICs). In order to mitigate the initial bottleneck of moving data from front-end ASICs to off-detector (back-end) systems based on field-programmable gate arrays (FPGAs), front-end ASICs must provide edge computing resources to efficiently use limited bandwidth through real-time processing and identification of interesting data. Front-end data compression algorithms have historically relied on zero-suppression, threshold-based selection, sorting or summing of data.

Artificial intelligence (AI), and more specifically machine learning (ML), has recently been demonstrated to be a powerful tool for data compression, processing, and analysis in physics [1–4] and many other domains. While progress has been made towards generic real-time processing through inference including boosted decision trees and neural networks (NNs) using FPGAs in off-detector electronics [5, 6], ML methods have not yet been used to address the significant bottleneck in the transport of data from front-end ASICs to back-end FPGAs.

The high-granularity endcap calorimeter (HGCAL) [7] currently under construction by the CMS experiment [8] for eventual use at HL-LHC provides an excellent example of the big data challenges facing high energy physics. As an *imaging calorimeter*, the HGCAL includes over 6 million readout channels, providing an unprecedented level of segmentation for calorimetry at high-energy colliders. In order to provide input to the real-time event filtering (trigger) system of CMS, the HGCAL transmits a stream of trigger data at a frequency of 40 MHz resulting in massive data rates. At data creation, two ASICs are used to digitize and encode trigger data before transmission to back-end FPGAs for further processing.

In this paper, we explore the application of ML algorithms to the task of processing large amounts of data with low latency and low power in a high radiation environment in order to maximize efficient use of limited bandwidth. We focus on an ASIC implementation of an autoencoder algorithm that uses a configurable NN to efficiently compress and encode data automatically before transmission. Subsequent stages of data processing can either decode the data or continue analyzing the encoded data. In our ASIC implementation, the NN architecture is fixed, but exceptional flexibility in application is preserved by making the NN *weights* programmable. We apply our methodology to the specific front-end data transmission challenge of the CMS HGCAL, showing that the advantage of our approach lies in the flexibility and configurability of the algorithm, which allows us to generate unique data compression algorithms depending on HGCAL sensor geometry, sensor location on the detector and the corresponding occupancy and signal patterns, changing accelerator conditions, or changing detector conditions.

The remainder of this paper is organized as follows. In Section II, we introduce the HGCAL challenge in greater detail and outline our conceptual approach. Then, in Section III, we elaborate on the design and training of the autoencoder NN for the specific case of the CMS HGCAL detector. In Section IV, we present the digital implementation of the trained NN in the ASIC. Finally, we summarize our work and discuss future directions in Section V.

II. System Constraints and Concept

The HGCAL is a major upgrade of the CMS endcap calorimeter planned to for the HL-LHC and provides a fitting demonstrator for the ASIC ML accelerator technology. The HGCAL is described in detail in Ref. [7]; relevant implementation details that have changed since publication of Ref. [7] are updated in this article.

This “imaging calorimeter,” which includes over 600 m^2 of active silicon and over 6 million readout channels, is composed of 50 layers of active shower-sampling media interleaved with dense absorber. The active medium of the 28-layer front electromagnetic compartment is silicon, while the 22-layer rear hadronic compartment includes both silicon and plastic scintillator. Silicon layers are tiled with 8” hexagonal sensor modules, with each module including 48 logical trigger cells (TC) arranged in three 4×4 matrices as shown in Fig. 1. While the NN can be configured for both the silicon and scintillator geometries, the silicon geometry is used throughout this paper to illustrate the concepts.

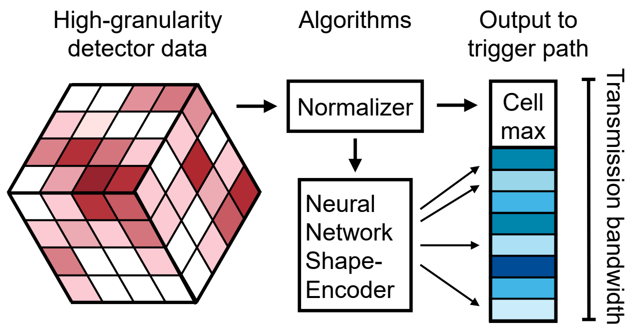


Fig. 1. Simplified version of the internal flow of the autoencoder compression task which takes the module energy deposits, normalizes them to the sum of the energy in the module, and then performs shape encoding

To provide input to the CMS trigger system, data must be transmitted from the on-sensor analog-to-digital ASICs to the all-FPGA back-end detector electronics system at the nominal HL-LHC collision rate of 40 MHz. Because bandwidth constraints prohibit transmission of data for all 48 TCs at 40 MHz, a front-end concentrator ASIC (ECON-T) is being developed to compress a single sensor's information before transmission to the back-end trigger electronics. Each sensor module produces 7 bits of floating-point charge data for each of the 48 TCs at 40 MHz. Thus, the lossy compression task of the ECON-T ASIC is to aggregate the 48 7-bit signals from a sensor and compress the data into a range spanning 48–144 bits while maximally preserving the energy pattern of the sensor. The range of the output bits depends on the location of the sensor module in the detector and the number of links available for a given ECON-T ASIC to transmit the data. The number of links allocated will roughly correspond to the average sensor occupancy, which varies by two orders of magnitude over all sensor locations. The exact task depends on handling of data framing and TC address information as well as on whether the ECON-T algorithm operates with fixed or variable latency. The ECON-T design provides the user a choice among three expert algorithms for TC compression including TC threshold application, sorting and selection of highest energy TC, and aggregation of adjacent TCs. Unused algorithms are clock-gated to conserve power.

The ECON-T ASIC is being developed for the LPCMOS (Low Power CMOS) 65 nm feature size technology and is under active development for CMS. Because it is located on-detector in a high radiation environment, its design also requires tolerance to single event effects. The allocated footprint for the fabrication of this chip is approximately $5\times5\,\mathrm{mm}^2$. It is expected that sufficient area will be available for potential inclusion of our NN compression logic with an approximate area of 4 mm^2. The constraints for the compression algorithms are that they should accept new input data at 40 MHz and complete processing in 50 ns. The power budget of the task is less than 100 mW.

Our contribution is a NN to perform the ECON-T compression task. It is a central tenet of our design that the compression algorithm is *reconfigurable*. Because we are implementing a design for an ASIC, the architecture of the NN will be fixed, but the *weights* of the NN need to be configurable such that the algorithm itself is adaptable. This has several advantages. Through reconfiguration, we will be able to:

- *enable* more computationally complex compression algorithms, which could improve overall physics performance or allow more flexible algorithms;
- *customize* the compression algorithm of each sensor based on their location within the detector;
- *adapt* the compression algorithm for changing detector and collider conditions (for example, if the detector loses a channel or has a noisy channel it can be accounted for or if the collider has more pileup than expected, the algorithm can be adjusted to deal with new or unexpected conditions without catastrophic failure).

For our compression algorithm, we choose to utilize an autoencoder architecture. It provides a generic and flexible compression solution, consisting of two NNs: an encoder and a decoder. The encoding network maps inputs to an intermediate *latent* representation with lower dimensionality than the space of inputs, after which the decoding network aims to recover the original signal. In the HGCAL application, the encoding NN would compress HGCAL data on the ASIC before transmission to the calorimeter trigger FPGAs for subsequent decoding. Ultimately, in a final realistic system, we do not anticipate using a full autoencoder architecture because FPGA resources on the back-end FPGA system will not be sufficient to do a full decoding for every sensor. However, in the absence of understanding how best to use the latent representation later in the processing chain, we optimize performance for an autoencoder because it is a reasonable proxy for the encoder NN encapsulating the salient sensor features such that the image can be decoded from the latent representation. Finally, in Fig. 1, the compression task is split into two parts: an overall normalization over the entire sensor to preserve the total energy in the sensor and the NN *shape* encoder, which encodes the energy pattern across the sensor.

For the automated design tool flow, it is very important to have a rapid co-design loop between the NN algorithm training and the implementation in hardware in order to understand whether the algorithm is meeting system constraints for power, area, and performance simultaneously. To achieve this, we use `hls4ml` [5] which translates NNs trained in common open-source ML software frameworks into RTL using high-level synthesis (HLS) tools [9]. The efficacy of this approach will be described in greater detail in the following section.

III. Algorithm Design and Performance

Our task is to design an algorithm that will reproduce the energy pattern in the sensor while simultaneously adhering to hardware constraints, i.e., fitting in the available area within the ECON-T ASIC chip while complying with system latency and power constraints. Because we are training the algorithm based on a single sensor's energy pattern, we will not be able to optimize for multi-sensor physics performance, such as particle energy resolution. Ultimately, the physics performance may determine the final system optimization, however, it is beyond the scope of this study. Therefore, our target is to design an algorithm that reproduces the original sensor energy pattern as accurately as possible through the autoencoder compression-and-decompression bottleneck.

There are a number of elements needed to design our compression algorithm: a sample of events for training and validation, a preprocessing and normalization block, an optimized NN architecture, and metrics for evaluating the NN performance, both for determining the training loss and the final network evaluation. An essential aspect of the training procedure is quantization-aware training (QAT), i.e., we approximate bit-accurate reduced precision for all of the NN calculations during training. QAT is known to be much more performant than post-training quantization (PTQ), where the training is done using 32-bit floating-point operations, which are then truncated post-training to fixed-point or integer representations. In a previous study of the `QKeras` [10] tool, QAT was performed for an LHC trigger task. It was found that with PTQ, the minimum bit width possible without loss of performance was 14 bits while with QAT, the same performance could be achieved with 6-bit weights. Thus, PTQ would lead to more than 4-fold increase in the area of an ASIC design based on the bit operations hardware design metric [11]. Therefore, we use `QKeras` for training the NNs in this study.

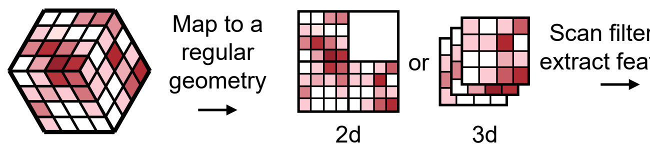
Training Sample: Test energy patterns in the sensors are simulated using top-quark-pair events overlaid with 200 simultaneous collisions per bunch crossing in the CMS software framework [12]. These simulated events create a sample of typical energy patterns in the HGCAL sensors, which we use as a realistic proxy for the sensor data.

Preprocessing: The compression task is factorized into normalization and shape extraction components, as illustrated in Fig. 1. The first stage of ECON-T processing for all algorithms is to expand the 7-bit floating-point TC data to the inherent 22-bit fixed-point TC data. The sum value of all 48 TC is identified and used to normalize the charge distribution across the full sensor (and the sum of TC charge is included in the final data payload to allow subsequent interpretation of normalized TC data). The normalized NN inputs are truncated to 8 bits to allow a more compact NN implementation, while ensuring that any omitted cells constitute less than 1% of the total energy recorded within a module.

NN Architecture: The encoding NN architecture consists of successive layers that sequentially process the input data. Convolutional layers are used to extract spatial features from images through the application of filters: matrices of configurable parameters. While convolutional layers use relatively few parameters, convolution requires many multiply-accumulate operations (MACs). Conversely, a fully-connected layer multiplies a vector of input elements by a matrix of configurable weights, generally requiring more configurable parameters and fewer MACs than a convolution. The impact of the choice of precision for all internal parameters (constrained by the available area on chip) is accounted for by training inherently quantized models with the `QKeras` package [10]. Because the HGCAL sensor data compression task takes as input an image data representation, we consider a convolutional NN layer as a natural approach. Typical convolutions rely on the input being in a Cartesian representation, though

NN encoding layers

- Highly-efficient design is necessary to achieve acceptable design footprint and power consumption
  - Borrow from image processing: convolutional NN (CNN) with filters trained to identify physical radiation patterns



- A fully-connected NN layer achieves further reduction, exploiting correlated features across the image
  - CNN operations scale as ~ (Image * Filter) volumes
  - Dense layer scales with encoder output dimension

other
hexag
ment
hardv

Fig. 2. Mapping the hexagonal sensor geometry to potential Cartesian representations for convolutional layer operations.

Training and Performance Metrics: The performance of the autoencoder is based on how well the original image is reproduced after encoding and decoding. We quantify the difference between raw and decoded HGCAL data with the energy mover's distance (EMD) [13]. Given a particular normalized energy distribution, one may physically relocate some energy fraction dE by a spatial distance dx, leading to a new distribution with an associated rearrangement cost $dEdx$. This notion can be extended to define an "optimal transport" between two energy distributions $\mathcal{A}$ and $\mathcal{B}$, as a re-mapping that minimizes this total rearrangement cost, EMD($\mathcal{A}, \mathcal{B}$). While the performance of the autoencoder is assessed with EMD, taking into account the complete hexagonal sensor geometry, this metric is not used directly in the algorithm training, as it involves nondifferentiable and computationally intensive operations. Models are instead trained with a modified χ^2 loss function incorporating cell-to-cell distances, as a fast approximation of EMD. Specifically, individual TCs are re-summed into all physical groups of approximately 2×2 and 3×3 "super-cells" based on the full hexagonal cells with corresponding χ^2 values computed for the coarsened images. The total loss sums each such χ^2 together, resulting in a comparatively lenient penalty when mis-reconstruction occurs only on small spatial scales. Including these additional χ^2 terms in the training procedure is found to yield significant improvements to the autoencoder performance, as measured with EMD. To ensure an unbiased NN optimization, the data are randomly partitioned into separate samples for training (80%) and validation (20%), with training termination set by the loss observed in the validation sample.

Baseline Encoder Model

A simple encoding NN with a single convolutional and dense layer architecture is investigated. Normalized inputs from hexagonal sensors are arranged into three arrays of 4×4 to form a regular geometry. The convolution layer consists of eight $3 \times 3 \times 3$ kernel matrices, giving a $8 \times 4 \times 4$ output after convolution. It was found that more than eight kernel matrices brought negligible performance improvement. These 128 values are flattened and fed through a dense layer to yield 16 9-bit output values. ReLU activations [14, 15] are applied before and after the dense layer. This leads to a total of 2,288 weight parameters (dominated by the 2,064 parameters used to configure the dense layer), each of which are specified with

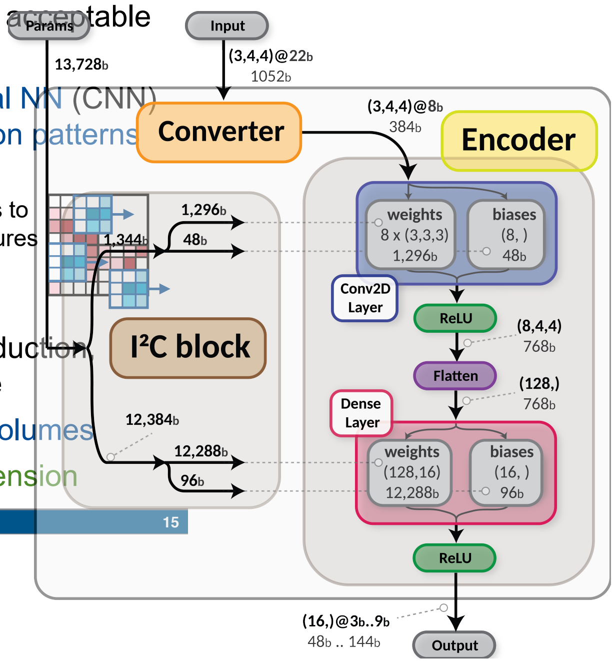


Fig. 3. The autoencoder neural network architecture and data flow for the baseline encoder model

6-bit precision. A single inference requires a total of 4,448 MACs, with similar requirements from the convolution (2,400) and dense layers (2,048). The size and complexity of this baseline model are constrained by area, on-chip memory and interfaces, and power, which impose additional optimization considerations. The encoder architecture with the reconfigurable weights is illustrated in Fig. 3

Optimization Considerations and Comparisons

While the presence of a single convolutional layer is critical for good physics performance of the algorithm (approximated by the EMD between input and decoded images), adding more filters or additional convolutional layers only weakly improves physics performance, at the expense of significantly increased area. Changes in the number and size of the dense layers yield more dramatic differences.

Figure 4 shows a sweep over the number of dense layer outputs, where remaining aspects of the design are fixed based on hardware constraints: the precision of outputs and weights are coherently varied to ensure that both the total number of outputs and the weight bits are fixed. Architectures featuring many outputs with lower relative precision consistently outperform their counterparts. The autoencoder is robust across a variety of conditions and performs well in the high-occupancy regime, which poses the greatest challenge for trigger reconstruction.

Reconfigurability: Figure 4 also demonstrates how the same NN encoder can be re-optimized and configured for new data-taking conditions, by comparing sensors in detector regions requiring low- and high-throughput. The maximum data throughput of 144 bits from 16 9-bit outputs can be reduced through fully configurable selective truncation. Expected use cases include transmission of (48, 80, 112, 144) bits from

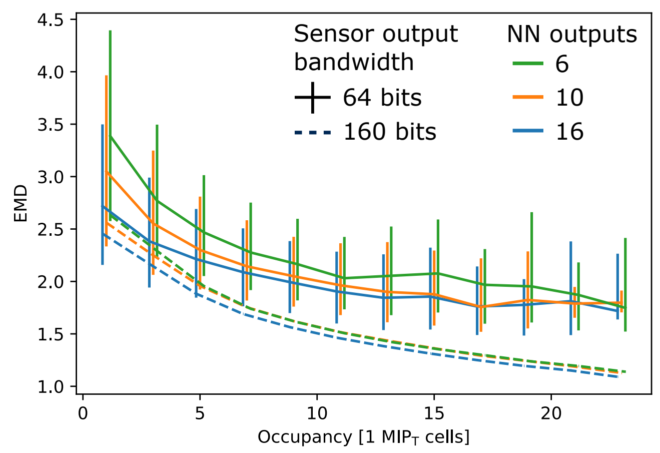


Fig. 4. Median EMD for decoded HGCAL images from the validation dataset, as function of sensor occupancy for six NN configurations. Vertical lines (suppressed for the 160-bit configurations) denote 68% EMD intervals. Occupancy is defined as the number of TCs with signals exceeding one minimum ionizing particle divided by cosh η where η is the pseudorapidity of the TC. (Results shown for version of NN with maximum of 10 bits for each of 16 outputs rather than 9 bits as described in the text.)

TABLE I
AREA BREAKDOWN FOR PIPELINED IMPLEMENTATIONS. THE RESULTS ARE FROM CATAPULT HLS ESTIMATIONS AND AREAS ARE IN μm^2.

Initiation Interval	Total Area	Register Area	MUX Area
1	1,138,242	925	0
2	891,195	5,228	12,989
4	765,877	8,503	16,089
8	699,988	8,509	16,252

16 (3, 4, 7, 9) bit outputs, though the network can also be configured to transmit fewer than 16 outputs, or a mix of precisions.

IV. Implementation Methodology and Results

In this section, we detail the implementation of the trained NN described in Sec. III in the ECON-T ASIC. We discuss the design and verification flow, the architectural and design exploration, steps required for deployment in a radiation environment, design performance metrics, and finally the implementation results.

Algorithm to Accelerator Development

For our design flow, we adopted the `hls4ml` framework [5] to automate the mapping of ML models onto reconfigurable logic. For this work, we extended `hls4ml` to our ASIC flow. Traditionally, hardware designers utilize hardware description languages (HDLs) and a level of abstraction known as the Register Transfer Level (RTL). In recent years, HLS has become an alternative for generating hardware modules from code written in programming languages such as C/C++. HLS comes with significant benefits: it raises the level of abstraction and reduces the simulation time; it simplifies the verification phases; and finally, it makes the exploration and evaluation of design alternatives easier. The original flow of `hls4ml` generates state-of-the-art synthesizable C++ code and HLS directives from the ML-model specifications. The generated code is then fed into the Vivado HLS tool to generate an accelerator in HDL RTL code for the deployment on Xilinx FPGAs [16]. We extended `hls4ml` to support Mentor's Catapult HLS [17] tool and target our specific 65 nm LP CMOS technology for ASIC fabrication. We integrated the HLS-generated code with a SystemVerilog RTL IP of the programmable I^{2}C peripheral[1]. We finally created a component database and layout to be incorporated into the ECON-T ASIC top-level assembly using a digital implementation flow. The standard flow was modified to accommodate automatic triple modular redundancy implementation for HLS-generated RTL integrated with other SystemVerilog modules.

We complemented our design flow with a robust validation and verification methodology across the various refinement steps. We validated the C++ HLS code against the `QKeras` trained model to guarantee the model's functional correctness. Earlier in the design flow, we also performed dynamic and static verification of the synthesizable specifications: we checked design rules with static analysis of the C++ HLS code (Mentor CDesignChecker [19]), measured coverage metrics (Mentor CCov [19]), and finally, ran simulation-based equivalence checking. For the HLS-generated RTL code, we followed a more traditional simulation-based verification to ensure optimized power, area, and speed.

Architectural Exploration

`hls4ml` coupled with the industry standard Catapult HLS (ver. 10.6) tool allowed us to explore the cost and performance trade-offs of various micro-architectural hardware implementations for our ML model. We decided on a pipelined implementation for our accelerator to increase concurrent execution as an early design decision. A pipelined design can process new inputs every N clock cycles, where N is the initiation interval (II) of the design. Table I shows the area breakdown for different II values (1, 2, 4, 8). It is noticeable that although the area is higher for II$=1$, the required resources are mostly functional logic to implement a highly-parallel datapath, i.e. there are no multiplexers. A higher II value implies less design parallelism and more functional-resource reuse. This choice reduces the overall area, but the resource breakdown shows an increase in control logic (MUX) and registers. An II of 1 was ultimately selected so that new inputs may be processed in sync with a single clock operating at 40 MHz LHC crossing frequency.

We used a fixed-point representation (`ac_fixed` [20]) for the input, intermediate, and output parameters of our ML model designed with `hls4ml`. This choice provided us with a high degree of flexibility for exploring the area and accuracy trade-off of the ML-model hardware implementations obtained with HLS. The RTL schematics for the encoder block are shown in Fig. 5. The basic structure of the convolution and dense layers can be seen at the schematic level. The top right and bottom right diagrams are zoomed in portions of this schematic depicting the output and MACs.

[1]The authors use *controller/peripheral* in place of *master/slave* when referring to such I^{2}C devices or processes [18].

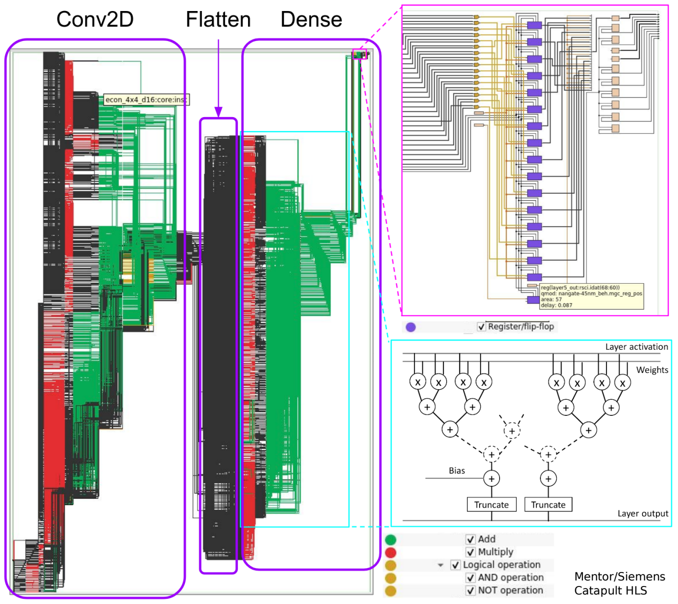


Fig. 5. Encoder RTL Schematics, the basic structure of the convolution and dense layers can be seen at the schematic level on the left and zoomed in images are provided for the output and MAC portions on the right

Digital Implementation in a Radiation Environment

The digital design consists of three major functional blocks: (i) A converter which is a classical module designed with HLS; (ii) An encoder, which uses `hls4ml`; (iii) and an I^{2}C peripheral which uses a SystemVerilog RTL code. The converter is used for normalizing the 48 (22 b) inputs to 48 (8 b). An encoder is used for data classification and further compression to 16 (9 b) outputs. To have a flexible and reconfigurable algorithm, all the parameters (13,728 b) can be setup via the I^{2}C interface on-chip. The programming of the I^{2}C peripheral takes less than 50 μs corresponding to a total of 1,716 I^{2}C clock cycles, utilizing an 8 b input bus. Once the weights are setup, the algorithm adds a total latency of 2 bunch crossing (BX) cycles to the trigger path—one cycle to convert and another cycle to encode resulting in total inference latency of 50 ns and a new input accepted every 25 ns.

Integrated Converter, Encoder and I^{2}C peripheral: An integrated approach to the development is needed in order to avoid routing congestion of connecting the weights to the appropriate layers across the encoder. The floor-plan of the digital implementation occupying 2.4 mm$\times$1.5 mm is shown in Fig 6. The converter logic is located near the data input at the top of the design, majority of the area is occupied by the encoder, interleaved with the distributed I^{2}C network.

Design Considerations for Total Ionizing Dose Performance: Apart from all requirements considered above, our design must guarantee on-detector circuit reliability in the high radiation environment of HL-LHC [21, 22]. The circuitry should withstand total ionizing dose of approximately 200 Mrad over the lifetime of the experiment along with high SEE rates [23–25]. Since previous measurement results have indicated that the average time delay of all cells from the 65 nm LP process library increases after 200 Mrad irradiation [26], minimum size cells are avoided. Normal Vt standard cell technology library is used. The implementation uses concurrent multi-mode multi-corner static timing analysis for ensuring performance. The foundry worst case libraries are a good stand-in for modelling radiation damage. All weights are stored in registers and no SRAMs or DICE cells [27] are used.

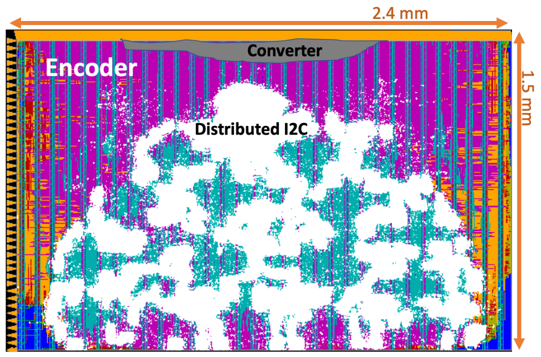


Fig. 6. Design floor-plan with an integrated converter, encoder and I^{2}C peripheral occupying a total area of 3.6 mm^2. The converter is highlighted in grey, the I^{2}C peripheral in white and the rest of the area is occupied by the encoder.

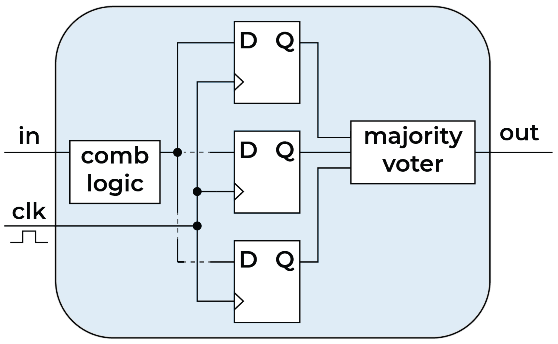


Fig. 7. Triple modular redundancy scheme used for the encoder and converter. Each register is triplicated and a majority voter determines the output.

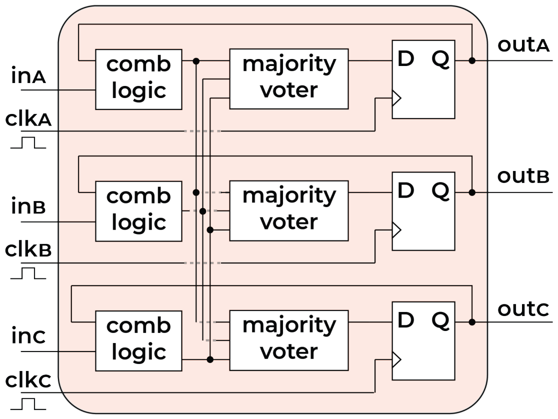


Fig. 8. Full module triplication is used for the I^{2}C peripheral. All combinational logic within the module is triplicated, which is used by three majority voters to form the inputs to triplicated registers. Feedback from the output of the registers enables autocorrection and protects against accumulating errors due to single event upsets over time.

TABLE II
DESIGN (D) AND VERIFICATION (V) METRICS FROM MODEL GENERATION TO VERIFIED IP.

STEP	TIME	ITER.	SIZE
Model generation (D)	0.98s	50-100	1089 C++ LoC
C Simulation (V)	0.14s		
High-level synthesis (D)	00:30:17	3-100	39,716 Verilog LoC
RTL Simulation (V)	00:00:46		
Logic synthesis (D)	06:04:19	6	769,481 gates
Gate-level simulation (V)	00:25:19		
Place and route (D)	39:33:11		806,255 gates
Post-layout simulation (V)	00:51:41		
Post-layout parasitic simulation (V)	01:51:30		
SEE simulation (V)	04:17:00		
Layout (D)	∼ 00:20:00	1	
LVS & DRC (V)	∼ 01:00:00		

Single Event Effect Mitigation: Mitigating single-event effects (SEEs) is a critical step in the ASIC implementation for effective performance in the HL-LHC environment. Several techniques have been proposed and used over the years to tackle this specific problem [28–30].

Triple modular redundancy (TMR) is a well-known technique to protect digital circuits against the undesirable effects of SEEs [31, 32]. Depending on the functionality of the block auto-correction features might be required for registers which store data.

We have used two different TMR implementations: simple TMR with triplicated registers and a majority voter for the data path shown in Fig. 7 and fully triplicating the entire module as shown in Fig. 8 for the I^{2}C peripheral for storing weights.

Since new data arrives to the encoder block every 25 ns, no auto-correction techniques are required. On the other hand, the values of the weights set by the I^{2}C peripheral (parameters of the NN) are vital as they are central to the vector multiplications used in NNs. Once programmed these are not expected to change over lengthy periods of time, hence, auto correction techniques are used to ensure that register errors due to single event upsets do not accumulate over time. As shown in Figure 8, all combinational logic within the module is triplicated, which is used by three majority voters to form the inputs to triplicated registers. Feedback from the output of the registers enables autocorrection. This method does require the I^{2}C peripheral to be clocked periodically.

Performance Metrics and Implementation Results

Table II lists metrics characterizing our design flow, such as the time spent for the design (D) and validation/verification (V) stages, the number of iterations, and the complexity of design representation at each stage. The table illustrates the main steps required to create a full and validated design. A co-design approach requires being able to have a rapid transition between each step to inform the other steps.

The HLS model description requires approximately 1,000 lines of code. This stage is fast (∼1 second) but requires several hundred iterations to optimize the algorithm performance, driven by the physics goals. The HLS stage determines the level of parallelism in the design, choice of pipelining, resource reuse factor, and clock frequency. This directly impacts the total area, power consumption, and the latency of the design. One hot encoding of finite state machines for robust Single Event Upset (SEU) prevention also needs to be introduced at this stage. Clock gating is employed to save system level power. The digital implementation stage is time intensive, requiring ∼65 hours of design and verification to meet the speed and area constraints with fewer iterations.

The final implementation results are presented in Table III. While there are challenges due to technology choices in making a comparison with a our design in an FPGA, we consider a fully unrolled FPGA implementation on a typical Xilinx Kintex Ultrascale FPGA device. Considering algorithm block power only and depending on configuration choices, an equivalent FPGA implementation consumes roughly 2.5–5 W with a latency of ∼300 ns [5] compared to the ASIC implementation, which consumes 95 mW. The ASIC implementation is expected to provide more than an order of magnitude improvement in power with a reduction in latency.

TABLE III
KEY SIMULATION PERFORMANCE PARAMETERS OF THE DESIGN.

Latency	Energy/inference	Power	Area
50 ns	2.38 nJ/inf.	95 mW	3.6 mm^2

V. SUMMARY AND OUTLOOK

A design methodology spanning from machine learning model generation to ASIC IP block creation has been presented. A low power, low latency reconfigurable data compression algorithm based on a convolutional neural network has been processed through synthesis and physical layout flows based on a 65 nm LP CMOS process, designed to withstand radiation environments of up to 200 Mrad.

For the ECON-T ASIC, our task is to perform efficient lossy compression of an HGCAL sensor energy pattern to transmit data to off-detector electronics. Compression is accomplished using a neural network autoencoder consisting of convolution and dense neural network layers. Optimal design and training of the algorithm is performed using quantization-aware training techniques to achieve good physics performance while optimizing for low power and area.

The encoder architecture set by the model requires approximately 225,000 multiply-accumulate operations to perform vector multiplications every 25 ns. In order to optimize for low power operation while maintaining data throughput of 40 MHz, a highly parallel architecture is chosen at the expense of larger area. The energy consumption per inference is 2.38 nJ. The final design consists of 800k gates and occupies a total area of 3.6 mm^2.

The design demonstrates how complex NN architectures can be implemented on the front-end ASICs with realistic area constraints, allowing for minimal loss of information in the trigger data stream. Furthermore, we show that in spite of a fixed ASIC implementation, ML algorithms can still be designed with sufficient flexibility to enable reconfiguration for new operational conditions. Apart from that, the $\mathrm{I}^2\mathrm{C}$ block allows real-time reconfiguration of weights, thus, facilitating the first steps toward real-time embedded AI. This is the first time that a radiation tolerant on-detector ASIC implementation of a neural network has been designed for particle physics applications.

We look forward to inclusion of the design IP in the ECON-T ASIC fabrication. This would allow us to test the design in a physical chip. Beyond the ECON-T ASIC, there is vast potential for future work using our design methodology including: other domain applications and adaptations for ultra low power, longer latency applications; other technology nodes and design considerations; and more types of neural network architectures, which could be scaled out to larger and more complex designs.

Authors' Contributions

The project was initiated and coordinated by FF, JH, and NT. CH, MK, and DN led the development of the algorithm training and HLS translation. GDG and CH performed HLS to RTL synthesis and validation of the RTL design. FF and YL focused on the digital implementation, power analysis, and algorithm interfaces to the rest of the chip including I2C with CG. FF, MBV, and LM developed techniques for making the design radiation tolerant. JH, SOM, and NT provided supervision for the project. JD, PH, VL, JN, MP, and SS provided support for the `hls4ml` infrastructure and initial algorithm implementations in HLS.

Acknowledgment

The authors would like to acknowledge CAD support from Sandeep Garg and Anoop Saha from Mentor Graphics for Catapult HLS and Bruce Cauble and Brent Carlson from Cadence for Innovus and Incisive. The authors would also like to thank the Fermilab ASIC group for incorporating the autoencoder block into the ECON-T ASIC; CMS HGCAL and Jean-Baptiste Sauvan for providing simulated module images for training; and Andre Davide for extensive input on network optimization.

We acknowledge the Fast Machine Learning collective as an open community of multi-domain experts and collaborators. This community was important for the development of this project.

References

[1] K. Albertsson *et al.*, "Machine Learning in High Energy Physics Community White Paper," *J. Phys. Conf. Ser.*, vol. 1085, p. 022008, 2018.

[2] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, "Machine learning at the energy and intensity frontiers of particle physics," *Nature*, vol. 560, p. 41, 2018.

[3] D. Bourilkov, "Machine and Deep Learning Applications in Particle Physics," *Int. J. Mod. Phys. A*, vol. 34, p. 1930019, 2020.

[4] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, p. 045002, 2019.

[5] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *JINST*, vol. 13, p. P07027, 2018.

[6] CMS Collaboration, "The Phase-2 upgrade of the CMS Level-1 trigger," CMS Technical Design Report CERN-LHCC-2020-004. CMS-TDR-021, 2020. [Online]. Available: https://cds.cern.ch/record/2714892

[7] ——, "The Phase-2 upgrade of the CMS endcap calorimeter," CMS Technical Design Report CERN-LHCC-2017-023. CMS-TDR-019, 2017. [Online]. Available: https://cds.cern.ch/record/2293646

[8] S. Chatrchyan *et al.*, "The CMS Experiment at the CERN LHC," *JINST*, vol. 3, p. S08004, 2008.

[9] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 4, p. 473, 2011.

[10] C. N. Coelho, A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, A. A. Pol, and S. Summers, "Automatic deep heterogeneous quantization of deep neural networks for ultra low-area, low-latency inference on the edge at particle colliders," 2020.

[11] A. Karbachevsky, C. Baskin, E. Zheltonozhskii, Y. Yermolin, F. Gabbay, A. M. Bronstein, and A. Mendelson, "Early-stage neural network hardware performance analysis," *Sustainability*, p. 717, 2021.

[12] CMS Team, "CMSSW on Github. http://cms-sw.github.io/."

[13] P. T. Komiske, E. M. Metodiev, and J. Thaler, "Metric space of collider events," *Phys. Rev. Lett.*, vol. 123, p. 041801, 2019.

[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 807.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *14th International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: JMLR, 4 2011, p. 315. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[16] Xilinx, "Vivado High-Level Synthesis," https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, 2020.

[17] Mentor/Siemens, "Catapult High-Level Synthesis," https://www.mentor.com/hls-lp/catapult-high-level-synthesis, 2020.

[18] Open Source Hardware Association, "A Resolution to Redefine SPI Signal Names," https://www.oshwa.org/a-resolution-to-redefine-spi-signal-names, 2020.

[19] Mentor/Siemens, "Catapult High-Level Synthesis - Verification," https://www.mentor.com/hls-lp/catapult-high-level-synthesis/hls-verification, 2020.

[20] Mentor/Siemens, "HLSLibs: Open-Source High-Level Synthesis IP Libraries," https://hlslibs.org, 2020.

[21] R. G. Alía, M. Brugger, F. Cerutti, S. Danzeca, A. Ferrari, S. Gilardoni, Y. Kadi, M. Kastriotou, A. Lechner, C. Martinella *et al.*, "LHC and HL-LHC: Present and future radiation environment in the high-luminosity collision points and RHA implications," *IEEE Trans. Nucl. Sci.*, vol. 65, p. 448, 2018.

[22] M. Huhtinen, "The radiation environment at the cms experiment at the lhc," Ph.D. dissertation, Helsinki U. Tech., 1996.

[23] R. D. Schrimpf and D. M. Fleetwood, *Radiation effects and soft errors in integrated circuits and electronic devices*. World Scientific, 2004, vol. 12.

[24] E. Petersen, *Single event effects in aerospace*. John Wiley & Sons, 2011.

[25] P. Dodd, M. Shaneyfelt, J. Schwank, and J. Felix, "Current and future challenges in radiation effects on cmos electronics," *IEEE Trans. Nucl. Sci.*, vol. 57, p. 1747, 2010.

[26] L. M. J. Casas, D. Ceresa, S. Kulis, S. Miryala, J. Christiansen, R. Francisco, and D. Gnani, "Characterization of radiation effects in 65 nm digital circuits with the DRAD digital radiation test chip,"

JINST, vol. 12, no. 02, p. C02039. 11 p, 2017. [Online]. Available: https://cds.cern.ch/record/2275135

[27] J. A. Maharrey, J. S. Kauppila, R. C. Harrington, P. Nsengiyumva, D. R. Ball, T. D. Haeffner, E. X. Zhang, B. L. Bhuva, W. T. Holman, and L. W. Massengill, "Dual-interlocked logic for single-event transient mitigation," *IEEE Trans. Nucl. Sci.*, vol. 65, no. 8, pp. 1872–1878, 2018.

[28] D. Fulkerson, "Single-event-effect hardened circuitry," Nov. 30 2006, uS Patent App. 11/136,920.

[29] S. Kuboyama, H. Shindou, Y. Iide, and A. Makihara, "Single-event-effect tolerant soi-based inverter, nand element, nor element, semiconductor memory device and data latch circuit," 2009, uS Patent 7,504,850.

[30] R. Gong, W. Chen, F. Liu, K. Dai, and Z. Wang, "A new approach to single event effect tolerance based on asynchronous circuit technique," *JETTA*, vol. 24, p. 57, 2008.

[31] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Dev*, vol. 6, p. 200, 1962.

[32] S. Habinc, "Functional triple modular redundancy (ftmr). vhdl design methodology for redundancy in combinatorial and sequential logic," *Gaisler Research, Design and Assessment Report (Version 0.2)*, 2002.