Ultra Fast Transformers on FPGAs for Particle Physics Experiments

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

This work introduces a highly efficient implementation of the transformer architecture on a Field-Programmable Gate Array (FPGA) by using the `hls4ml` tool. Given the demonstrated effectiveness of transformer models in addressing a wide range of problems, their application in experimental triggers within particle physics becomes a subject of significant interest. In this work, we have implemented critical components of a transformer model, such as multi-head attention and softmax layers. To evaluate the effectiveness of our implementation, we have focused on a particle physics jet flavor tagging problem, employing a public dataset. We recorded latency under 2 μs on the Xilinx UltraScale+ FPGA, which is compatible with hardware trigger requirements at the CERN Large Hadron Collider experiments.

1 Introduction

Accelerated Machine Learning (ML) inference is necessary to run the algorithms in the online event selection systems of the particle physics experiments. Due to the extremely high particle collision frequency of 40 MHz at the Large Hadron Collider (LHC) [1] at CERN [2], it is impossible to read out and store all the collision events. As a result, the LHC experiments [3, 4, 5, 6], try to read out only the interesting via an online selection system called the trigger. Most of the LHC experiments use a two-stage trigger system, hardware-based Level-1 trigger and software-based High-Level trigger. The Level-1 trigger operates at 40 MHz, so the algorithms usually run on application-specific integrated circuits (ASICs) or FPGAs. As the average number of collisions at the LHC is expected to increase with time, sophisticated ML algorithms will be crucial for Level-1 triggers to efficiently filter events. There have been numerous efforts to port ML algorithms like Deep Neural Networks [7], Convolution Neural Networks [8], Recursive Neural Networks [9, 10], Graph Neural Networks [11] onto FPGAs for physics applications using High-Level Synthesis (HLS) languages with the `hls4ml` package [7, 12]. `hls4ml` is an HLS-based compiler for a neural network to FPGA firmware conversion.

In recent years, the transformer [13] architecture became popular for their great performance in language modeling tasks like encoder-only BERT [14], decoder-only GPT [15], etc. Over time, the utility of transformer models extended beyond language modeling, impacting a wide range of ML applications. They are now widely used in particle physics for offline computing tasks like particle reconstruction [16], identification [17, 18, 19], etc. Often the transformer-based models show better performance over other architecture, but they are very computing intensive and suffer from a slow inference rate. Because of the computationally intensive nature, it becomes challenging to implement [20, 21, 22, 23] them on hardware like FPGAs, where a limited amount of resources is available. Another previous work [24] explored this design space in the context of a particle physics experiment by studying a small transformer for jet classification.

In this work, we present a flexible and efficient implementation of transformers written in HLS for the `hls4ml` package. This integration into `hls4ml` opens the door for wider low-latency applications of the Transformer models. Here the main focus is on the trigger applications in the LHC experiments. However, our implementation is very general and it is relevant to many real-time detector systems across fundamental science where low-latency high throughput inference is necessary.

2 Benchmark study

To benchmark our implementations, we study the open data samples from the Compact Muon Solenoid (CMS) experiment which contain top quark pairs decaying hadronically with center-of-mass energy of 7 TeV [25]. These events contain many bottom quark jets (b jets), charm quark jets (c jets) and jets from light quarks, and gluons (light jets) originating from top quark decay. The jets in the dataset are labeled as b, c, and light jets depending on whether they contain bottom quarks, charm quarks, or neither, respectively. The main feature that separates b jets (and c jets) from light jets is the presence of the displaced vertex corresponding to the decay of the hadron containing the b (or c) quark. These hadrons are long-lived due to their mass, and the decay time depends on their momenta. Our proposed algorithm aims to identify the presence of tracks that are consistent with these displaced vertices using a transformer architecture.

All the jets are reconstructed using the anti-kt algorithm with a distance parameter of $R = 0.5$. The jets are required to have transverse momenta (p_T) larger than 30 GeV and absolute pseudorapidities less than 2.0. Charged particle tracks with p_T larger than 1 GeV are associated with the nearest jet if they are within the angular distance ΔR (track, jet) of 0.5. Tracks within a jet are ordered by the significance of their transverse impact parameter ($\mathcal{S}(d_0)$), and only the first 15 tracks are used for this study. Each track is represented by a vector of six features: transverse and longitudinal impact parameters (d_0, d_z) and their significances ($\mathcal{S}(d_0)$, $\mathcal{S}(d_z)$), $\Delta R(\mathrm{track}, \mathrm{jet})$, and relative transverse momentum between the track and the jet ($p_\mathrm{T}(\mathrm{track})/p_\mathrm{T}(\mathrm{jet})$).

The flavor tagging classifier model is constructed using Keras+TensorFlow, using a transformer architecture with 9135 trainable parameters. The padded sequence of tracks, with a maximum length of 15, is directly fed into a transformer encoder block. No positional encoding is used, as the ordering is not crucial for this problem. Each encoder block contains a multi-head attention (MHA) layer with two heads, running two scaled dot-product attention layers in parallel, and a feed forward network with two dense layers. Outputs of the MHA layer are passed through a feed forward block where the layer dimensions are 8 and 6, respectively. Due to the simplicity of the flavour tagging problem, we did not include a layer normalization after the MHA layer. The structure of the encoder block is shown in Fig. 1a. The outputs of the encoder blocks are flattened and passed through three dense layers with 32, 16, and 8 units. The output layer uses a softmax function and predicts three class probabilities corresponding to b, c, and light jets. The model contains three encoder blocks and the architecture is shown in Fig. 1b. The training is performed with a categorical cross-entropy loss, with 30% of the training data retained for validation and testing.

3 Implementations

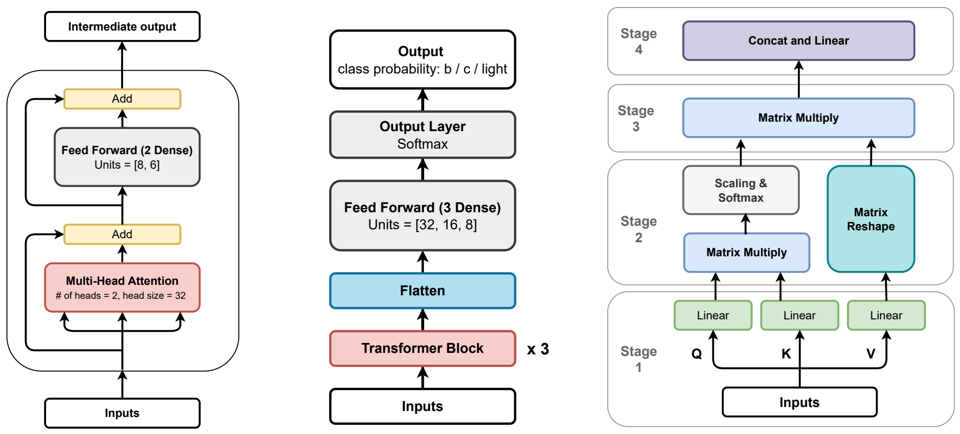
One of the main focuses of this work is to implement the MHA layer in HLS. The implementation of the MHA layer is divided into four sequential pipeline stages shown in Fig. 1c. Each stage is explained below.

The first stage is the Linear projection step where the inputs are transformed into Query (Q), Key (K), and Value (V) vectors using separate weight matrices. A matrix times a vector operation is performed at each time step as a pipeline. To optimize FPGA resources, the vectors from this stage are stored in a First In, First Out (FIFO) memory structure. This aligns perfectly with our sequential data processing, ensuring efficient memory utilization and fostering an effective data flow for the subsequent stages. Multiple FIFO memories are stacked together to increase the bandwidth.

The second stage starts computing the attention mechanism by taking the dot product of the Q and K vectors, producing a relevance score for each element of the input sequence. This score determines how elements influence each other in the sequence. The product is then divided by the dimension of the key vectors, $\sqrt{d_k}$, before passing it through a lookup table-based softmax function. The softmax output is stored in FIFO memory. Simultaneously, the matrix V is reshaped into a fully accessible array for later stages.

The third stage involves the matrix multiplication of the scores matrix and the corresponding V vectors. The V vectors are stored in a fully accessible register for the parallel multiplication process. The results are stored back in the FIFO memory and passed to the next stage of processing.

The fourth stage includes two key processes: the concatenation of the output from all attention heads and the subsequent linear transformation of the concatenated result. Each attention head provides an output vector loaded row by row, aligning with the temporal sequencing of the data. Once loaded, the outputs are concatenated together to form a single, unified data stream. Then the data stream is passed through a linear layer. The linear layer is also pipelined, and it inputs and outputs one row of data at a time. This stage manages the output from all heads and efficiently generates the final output.



(a) Transformer encoder block (b) Model architecture (c) Pipeline stages

Figure 1: The encoder block used for the transformer model is shown in (a). The full model architecture is shown in (b). The pipeline stages for the multi-head attention layer is shown in (c).

Apart from implementing the MHA layer, we have also optimized the softmax HLS implementation inside the `hls4ml` tool to reduce the computational cost. Softmax is used many times in the model, so it is crucial to have an efficient HLS implementation to run inference on an FPGA.

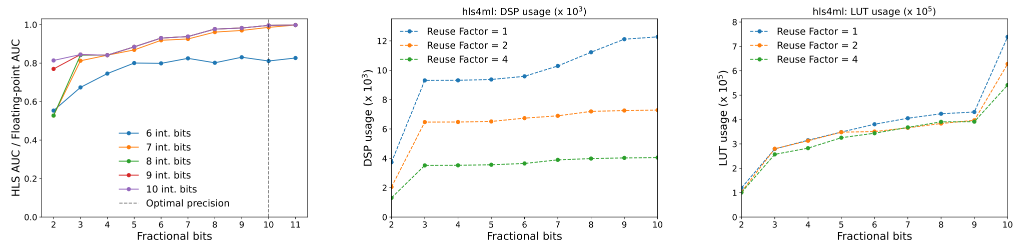
4 Results

The flavour tagging model described in Sec. 2 is translated into an HLS model using the `hls4ml` framework. Our tests were done using Vivado HLS 2020.1 with a Xilinx UltraScale+ FPGA VU13P (part number `xcvu13p-fhga2104-2L-e`) as the target device. For the HLS implementation two different optimizations are studied: quantization and parallelization.

The quantization process reduces the numerical precision of the model parameters, such as weights and biases, as well as inputs. Typically, ML model parameters are stored as 32-bit floating-point numbers Although floating-point numbers offer an extensive dynamic range, they consume significant computing resources when implemented on an FPGA. Therefore, for FPGA implementation, fixed-point numbers with fixed precision are preferred. This shift to fixed-point representation greatly accelerates computation by reducing both computational resource usage and memory utilization. In our study, we systematically explore fractional bit variations while maintaining a fixed precision of 6, 7, 8, 9, or 10 bits for the integer part. We evaluate the receiver operating characteristic (ROC) curve for the transformer-based classifier employing the area under the curve (AUC) as a performance metric. The ratio of the AUCs (fixed-point HLS model / floating-point Keras model) is shown in Fig. 2a as a function of fraction bits for integer bits. From the figure, it is clear that we need at least 10 integer bits and 10 fractional bits to get a similar performance as the floating-point model.

The `hls4ml` offers a valuable feature known as the “reuse factor" parameter, which plays a pivotal role in governing the optimization of parallelization and the efficient utilization of computing resources. This factor determines the number of times each multiplier is used for computing the neuron values within a given layer. If the reuse factor is set to 1, the computation becomes fully parallel, as each multiplication operation is executed independently by a dedicated digital signal processing (DSP) block. As we increase the reuse factor the computing resource utilization decreases, but the latency increases proportionally. To study the resource-latency trade-off and find an optimal implementation for our model, we have synthesized (full Vivado synthesis) it with varying values of the reuse factor and fractional bit precision. For each case, we quantified the utilization of FPGA resources of different categories like memory (BRAM), DSPs, flip-flops (FFs), and lookup tables (LUTs). The utilization of DSPs and LUTs are shown in Fig. 2b and Fig. 2c, respectively, as a function of fractional bits (integer bit = 10) for reuse factors of 1, 2, or 4. As anticipated, the resource utilization goes up as we reduce the reuse factor. It's worth noting that the target board has a total of 12288 DSPs and 1.72 million LUTs, providing us with flexibility in selecting any of the three reuse factors to achieve the optimal precision of (int. = 10, frac. = 10) during the model synthesis.

Remarkably, the observed latency aligns with the requirements of the LHC hardware trigger. For the fully parallel scenario with a reuse factor 1, the model's inference latency is 2.077 μs. Here, the clock period is 6.58 ns, resulting in output generation every 49 clock cycles or 322.42 ns. However, the latency increases to 3.467 μs and 5.853 μs for reuse factors of 2 and 4, respectively.



(a) AUC ratio (b) DSP usage (c) LUT usage

Figure 2: (a) Ratios of the fixed-point and floating-point AUCs as function of fractional bits. Five different values between 6 and 10 bits are chosen for the integer precision. Utilization of (b) DSP and (c) Lookup tables are shown as a function of fractional bits while keeping the integer part fixed to 10. Three different configurations with reuse factor of 1 (blue), 2 (orange), or 4 (green) are shown. The target board (part number `xcvu13p-fhga2104-2L-e`) has a total of 12288 DSPs and 1.72 million LUTs.

5 Summary and Outlook

We have successfully implemented a transformer architecture with multi-head attention in HLS for FPGA inference. This implementation has been seamlessly incorporated into the `hls4ml` package, which facilitates the automatic translation of transformer models for low-latency inference applications. It is essential to note that some critical features, including positional encoding and layer normalization, have been left for future work. To demonstrate the effectiveness of the current implementation, we conducted a study using a flavor tagging model. Notably, the model's inference latency falls within a range of 2 to 6 μs, fully complying with the stringent timing constraints of the hardware triggers. What sets our implementation apart is its exceptional versatility. It can readily adapt to models with different configurations, such as varying sequence lengths and the number of attention heads, without the need for extensive customization. As a result, this integration represents a pivotal development, and paves the way for the widespread utilization of low-latency applications employing transformer models.

6 Broader Impact

Although we demonstrate the performance of one specific algorithm here, this work could be used to accelerate other reconstruction algorithms in particle physics experiments. In fact, `hls4ml` transformer can be used for low latency inference for other scientific domains like neuroscience, gravitational wave, material science, etc., and various non-scientific domains.

References

[1] Lyndon Evans and Philip Bryant. LHC machine. *JINST*, 3(08):S08001–S08001, aug 2008.

[2] Cern accelerating science. (n.d.). `https://home.cern/science/accelerators/large-hadron-collider`, 2016.

[3] ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *JINST*, 3:S08003, 2008.

[4] ALICE Collaboration. The ALICE experiment at the CERN LHC. *JINST*, 3(08):S08002–S08002, aug 2008.

[5] CMS Collaboration. The CMS experiment at the CERN LHC. *JINST*, 3(08):S08004–S08004, aug 2008.

[6] LHCb Collaboration. The LHCb detector at the LHC. *JINST*, 3(08):S08005–S08005, aug 2008.

[7] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu. Fast inference of deep neural networks in fpgas for particle physics. *Journal of Instrumentation*, 13(07):P07027, jul 2018.

[8] Thea Aarrestad, Vladimir Loncar, Nicolò Ghielmetti, Maurizio Pierini, Sioni Summers, Jennifer Ngadiuba, Christoffer Petersson, Hampus Linander, Yutaro Iiyama, Giuseppe Di Guglielmo, Javier Duarte, Philip Harris, Dylan Rankin, Sergo Jindariani, Kevin Pedro, Nhan Tran, Mia Liu, Edward Kreinar, Zhenbin Wu, and Duc Hoang. Fast convolutional neural networks on fpgas with hls4ml. *Machine Learning: Science and Technology*, 2(4):045015, jul 2021.

[9] Elham E. Khoda et al. Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml. *Mach. Learn. Sci. Tech.*, 4(2):025004, 2023.

[10] Zhiqiang Que et al. Accelerating Recurrent Neural Networks for Gravitational Wave Experiments. In *32nd IEEE International Conference on Application-specific Systems, Architectures and Processors*, 6 2021.

[11] Abdelrahman Elabd et al. Graph Neural Networks for Charged Particle Tracking on FPGAs. *Front. Big Data*, 5:828666, 2022.

[12] Farah Fahim et al. hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices. In *tinyML Research Symposium 2021*, 3 2021.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[16] Alexander Shmakov, Michael James Fenton, Ta-Wei Ho, Shih-Chieh Hsu, Daniel Whiteson, and Pierre Baldi. SPANet: Generalized permutationless set assignment for particle physics using symmetry preserving attention. *SciPost Phys.*, 12(5):178, 2022.

[17] Huilin Qu, Congqiao Li, and Sitian Qian. Particle Transformer for Jet Tagging. 2 2022.

[18] V. Mikuni and F. Canelli. Abcnet: an attention-based method for particle tagging. *The European Physical Journal Plus*, 135(6):463, 2020.

[19] Vinicius Mikuni and Florencia Canelli. Point cloud transformers applied to collider physics. *Machine Learning: Science and Technology*, 2(3):035027, jul 2021.

[20] Bingbing Li, Santosh Pandey, Haowen Fang, Yanjun Lyv, Ji Li, Jieyang Chen, Mimi Xie, Lipeng Wan, Hang Liu, and Caiwen Ding. Ftrans: Energy-efficient acceleration of transformers using fpga, 2020.

[21] Hongwu Peng, Shaoyi Huang, Tong Geng, Ang Li, Weiwen Jiang, Hang Liu, Shusen Wang, and Caiwen Ding. Accelerating transformer-based deep learning models on fpgas using column balanced block pruning. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 142–148, 2021.

[22] Georgios Tzanos, Christoforos Kachris, and Dimitrios Soudris. Hardware acceleration of transformer networks using fpgas. In *2022 Panhellenic Conference on Electronics Telecommunications (PACET)*, pages 1–5, 2022.

[23] Seongmin Hong, Seungjae Moon, Junsoo Kim, Sungjae Lee, Minsub Kim, Dongsoo Lee, and Joo-Young Kim. Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation, 2022.

[24] Filip Wojcicki, Zhiqiang Que, Alexander D Tapper, and Wayne Luk. Accelerating transformer neural networks on fpgas for high energy physics experiments. In *2022 International Conference on Field-Programmable Technology (ICFPT)*, pages 1–8, 2022.

[25] Mc: Ttbar sample from the cms hep tutorial. `http://opendata.cern.ch/record/204`.