UNIVERSITY of WASHINGTON

Deploying ML into APU

Zhixing Jiang, Bowen Zuo, Maayan Tamarri, Liron Barak, Boping Chen, Jeff Eastlack, Scott Hauck, Shih-Chieh Hsu

Abstract

During the next update of the High-Luminosity Large Hadron Collider (HL-LHC) of ATLAS, a new global trigger subsystem will be installed into the L0 Trigger. New and improved hardware and algorithms will be deployed during the upgrade to increase the performance of the trigger system. The global trigger subsystem consists of various components, including the FPGA-based Global Event Processor (GEP), which processes the data through the trigger algorithm. Within the GEP, data will be pipelining through different Algorithm Processing Units (APU), which handle individual subtasks of the overall trigger.

We present our work in creating an APU specification and sample APU as a guide to future APU developers. We also present a redesign of the APU interface to follow the AXI-stream protocol, which allows streaming computations that overlap operations at multiple pipeline levels, potentially improving overall throughput. Finally, we present our work deploying HLS4ML (high-level synthesis for machine learning) into the APU development. HLS4ML is a design tool for generating a deep neural network (DNN) algorithm model with ultra-low delays, and has been developed specifically to support the needs of high-energy physics experiments. Our goal is to demonstrate that the application of HLS4ML to APU development is practical, and we have already implemented a convolution neural network (CNN) model for the APU. The performance of the CNN APU is tested using a test vehicle and a sandbox provided by the ATLAS developers. The next step is developing another new algorithm using a deep neural network.

Architecture of the APU

In the Global Event Processors (GEP), each trigger algorithm will be performed in an Algorithm Processing Unit (APU). At each Bunch Crossing, the detector would send the data to the GEP, and GEP would process those data using APU in pipelining.

> The APU receives data from a BRAM that stores the data from a MGT, or an upstream APU. After performing an algorithm on the data, the APU will send the data to a downstream BRAM for the next APU to read.

> APUs that get data from multiple sources will have a separate BRAM for each incoming data stream.

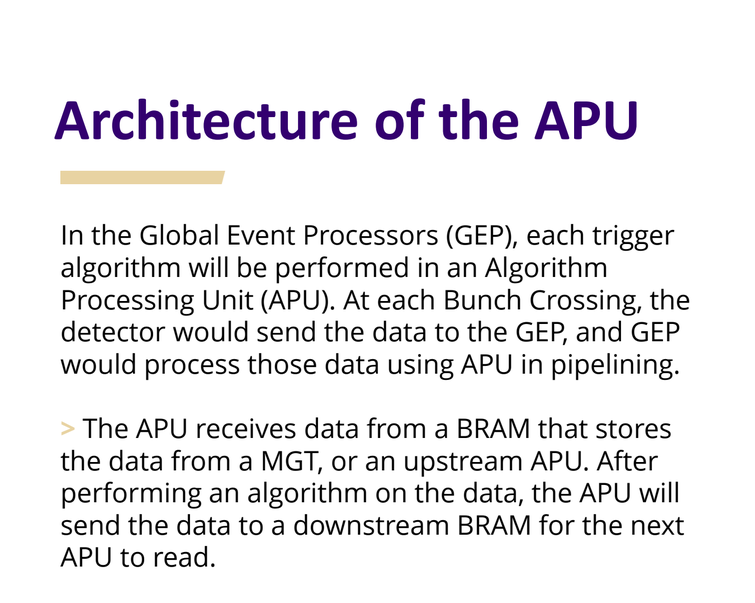

Fig1. The APU placement within the APP layer

HLS4ML

What is HLS4ML?

HLS4ML is a Python package for machine learning inference in FPGAs. We can use hls4ml to create firmware implementations of machine learning algorithms using high level synthesis language (HLS)

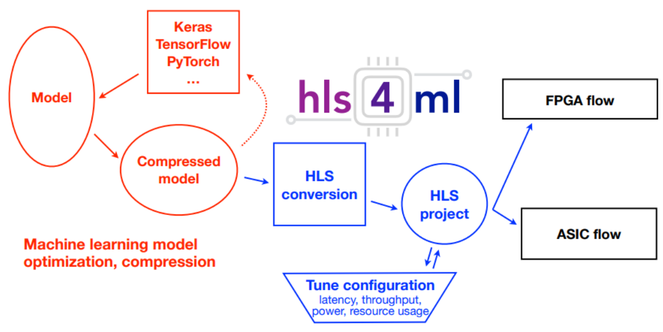

Fig2. The workflow of hls4ml

The benefits of using hls4ml for particle study:

> A convenient and efficient way to convert the machine learning model into the FPGA.

> Super low latency for faster computation.

> Supporting many different models, such as RNN, DNN, CNN…

Deployment of ML model

To prove the hls4ml generated ML model could function properly within the APU, our team used two arbitrary models for testing. One is a DNN (dense neural network) model, and the other one is a CNN (convolution neural network) model.

> The DNN model is a classification task of Jet tagging, which uses the input data to classify the type of each particle.

> The CNN model is an image classification task. This model is not particle related, since the prepose is to test the APU is able to adapt different model. Our team also develop an AXI-adaptor to converting the APP's i/o to follow the AXI-4 protocol to increase the speed of the APUs' communication.

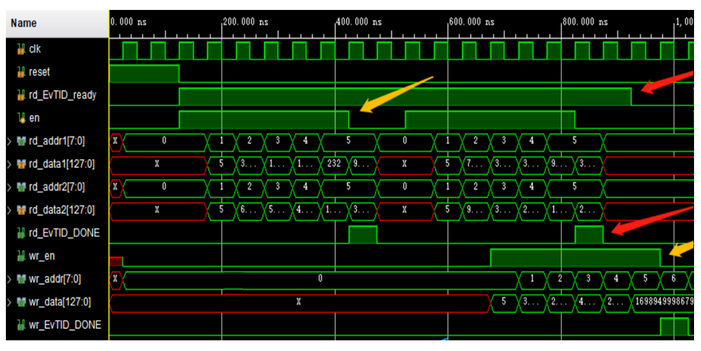

Fig3. The timing of the DNN model in the APU

From the above figure, the latency is very small, about 10 cycles, which indicates the hls4ml generated algorithm has an ultra low latency.

Resource	Utilization	Available	Utilization %
LUT	54333	1182240	4.60
LUTRAM	2	591840	0.01
FF	9557	2364480	0.40
BRAM	1.50	2160	0.07
DSP	2498	6840	36.52
IO	416	832	50.00
BUFG	1	1800	0.06

Fig3. The resource usage of the ML APU

Testing the ML APU

After deploy different model in the APU, we used a test vehicle and a sandbox provided by the ATLAS engineer for testing our design. The test vehicle has the following features:

1. It can simulate the pattern of real data flow.
2. It can support the communication of two APUs. The upper steam generated data will pass to the down stream.

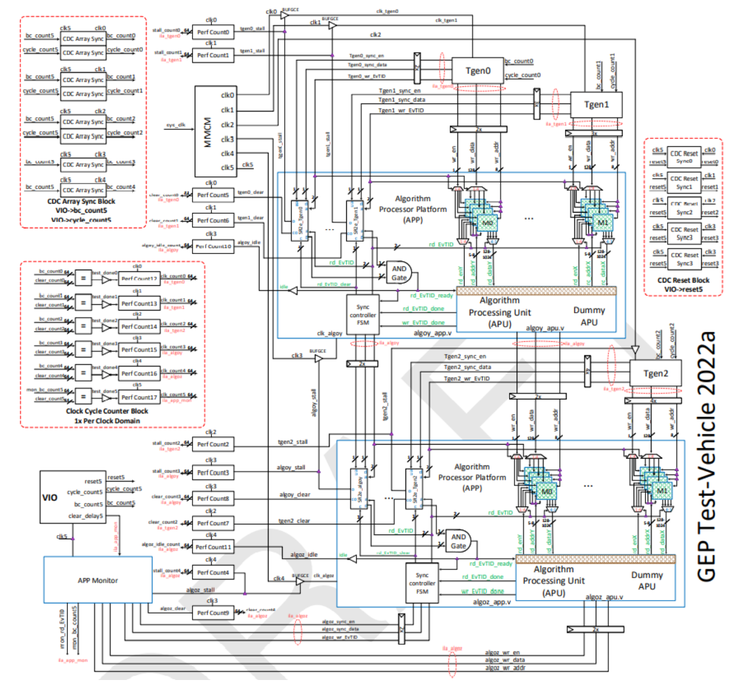

Fig4. The placement of the test vehicle for FPGA on board testing

Fig5. The testing result of on-board testing using test vehicle.

Conclusion

In this research, we proofed using the machine learning model generated by hls4ml to design the algorithm in APU would be practical and suitable. We proofed it by:

1. Discover the structure of the APU
2. Discover the structure of the hls4ml model
3. Deploy the model into APU
4. Testing the result of the ML APU.

After the testing, we deployed a real algorithm, Tau, into the APU. The latency and the accuracy of Tau met our expectation.

Reference

Elabd A, Razavimaleki V, Huang S-Y, Duarte J, Atkinson M, DeZoort G, Elmer P, Hauck S, Hu J-X, Hsu S-C, Lai B-C, Neubauer M, Ojalvo I, Thais S and Trahms M (2022) Graph Neural Networks for Charged Particle Tracking on FPGAs. Front. Big Data 5:828666. doi: 10.3389/fdata.2022.828666

Global Trigger Community (2021) ATLAS TDAQ Phase-II Upgrade: Firmware Specifications for the Global Trigger. ATL-COM-DAQ-2021-098