Contribution ID: **181** Type: **Oral**

Developments in hls4ml and its application

Wednesday, 21 September 2022 09:20 (20 minutes)

Neural Network (NN)-based inference deployed in FPGAs or ASICs is a powerful tool for real-time data processing and reduction. FPGAs or ASICs may be needed to meet difficult latency or power efficiency requirements in data acquisition or control systems. The software package, hls4ml, was designed to make deploying optimized NNs on FPGAs and ASICs accessible for domain applications. We will discuss recent improvements and applications, and discuss our recent collaboration with the AMD/Xilinx FINN group to develop a Quantized ONNX (QONNX) representation to serve as a common way to to represent quantized NNs.

Summary (500 words)

Neural Network (NN)-based inference is often deployed in FPGAs or ASICs when latency or power-efficiency are important design constraints. Examples include data acquisition and control systems. The software package, hls4ml, was designed to make deploying optimized NNs on FPGAs and ASICs accessible for domain applications. Though originally written for triggering at the CMS detector at CERN, the tool is general, with applications both in and out of high energy physics.

The package, hls4ml, takes machine learning input from standard tools like Keras or PyTorch and usually produces High Level Synthesis (HLS) code that can be synthesized to program FPGAs or design ASICs. The output is controlled by the selected backend, and is not restricted to HLS. The initial backend was for Vivado HLS, and now the Intel HLS compiler is supported in the main development branch. A special backend also targets accelerator boards, like Alveo or PYNQ, for direct deployment.

While creating new backends, it became evident that hls4ml had to evolve its internal representation (IR). More ideas from compiler design were incorporated. Features that were specific to particular backends were stripped from the IR, and the concept of optimizers and flows was emphasized. A flow is a sequence of steps to follow to transform the IR into the output. Some of the flows are general, while others are backend-specific. The output code fragments are generated by the backend-specific optimizers at the end.

There has been a focus to improve the networks that hls4ml supports. CNN support for larger networks was significantly improved. A direct implementation of a CNN results in deeply nested loops, quickly running into HLS synthesizability limits. Although it is possible to try to block the design, we found a streaming implementation to be superior. We are also currently integrating support for RNNs for both Vivado and Intel backends.

Recently we have collaborated with the AMD/Xilinx FINN group to develop a Quantized ONNX (QONNX) representation to serve as a common format for quantized NNs. Through tools we provide, both Brevitas and QKeras can produce QONNX, which can then be ingested by both FINN and hls4ml. QONNX is lightweight, introducing only three operators to implement uniform quantization, and abstract, purposely not being tied to the low-level representation of quantization in particular implementations. It will be proposed for inclusion into the ONNX standard.

Usage of hls4ml continues to grow. A demonstrator system is being built to control the Fermilab Booster magnet power suppy using reinforcement learning, and an accelerator Real-Time Edge AI for Distributed Systems (READS) project is making progress. Both of those projects use Intel FPGA SoCs, and we are collaborating with Crossfield Technology on configuring the distributed system. A computational storage system employing FPGAs is being studied for use by the DUNE experiment to detect supernova neutrino bursts. Together

with the FINN team we also produced a common submission to the MLPerf Tiny Inference Benchmark v0.7 open division. The results are given in the supplementary material.

Primary author: MITREVSKI, Jovan (Fermi National Accelerator Lab. (US))

Presenter: MITREVSKI, Jovan (Fermi National Accelerator Lab. (US))

Session Classification: Programmable Logic, Design Tools and Methods

Track Classification: Programmable Logic, Design Tools and Methods