TWEPP 2022 Topical Workshop on Electronics for Particle Physics

Contribution ID: **187**

Type: **Poster**

Exploring FPGA in-storage computing for Supernova Burst detection in LArTPCs

Thursday 22 September 2022 16:40 (20 minutes)

Latency and computational resources are key constraints for high bandwidth, low latency trigger systems. In these systems, even if a GPU/FPGA is used to accelerate computation, transferring data between components is still a costly operation for the host. In this work, we study a computational storage system employing FPGAs to detect supernova neutrino bursts, with a particular focus on LArTPC experiments. The computational storage system filters on interesting signals via machine learning wire reconstruction deployed on an FPGA and reduces data volume before storing in real-time. This workflow is applicable in other analysis/trigger applications including cosmology and particle physics experiments.

Summary (500 words)

Neutrino detectors, such as the Deep Underground Neutrino Experiment (DUNE) "far detector" are usually located deep underground in order to filter background noise. These detectors can be used to observe supernova neutrinos, and serve as a trigger to direct other observers to capture the supernova evolution early for multi-messenger astronomy. The neutrino detectors need to point the other observers to the supernova bursts.

Detector data is initially buffered underground. Providing the supernova location only after transferring all that data to the surface for processing would delay the message too long for others to capture the evolution. Therefore, at least some processing needs to be done in the cavern, either to fully point to a supernova, or to select a small subset of data to send to the surface for processing.

In order to not burden the processor, we want to exploit "in-storage computation." In particular, we seek to use an accelerator that accesses the data directly from storage for the processing. For our demonstrator, we are using a Xilinx Alveo accelerator, accessing SSD storage using PCIe peer-to-peer transfers.

One of the primary tasks that the computational storage system is performing is running a machine learning algorithm to identify regions of interest within LArTPC waveforms. This model was adapted and retrained on simulated DUNE LArTPC data, and further optimized by hand, along using an automated hyperparamter tuning platform determined.ai using the ASHA algorithm. The model is small, taking an input of 200 points of 1D waveform data, and consisting of three 1D convolutional layers with one dense output layer. In total, the model has approximately 21,000 parameters. After training and optimization, it is then converted into FPGA firmware via the hls4ml software package.

The hls4ml software package was designed to make deploying optimized NNs on FPGAs and ASICs accessible for domain applications. hls4ml takes ML input from standard tools like Keras or PyTorch and usually produces High Level Synthesis (HLS) code that can be synthesized by for example, Vivado HLS. It was originally written to help the design of the first level triggering system for the CMS detector at CERN. The hls4ml generated HLS is combined with a data parser and run as a kernel in the Vitis accelerator methodology.

The hls4ml package provides tunable parameters for various tradeoffs between size and latency. We can also instantiate multiple kernels. We are exploring other processing to also do in the accelerator to best achieve our goal of providing pointing information quickly.

Authors: HAWKS, Benjamin (Fermi National Accelerator Lab); SHEN, Jieran; MITREVSKI, Jovan (Fermi National Accelerator Lab. (US)); SCHOLBERG, Kate; WANG, Michael; TRAN, Nhan (Fermi National Accelerator Lab. (US)); DING, Pengfei (Fermi National Accelerator Laboratory); CAI, Tejin; YANG, Tingjun (Fermi National Accelerator Lab. (US)); JUNK, Tom (Fermi National Accelerator Lab. (US))

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

Session Classification: Thursday posters session

Track Classification: Programmable Logic, Design Tools and Methods