EE 511: Machine Learning Hardware Accelerators
Final Project

University of Southern California

Fall 2025

Due: Decemeber 17

Project: SqueezeNet v1.0 Implementation on Kria KV260 using HLS

Objective. In this project, you will implement the *original* SqueezeNet v1.0 convolutional neural network (as described in the 2016 paper by Iandola et al.) on the Xilinx Kria KV260 Vision AI Starter Kit. The implementation must be done using High-Level Synthesis (HLS), and must **not** include any residual or bypass connections (i.e., no skip connections between Fire modules).

1. Background

SqueezeNet is a compact convolutional neural network designed to achieve AlexNet-level accuracy on ImageNet with dramatically fewer parameters. Its fundamental building block is the *Fire module*, consisting of:

- A **squeeze** layer: 1×1 convolutions that reduce the number of channels.
- An **expand** layer: a combination of 1×1 and 3×3 convolutions whose outputs are concatenated along the channel dimension.

The full architecture (without any bypass connections) is described in **the original SqueezeNet v1.0 paper**.

2. Tasks

Below are the required components you must complete in order to finish the project.

Task 1: Training SqueezeNet
Train a full-precision SqueezeNet model on the CIFAR-10 dataset to convergence. Report the final test accuracy and include the complete training curve (loss and accuracy over epochs). Provide the full hyperparameter configuration, including optimizer, learning-rate schedule, batch size, number of epochs, weight decay, and any other relevant settings.

Task 2: SqueezeNet Quantization

Starting from the converged checkpoint from Task 1, perform Quantization-Aware Training (QAT) to quantize both weights and activations to 8-bit fixed-point. The allocation of bits between integer and fractional parts is up to you, but you must report the exact configuration used. After QAT, report the final test accuracy of the quantized model.

Task 3: HLS Modeling of SqueezeNet

- Using Vitis HLS, implement reusable and flexible modules for:
  - 1×1 and 3×3 convolution,
  - max-pooling,
  - the full Fire module (squeeze + expand + concatenation).
- Design a controller that is responsible for:
  - Passing the correct arguments and memory addresses to each function,
  - Maintaining the correct order of computations by enabling and disabling the appropriate modules.
- Apply appropriate HLS pragmas (loop unrolling, pipelining, array partitioning, etc.) to explore the trade-off between latency and resource utilization.
- Verify functional correctness of your HLS model using C/C++ testbenches.

Task 4: Running the Kernel on the Kria KV260

- Follow the proper workflow to generate all required files for deploying your kernel onto the board.
- Write a correct PYNQ-style host program that:
  - Loads and preprocesses an input image,
  - Invokes the hardware kernel,
  - Measures kernel latency,
  - Verifies correctness of the classification results.

3. Constraints and Notes

- The dataset that you are going to train on is the CIFAR-10 dataset, but the SqueezeNet is designed for the ImageNet dataset. Hence, you must upsample your input image to match the input dimensions of the model.
- You must implement the **original** SqueezeNet v1.0 architecture **without** any bypass or residual connections.
- Instead of calling functions sequentially inside Vitis HLS, you must introduce an enable signal for each module and design a dedicated controller function responsible for activating/deactivating these enable signals and passing the correct arguments and read/write addresses.

4. What to Submit

1. A written report (in IEEE format) including:
    - Introduction,
    - Related Work,
    - Model quantization and training (Techniques used for the training and achieved quantization precision),
    - High-Level Implantation (Architecture specification and diagrams, Description of the HLS design and optimization strategies),
    - Architecture specification and diagrams,
    - Description of the HLS design and optimization strategies,
    - Resource utilization and performance results,
    - Results (final accuracy on the CIFAR-10 dataset for both tasks 1 and 2, resource utilization and performance results, and screenshots of the verified results on the FPGA board),
    - Conclusion.
2. All HLS project files and source code.
3. The generated `.hwh`, `.tcl`, and `.bit` files, along with the PYNQ host code.

Contact

For any questions regarding this project, please email: sadeghim@usc.edu.