209 AS TA Office Hour 2

Shurui Li

Project 2 part 1

- Verilog + Genus exercise
- Goal is to understand the impact of output bitwidth (and truncation methods) on energy and area of the dot product unit
- Shouldn’t be too complicated to implement

Notes on part 1

- Make sure synthesis is completed without any error. Warnings are generally fine.
- Make sure you get positive slack in your timing report.
- The output bitwidth requirement is for the final dot product output.

Project 2 Part 2 Overview

- Quantization-aware training exercise with a special way to quantize the weights
- Same weights are used for both 8 and 4 bit quantization
- This setup is useful for hardware that support dynamic precision adjustment
  - Bit-serial processing
  - Analog NN accelerator such as Compute-in-memory (CIM)

Why QAT

- Direct post-training quantization often results in high accuracy drop if the bitwidth is low
- Need to retrain the model with quantization to let the model (weights) aware of the quantization behavior -> Quantization-aware training (QAT)
- Neural network use back propagation (chain rule) and gradient descent to update the weights -> Is quantization differentiable?
- No. Then how to calculate the gradient?

Straight through estimator (STE)

- Purpose: Enables gradient-based training with non-differentiable functions.
- How It Works:
  - Forward Pass: Applies the non-differentiable function (e.g., quantization).
  - Backward Pass: Approximates gradient as if the function were the identity.
- Advantages:
  - Simple to implement.
  - Allows training with discrete variables.
- Sample implementation is provided

Notes on quantization

- For fixed-point quantization, you need to determine the quantization range
- There is a trade-off between range and precision
  - Range too small -> outliers will be clipped, but often those large values are quite important
  - Range too large -> poor precision, smaller values have high % quantization error, and most values are those small values
- Multiple ways to determine the optimal range
  - Offline profiling -> check the distribution of pre-trained weights and try a few options
  - On-the-fly profiling -> adjust the range through some heuristics on-the-fly during training
  - Trainable range -> make the quantization range a trainable parameter and trains for it. Maybe works, but may not as good as you'd expected

Additional notes

- You need to submit your retrained weights in FP format
- The code you submit should take your weights, perform inference, and report the accuracy for 8-bit and 4-bit cases.