FPGA-Based LeNet-5 CNN Accelerator for Handwritten Digit Recognition

Team member: Yizheng Tang, yt2992; Chenxi Shen, cs4634; Tian Li, tl3468; Weiwei Wu, ww2766; Sirui Chen, sc5746

1. Project Motivation

Convolutional Neural Networks (CNNs) are widely used in computer vision applications. However, CNN inference is computationally intensive, making hardware acceleration essential to improve performance and energy efficiency.

In our practical system, neural networks are trained in a software environment such as C, while inference is accelerated using specialized hardware.

In this project, we propose a software–hardware co-design framework based on the LeNet-5 convolutional neural network. The model will be trained in a software environment using the MNIST handwritten digit dataset, where convolution kernels and weights are obtained through training. These trained parameters will then be exported and deployed on an FPGA platform. The FPGA will implement the inference engine, including convolution, pooling, and fully connected operations, enabling hardware-accelerated digit classification.

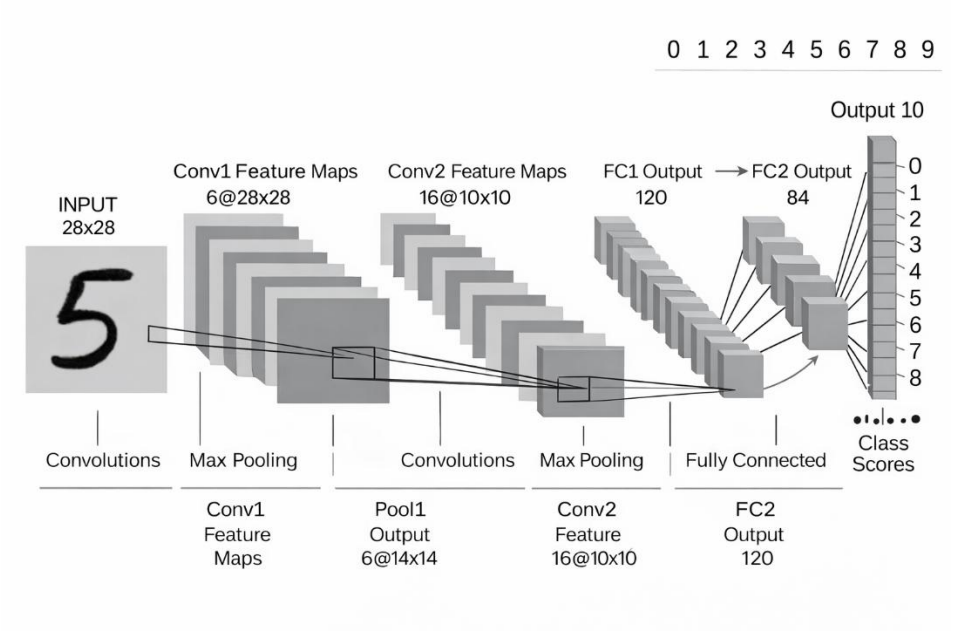

Fig 1. CNN Model Architecture

2. Project Objective

The objective of this project is to design and implement a software–hardware co-design framework for the LeNet-5 convolutional neural network on an FPGA platform. In the

software stage, the network will be trained using the MNIST handwritten digit dataset to obtain the convolution kernels and network weights. In the hardware stage, these trained parameters will be deployed to an FPGA-based inference engine that performs convolution, pooling, and fully connected operations for digit classification. The project also aims to evaluate the system in terms of classification accuracy, inference latency, and FPGA resource utilization.

3. CNN Architecture

This project uses the LeNet-5 architecture, one of the earliest and most influential convolutional neural networks designed for handwritten digit recognition. The block diagram is as follows:

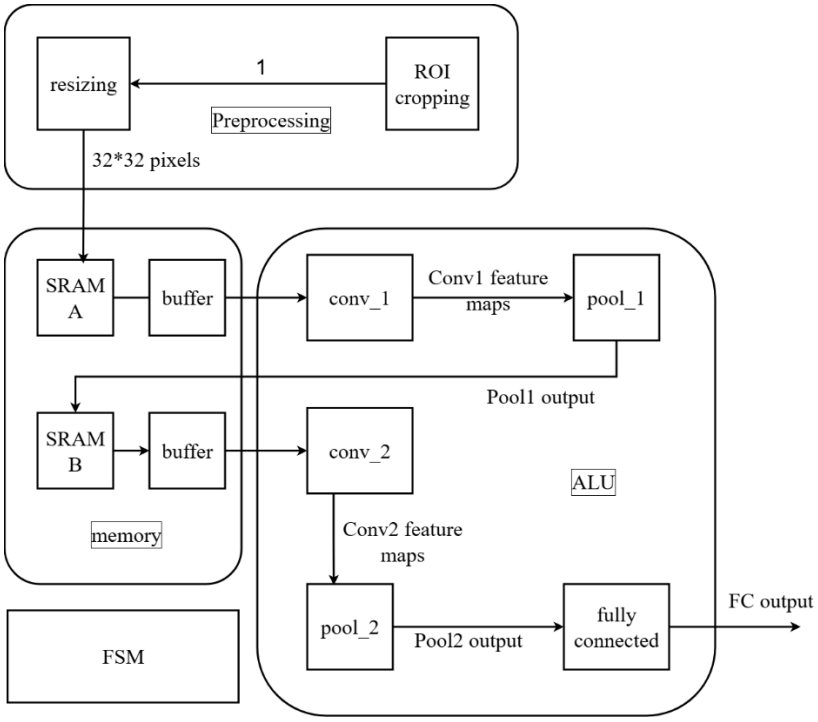

Fig 2. Block Diagram

The FPGA accelerator will implement the inference stage of the CNN. The hardware architecture will include the following modules:

- **Input Buffer**: Stores the input image data.
- **Convolution Engine**: Performs convolution operations using multiply-accumulate (MAC) units or systolic array.
- **Pooling Unit**: Implements average pooling to reduce feature map size.
- **Fully Connected Unit**: Computes the final classification outputs.
- **Control Unit**: Coordinates data flow and layer execution.

The simplified hardware architecture is shown below

4. Software Training

The CNN model will be trained in a software environment using the MNIST handwritten digit dataset. The training process will produce the convolution kernels and network weights required for inference.
The steps include:

- Implement LeNet-5 in software.
- Train the model using the MNIST dataset.
- Export trained weights and biases.
- Convert the parameters into a format compatible with FPGA memory.

These parameters will then be loaded into the FPGA hardware accelerator for inference.

5. Hardware requirement

A camera module will capture handwritten digit images, and a push button on the FPGA board will trigger image acquisition and CNN inference. The captured image will be preprocessed by ROI cropping and resizing before being processed by the CNN accelerator. The predicted digit will be transmitted to a host computer via a serial interface and displayed on the terminal.

6. Evaluation Metrics

The performance of the proposed system will be evaluated using the following metrics:

- Classification Accuracy on the MNIST dataset
- Inference Latency
- Throughput
- FPGA Resource Utilization: LUT, DSP and BRAM

These metrics will help evaluate the effectiveness of the FPGA-based CNN accelerator.

7. Expected Outcome

This project aims to demonstrate a complete software–hardware co-design workflow for CNN inference acceleration. By training the model in software and deploying inference on FPGA hardware, the system is expected to achieve efficient digit classification while utilizing moderate FPGA resources.