Term Project Report: CNN accelerator

ZHOU Tiancheng 2025-81701
Tadiwos Yehualashet KEBEDE 2023-13884

Date of Submission: 20/12/2025 G.C.

I. Introduction

In this term project, we were tasked with implementing Convolutional Neural Network (CNN) compute primitives as hardware modules using the Verilog Hardware Description Language (HDL). Our goal was to accelerate CNN inference for images on an FPGA Arty-7 A100 model.

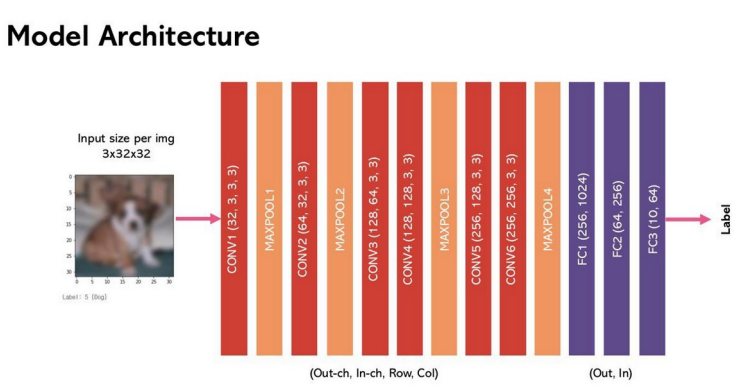
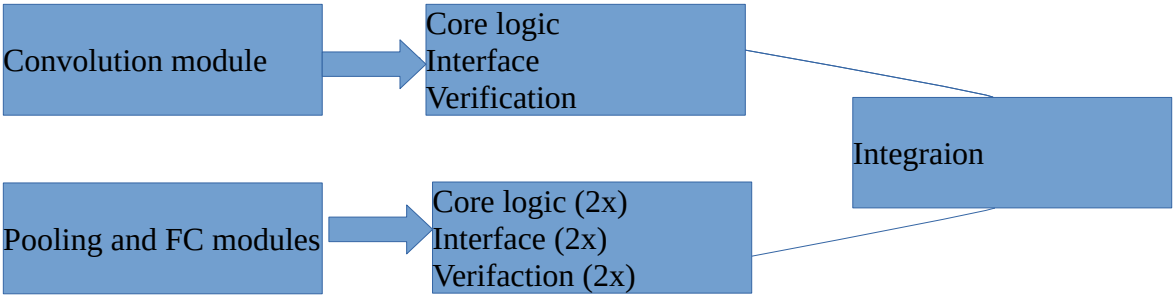


Figure 1: CNN architecture

The CNN architecture used for this project was a modified VGGNet consisting of 6 convolution layers, 4 pooling layers, and 3 Fully Connected (FC) layers, as shown in Figure 1. Using a provided pretrained model and the CIFAR-10 dataset, our primary objective was to perform an end-to-end inference process from the host CPU to the FPGA board.

II. Methodology

Our project was completed in a group of two. To ensure a fair and efficient workload, we adopted a parallel workflow. Our plan was to implement and verify each module separately before integrating them into the final design.



Drawing 1: Overall project flow

- **Work Distribution**: Tiancheng (2025-81701) focused on the Pooling and FC modules, while Tadiwos (2023-13884) developed the Convolution module .

- **Integration Plan**: After validating each module through simulation, we planned to fuse them together for board-level verification.
- **Tools**: We used Xilinx Vivado for all Verilog coding and simulations.

1. **Convolution Module Design**

The design for the convolution layers had to be parameterized because the six layers in the architecture have varying input and output channel dimensions. We designed this module using three Finite State Machines (FSMs):

FSM1 (Data Loading): This FSM handles the uploading of image data, weights, and bias parameters into dedicated SRAMs (implemented via BRAMs).

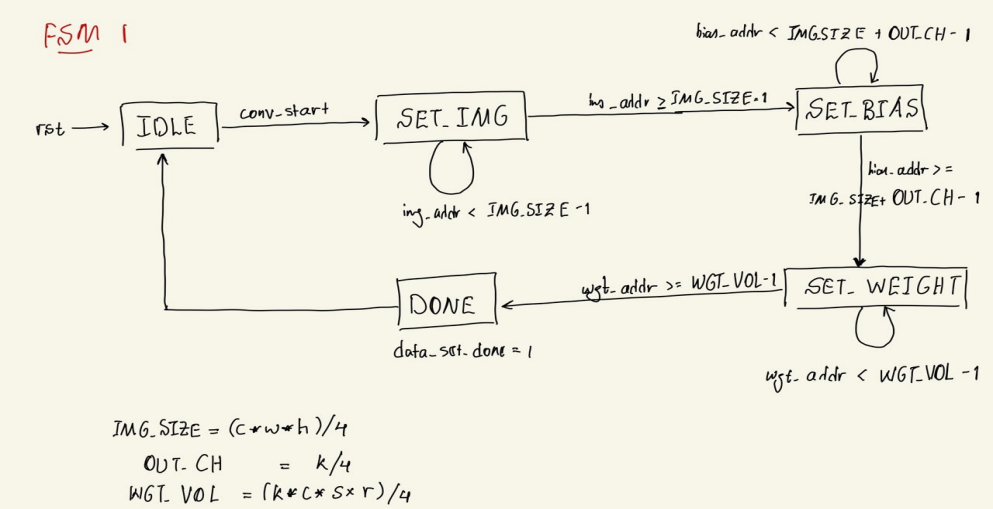


Figure 2: convolution module FSM1

FSM2 (Computation): To speed up high-dimensional convolution loops, we utilized an IM2COL (Image to Column) transformation to convert the operation into matrix multiplication. We then implemented a Weight Stationary Systolic Array (WS SA) to execute these calculations efficiently. This specific architecture was chosen to reduce memory reads by reusing weight parameters.

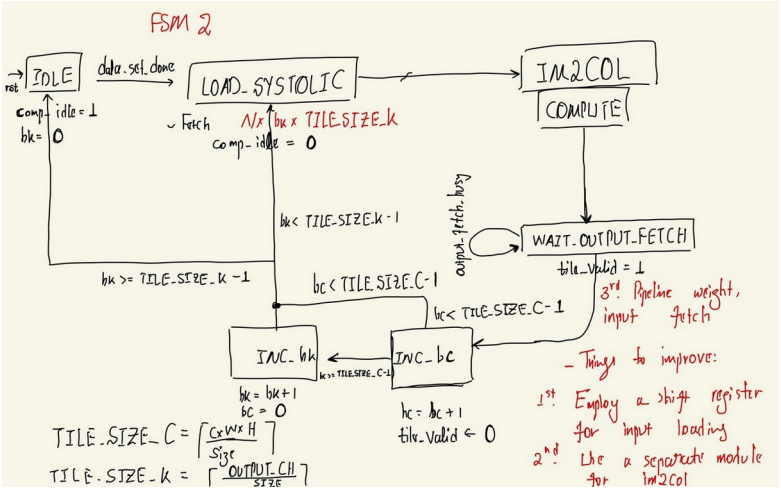


Figure 3: convolution module FSM2

FSM3 (Output & Activation): This FSM fetches the results from the systolic array, performs Quantization (8-bit), add bias to the outputs and lastly perform ReLU activation (clipping negative values to zero) before sending the final data back to the host.

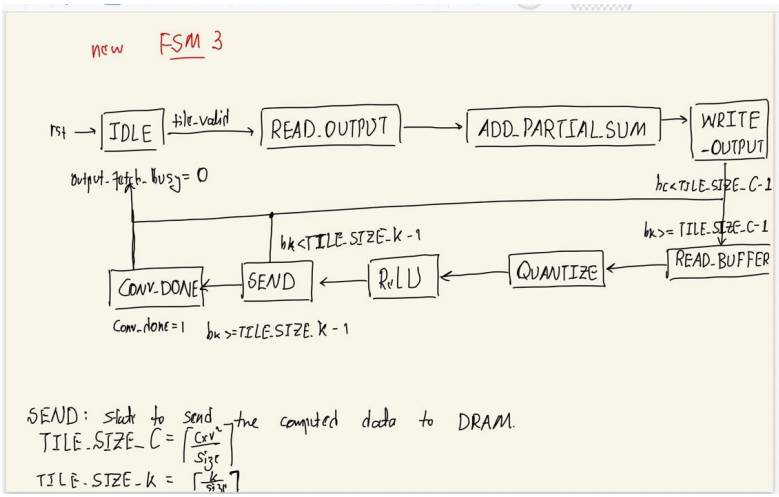


Figure 4: convolution module FSM3

2. **FC Layer Module Design**

The FC module implements the core dot-product operation: Result = sum(Input*Weight) + Bias. It features internal registers for resource management and pipelining. Key features include:

- **ReLU Activation**: Integrated internally to clip negative values.
- **Argmax Tracking**: For the final layer, the module tracks the highest probability index to determine the correct CIFAR-10 class.

3. **The Maximum Pooling Layer Module**

This module performs downsampling with a **window size of 2 and a stride of 2**, effectively halving the spatial dimensions of the feature maps to reduce computation. It operates by comparing data across rows and columns to find the maximum value within each window .

IV. Results and Findings

While we did the planning and code writing for the FC and Max Pooling modules, along with a significant portion of the Convolution module, we encountered challenges during the testing phase. Unfortunately, our modules did not pass the required test cases, which prevented us from moving on to board implementation and optimization.

We've identified several factors that contributed to this outcome:

- **Time Constraints**: Balancing this project with other classes proved difficult.
- **Communication & Planning**: We underestimated the complexity of the project and could have benefited from stronger communication and more frequent coordination.

Despite these hurdles, the project was a good learning experience. We gained practical knowledge about hardware acceleration, specifically the inner workings of Systolic Arrays and the IM2COL transformation for matrix multiplication .

V. conclusions and Recommendations

Our goal was to create dedicated hardware components to accelerate CNN inference through a multilayered architecture. Although we didn't reach the final board implementation, we now have a much deeper understanding of the design complexities involved. For future projects, we recommend:

- **Early & Constant Coordination**: Teams should dedicate significant time to collaborative debugging early in the process.
- **Proactive Support**: It would be very helpful if Teaching Assistants could provide more practical guidance for teams who are lagging behind in the early stages, helping to identify and solve roadblocks before the final deadline.