JANUARY 15, 2025

IC DESIGN CAPSTONE PROJECT HARDWARE ACCELERATOR FOR CONVOLUTION IN IMAGE PROCESSING

MUHAMMAD MISBAH
DREAMBIG SEMICONDUCTOR
Jr. Hardware Engineer

Table of Contents

1. Introduction

Convolution Neural Networks (CNNs) are widely used in AI tasks such as character recognition, robot vision, and image and speech recognition due to their remarkable accuracy. Convolution calculations, which account for approximately 90% of the total processing workload, are the most computationally intensive component of CNNs [1]. To address this challenge, numerous new accelerator architectures have been designed to reduce CNN processing time.

In this project, we designed and evaluated a hardware accelerator architecture specifically tailored for processing **32 × 32** images. The next section provides a detailed discussion of the methodology employed in designing and implementing the proposed hardware accelerator architecture.

The complete project code is available on GitHub at https://github.com/misbahghufran99/hw-acc-for-convolution-in-image-processing.git

2. Methodology

The high-level architecture of the proposed design consists of five primary components: an **Input Memory**, four computational stages (**Stage 1 to Stage 3**), an **Output Memory**, and a **Control Unit** that orchestrates the overall data flow and process synchronization.

The Input Memory stores the pixel data, which is sequentially fetched by Stage 1 for pixel loading. This data is then processed in Stage 2, where it undergoes multiplication with the specified kernel pixels, enabling convolution operations. In Stage 3, the multiplied values are summed through addition, preparing the data for storage.

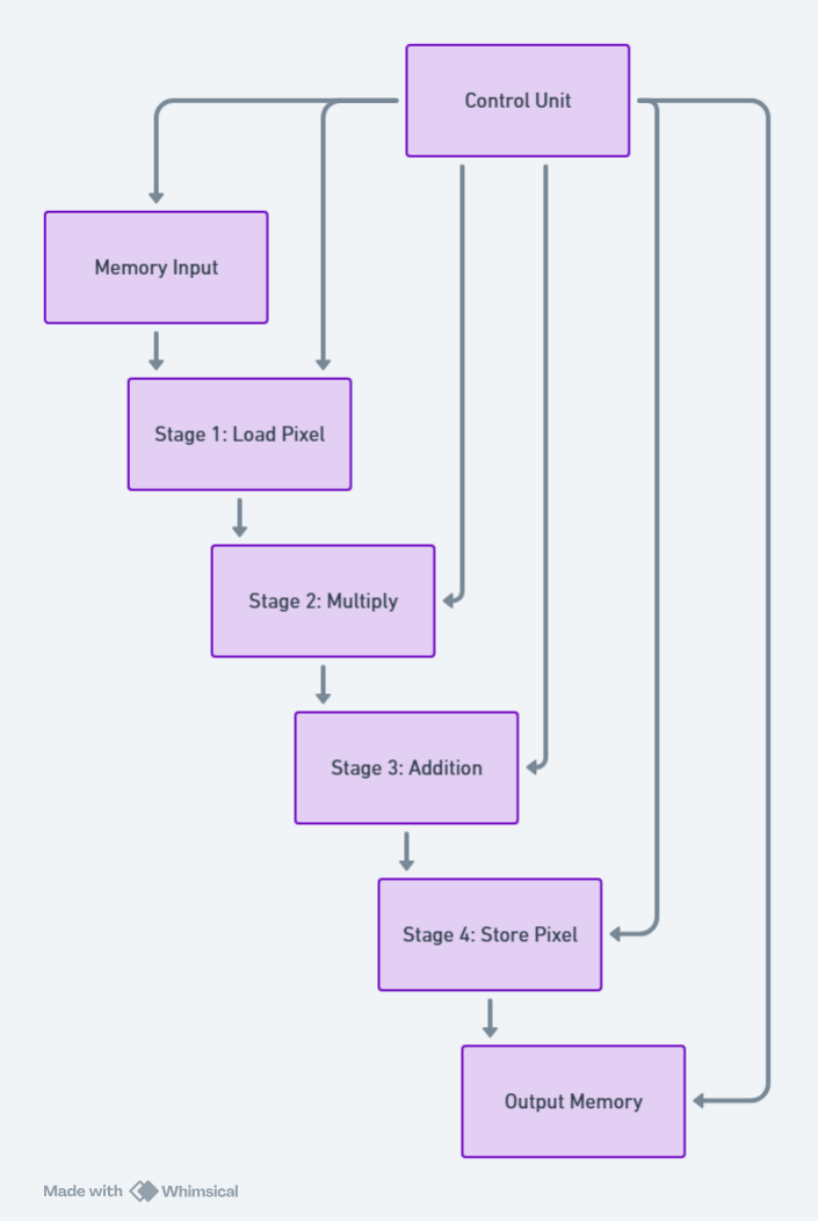

Figure 1:High Level Architecture

The control stage operates using two primary input signals: **start** and **reset (rst)**. When the start signal is asserted, the control unit initiates data transfer by generating **wren** (write enable) and **addr** (address) signals, which are transmitted to the input memory. These signals facilitate the retrieval of pixel data from the designated memory locations. The retrieved pixel data is then directed to a **demultiplexer**, which determines the appropriate destination register based on the **demux_sel** signal. Initially, the demultiplexer routes the first pixel to Pipeline Register 1. Subsequently, as new pixel data is fetched, the demultiplexer selects the next register in the sequence, assigning Pipeline Register 2 for the second pixel, and continues this process until all nine pipeline registers are populated. This controlled data distribution ensures that each pixel is correctly staged for further processing in subsequent phases, maintaining precise timing and synchronization within the pipeline.

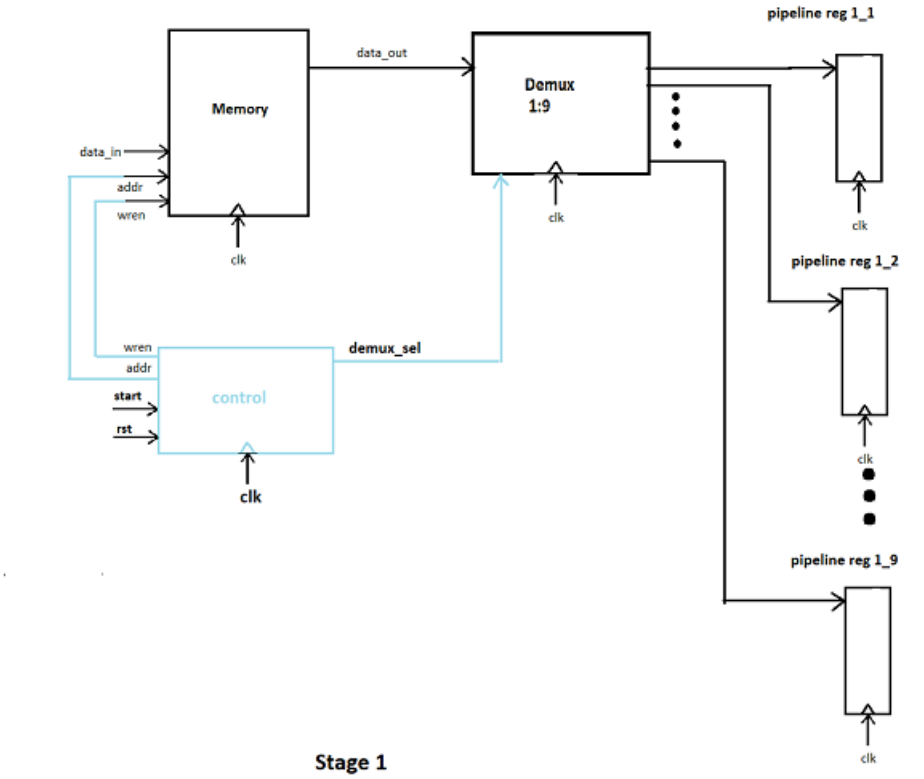

Figure 2: Stage1

In Stage 2, the architecture is designed to perform element-wise **multiplication** using a set of nine parallel **multipliers**. Each multiplier receives two inputs: one from the corresponding Pipeline Register of Stage 1 and a fixed kernel pixel value. The multipliers simultaneously process their respective inputs, performing a multiplication operation for each pixel from Stage 1 with its associated kernel value. The resultant values of these multiplications are then stored in the corresponding Pipeline Registers of Stage 2. This parallel structure, with dedicated multipliers for each of the nine input pixels, ensures efficient and concurrent computation, minimizing processing latency and maintaining the integrity of the data pipeline for further operations in subsequent stages.

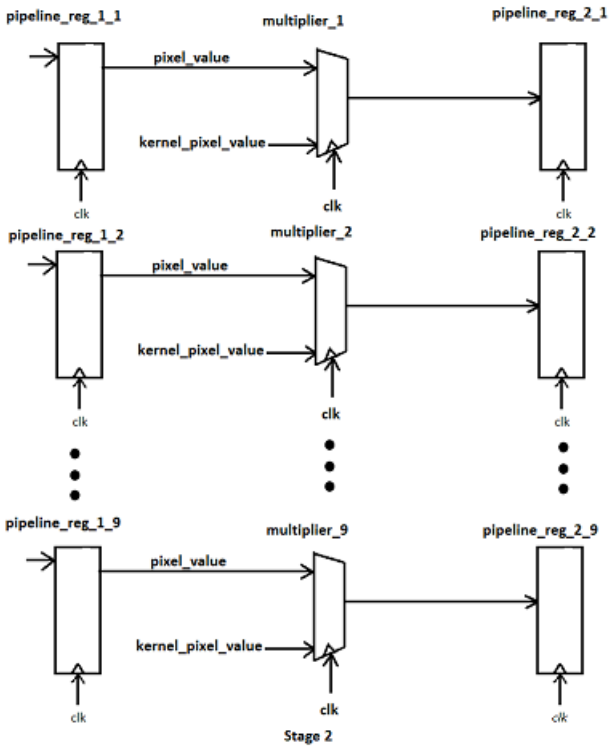

Figure 3: Stage 2

In Stage 3, the architecture focuses on performing **addition** operations to aggregate the results of the previous stage. A **9:1 multiplexer** is utilized to manage data flow, with its nine inputs connected to the outputs of the Pipeline Registers from Stage 2. The **control unit** generates a **mux_sel** control signal to sequentially select inputs for processing. Each selected input is routed through the multiplexer to a single **adder,** which cumulatively sums the values. This addition operation continues until all nine values have been processed, resulting in a single summed output. This aggregated value is then can be stored in output memory.

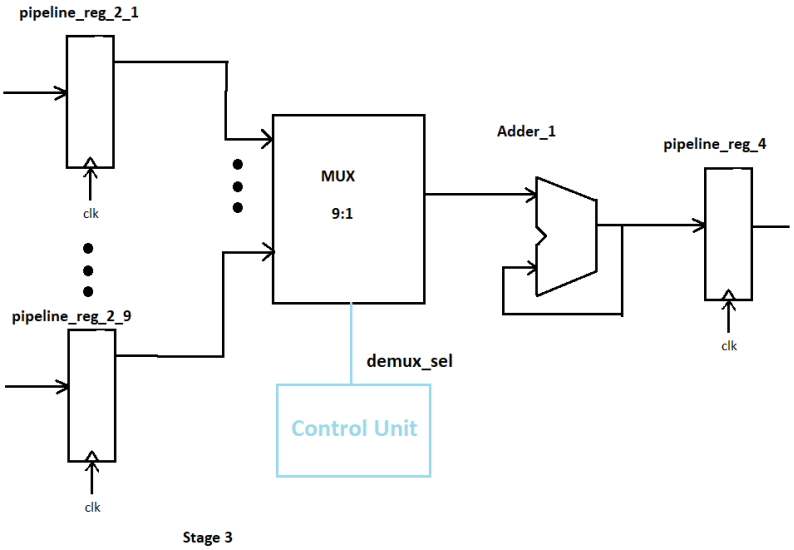

Figure 4: Stage 3

3. Results

The implemented edge detection filter was applied to a **32×32** grayscale input image. The convolution operation was performed using a standard edge detection kernel. As a result, the output image dimensions were reduced to **30×30** due to the nature of the convolution operation, where a **3×3** filter reduces the image size by **(filter size - 1)** in both dimensions.

The processed image effectively highlights the edges present in the input image, demonstrating the ability of the filter to detect intensity variations. The reduction in image size is an expected outcome, as border pixels do not have sufficient neighbors for full convolution.

Visual comparisons between the input and output images are presented in Figure 5 and 6, showing the detected edges. The results confirm the successful implementation of the convolution-based edge detection technique.

Figure 5: Input image

Figure 6: Output image

4. Conclusion

In this project, a hardware accelerator for convolution in image processing was successfully designed and implemented. The results demonstrate that the accelerator effectively applies edge detection filters, producing accurate output images with the expected reduction in size due to convolution operations. The implemented design achieves satisfactory performance for small-scale images, such as the **32×32** input used in testing.

While the current implementation processes images of limited size, the architecture can be scaled up to handle larger images by incorporating external memory for efficient data storage and retrieval. Future improvements may include optimizing memory access, parallelizing computations, and integrating advanced convolution techniques to enhance performance for real-time image processing applications.

This work highlights the potential of hardware acceleration in image processing, offering a foundation for further research and development in high-performance computing for vision-based systems.

5. References

[1] M.-C. Chang, "Hardware Accelerator for Boosting Convolution Computation in Image Classification Applications," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE 2017)* , Changhua, 2017.