Hardware Implementation of Convolutional Neural Networks

L. Esphanhan and C. Javier
Applied Research Experience in Electrical and Computer Engineering
ApREECE 2018
School of Engineering and Computer Science, Oakland University
Faculty Mentor: D. Llamocca

Introduction

A Convolutional Neural Network (CNN) is a form of artificial intelligence primarily used for image recognition and, in turn, requires the use of high-end processing computers. This work explores creating custom pipelined hardware for the three main stages of a CNN: convolution, rectification (ReLU), and pooling for image classification. The ultimate goal is to create custom hardware for all CNN stages in order to optimize the hardware for analyzing and detecting objects in images.

VHDL is used to design the custom hardware components (convolution, ReLU, and pooling); then this hardware description is mapped onto a field-programmable gate array (FPGA). For hardware verification, the outputs of the implemented custom hardware are compared to the outputs of a floating-point model in MATLAB.

Tools

Software

- Vivado Design Suite 2018.1
- MATLAB

Hardware

- ZYBO Zynq-7000 Development Board

Figure 1: Zynq-7000 Board

CNN Hardware Design

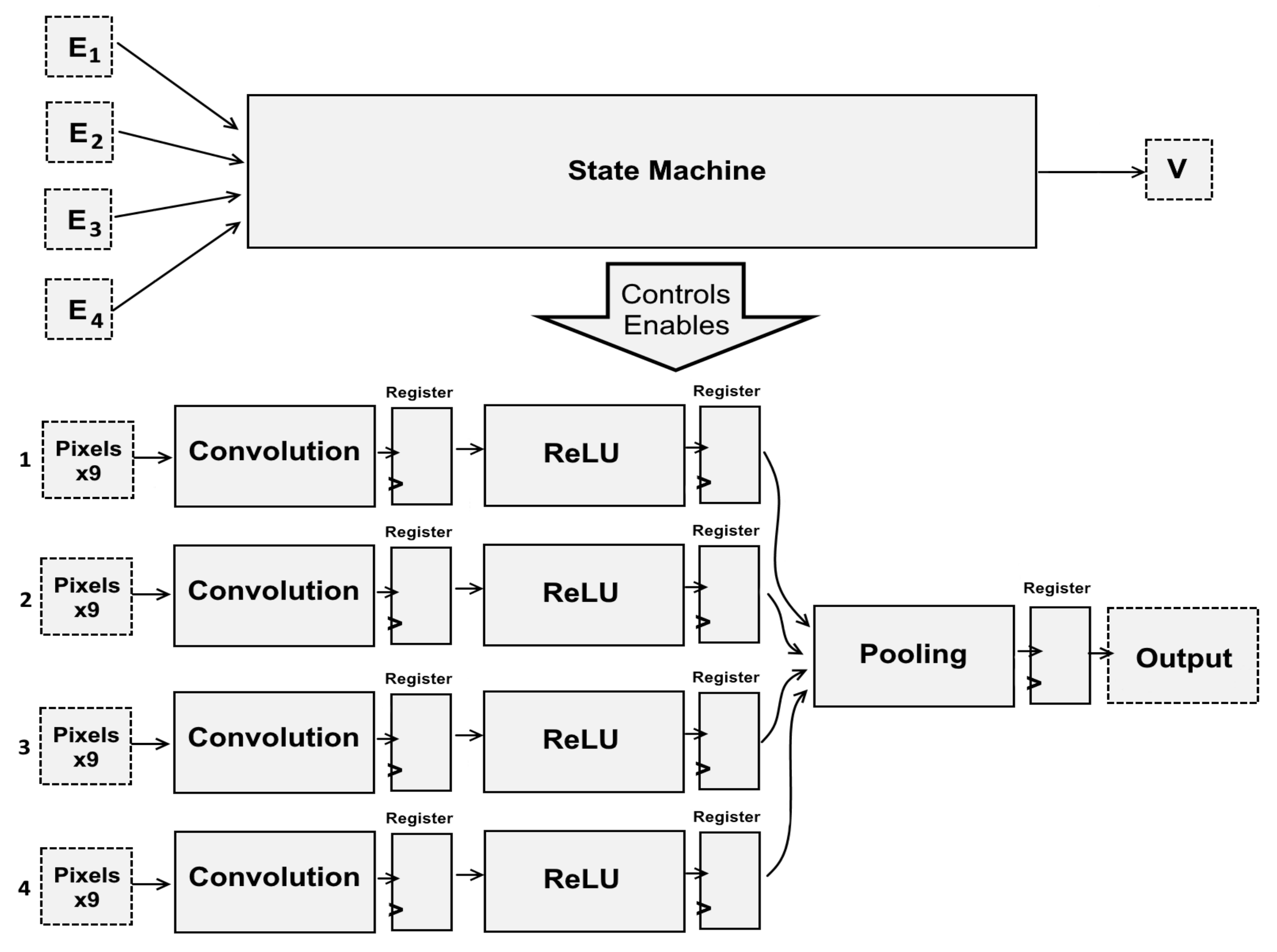

Figure 2: Design Block Diagram

The hardware description requires an enable signal and thirty six pixel values. Each convolution has 9 input pixels in a 3x3 matrix. For each 36 input pixels, it outputs one numeric value and a valid signal (v).

Within the architecture, each stage of the CNN process is a separate component.

The pooling stage requires four input values, so in order to acquire these four values a state machine keeps track of each 3x3 input convolution and ReLU. After all four pooling input registers are filled, the pooling stage is able to commence.

The valid signal is turned on after the pooling stage is done. This signal is used in the hardware verification to let the FPGA know when to write the final output value and when to re-enable this hardware architecture again.

Connecting the Custom Hardware

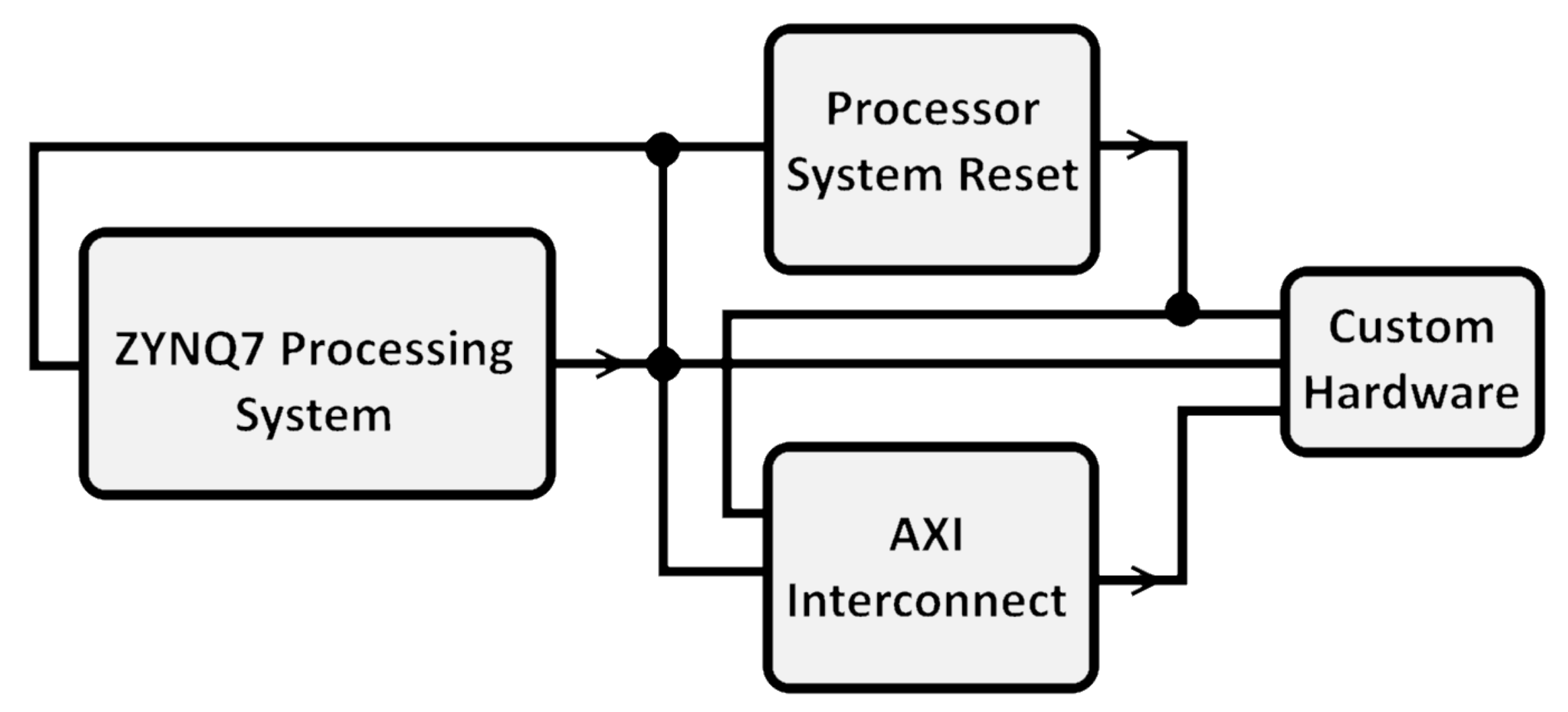

Figure 3: Connecting Hardware: Block Diagram

To connect the custom hardware loaded on the FPGA to the ZYNQ7 processing system on the ZYBO Zynq-7000 Development Board, an AXI Interconnect was modified based on the needs of the custom hardware (4 registers).

Results

The outputs of the custom hardware loaded onto the FPGA were the same as the outputs from the MATLAB code. An example of inputs/outputs can be seen in Figure 5.

If completely optimized, the designed custom CNN hardware is able to, from the first enable to valid output, complete in 14 clock cycles.

Each input required one cycle, and from the last input to the output is 10 cycles, totaling 14 clock cycles. This can be seen in Figure 6.

xA1	xB2	xC3	x31	xB2	xC3
xD4	xF0	xE1	x34	x7F	x7F
xD2	xC3	xB3	x7F	x7F	x7F
x31	xB2	xC3	x31	xB2	xC3
x34	x70	x21	xD4	xF0	xE1
x12	x03	x7F	xD2	xC3	xB3

Output: x647

Figure 5: Example 6x6 Pixel Inputs

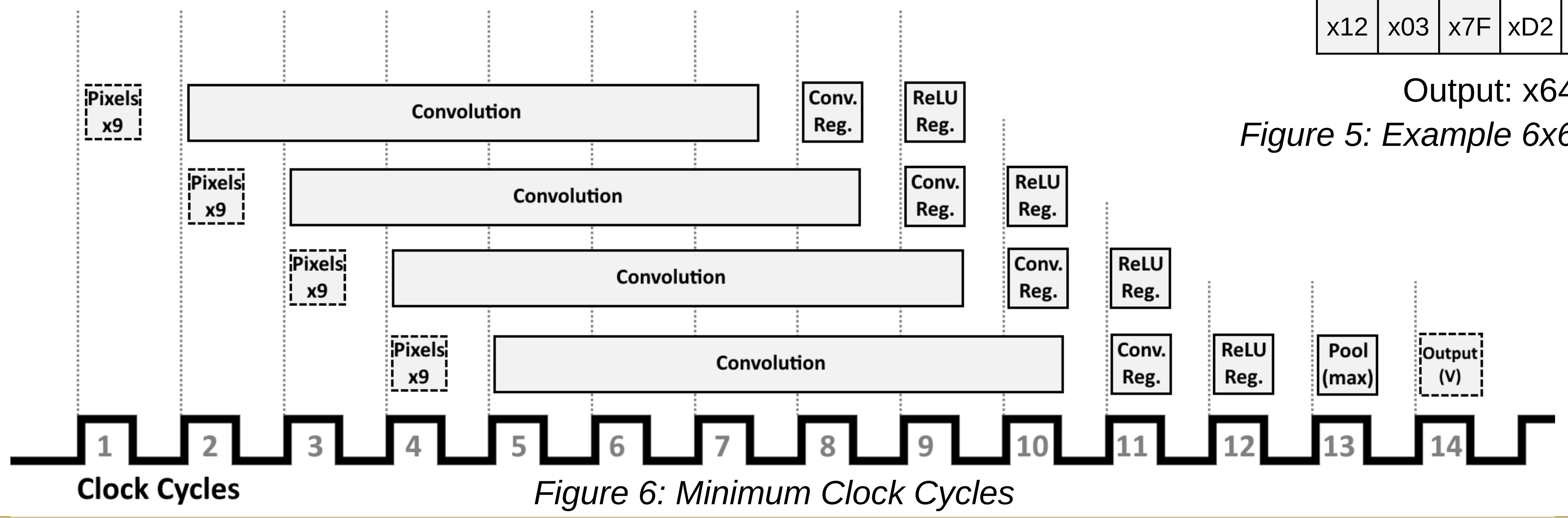

Figure 6: Minimum Clock Cycles

Conclusion

In this work, hardware has been designed for the convolution, ReLU, and pooling stages. The designs have been verified through simulations and running it on an FPGA. These steps have contributed to the creation of a fully operating Convolutional Neural Network on custom hardware.

A future goal for this project is to investigate the feasibility of designing custom hardware to implement the fully connected layer (neural network), which will enable the design of a complete hardware architecture for a CNN.

This work was sponsored in part by the National Science Foundation (NSF) under award number EEC-1659650.