ELEN6350 VLSI Design Lab
NPU--LCNet Accelerator
Project Proposal

Instructor: Pro. Peter Kinget
Teaching Assistant: Alfred Davidson
Sponsor: Apple Inc.

Yixuan Liu
Huitong Chen
Kaidi Chen
Yumeng Wang

1. Overview

Our team has developed a specialized Neural Processing Unit (NPU) chip designed to accelerate LCNet, a Lightweight Classifier Convolutional Neural Network (CNN) . It is used to classify images by extracting features through layers. And the primary advantage of our NPU lies in its architecture, featuring four parallel Processing Elements (PE) units—comprising 12 multipliers and 4 adders—designed to significantly accelerate CNN computations. A notable feature is its requirement for an external master for complete performance. This external host aids in controlling the input data flow and collecting the output data stream.

2. Features

- Power consumption:
- Operating temperature: 25 degree Celsius
- Maximum clock frequency: 10MHz
- Operating voltage: 1V
- 3 different output modes.

3. LCNet Architecture and Application Scenarios

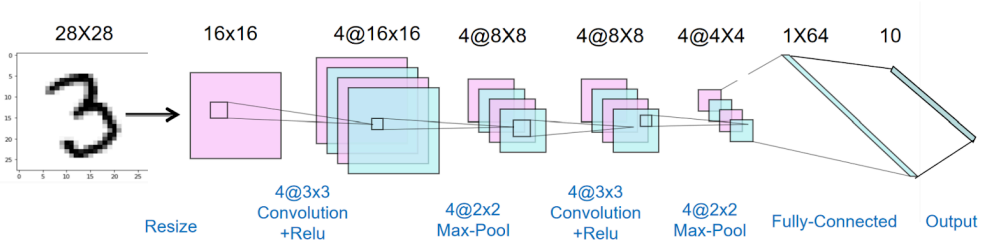


Figure 1 The structure of LCNet

We develop this LCNet for image classification. The architecture of the proposed LCNet is illustrated in Fig 1. Each of the first two layers of the LCNet consists of three parts: a convolution layer, a max-pool layer and an activation function. The details of the parameters are in Table 1.

Table 1 The parameters for LCNet

	Input	Output	Kernel size	Stride	Padding	Channel
Conv1	16x16	4@16x16	3x3	1	1	4
Maxpool1	4@16x16	4@8x8	2x2	2	NA	NA
Conv2	4@8x8	4@8x8	3x3	1	1	4
Maxpool2	4@8x8	4@4x4	2x2	2	NA	NA
Fully-Connected	4@4x4	10	4x4x4	NA	NA	10

Our Neural Processing Unit (NPU) is designed to accelerate the LCNet. So it is used to excel in AI image classification tasks. Its performance has been validated on benchmark datasets like MNIST and EMNIST, where it demonstrated flawless execution. This evidences the chip's suitability for diverse AI challenges in image classification.

4. Top Level Architecture and Functional Description

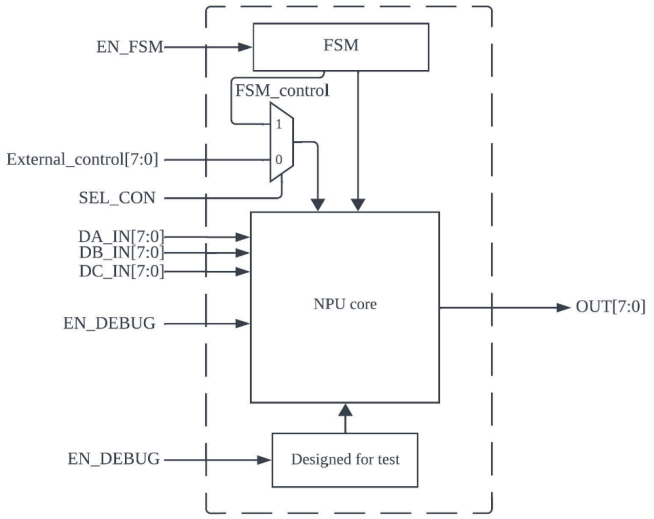


Figure 2 Top level architecture

There are three main blocks in the top level, namely FSM, NPU Core and DFT(Design for test). The FSM outputs control signals, the NPU handles main computations, and the DFT unit enhances chip robustness.

The LCNet Accelerator Chip supports three operational modes using different parts of these three blocks: two active working states--one uses FSM and NPU core, the other only uses NPU core, and a specialized debug mode using NPU core and NFT. Main functions are shown in Table 2.

Table 2 Different working modes and their functions

Mode	Function
Mode 1(AUTO mode)	Use FSM to give out control signals
Mode 2(MANUAL mode)	Use external_control to give control signals
Mode 3(DEBUG mode)	All units in NPU stop working, and output intermediate results

To enhance the usability of our chip, we placed a selector before the NPU. This means that even if the FSM breaks down, as long as this MUX remains functional, you can input control signals externally to complete all computations of the NPU. Users can pull SEL_CON pins high to use the internal FSM to generate all control signals for NPU Core automatically(Mode 1), or it can be pulled low for NPU Core to accept external manual control signals through DC port(Mode 2).

Activating the debug mode by setting the EN_DEBUG signal high turns off all NPU enabled signals, allowing for a scan test by the DFT unit to output intermediate results for efficient bug identification, making it a versatile tool for both performance and troubleshooting in applications.

5. Pin Configuration and Description

Table 3 Pin Configuration and Description

Number	I/O	Name	Function
1-8	Input	DA	Data input
9-16	Input	DB	Data input
17-24	Input	DC	Data input
25-32	Input	External_control	Input control signals for mode 2
33-40	Output	DO	Data output
41	Input	EN_FSM	Enable Finite State Machine
42	Input	RST_FSM	Reset FSM
43	Input	EN_WR	Start to write data into NPU
44	Input	EN_DEBUG	Enable mode 3
45	Input	SEL_CON	Select between mode 1 and 2
57-60	Power	GND	Ground
61-64	Power	VDD	Power source for NPU

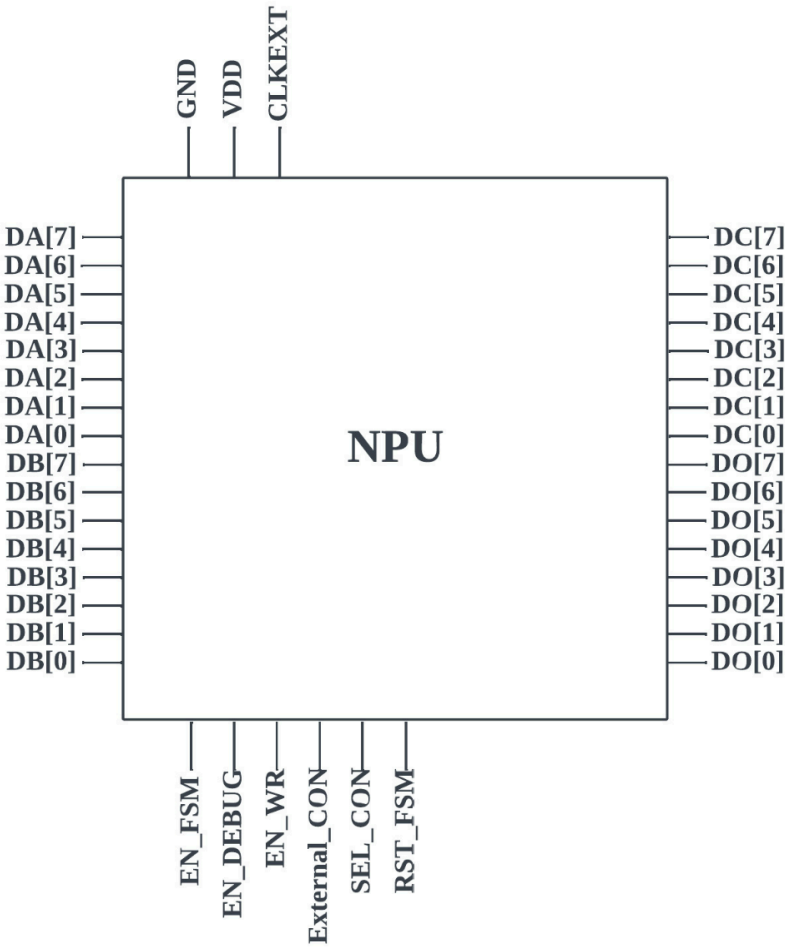


Figure 3 Pin Configuration

6. Electrical Characteristics

Table 4 electrical parameters

Parameter	Specification	Notes
Input Voltage	2.5V	Input voltage of the chip
Output Voltage	1V	Output voltage of the chip
Operating Voltage	1V	Core voltage range
Operating Current		Active mode, excluding I/O currents
Power Consumption		Total power consumption
Quiescent Current (Iq)		Sleep mode, Vdd = 1V
CLK	5MHz	Recommended

7. NPU Core Specification

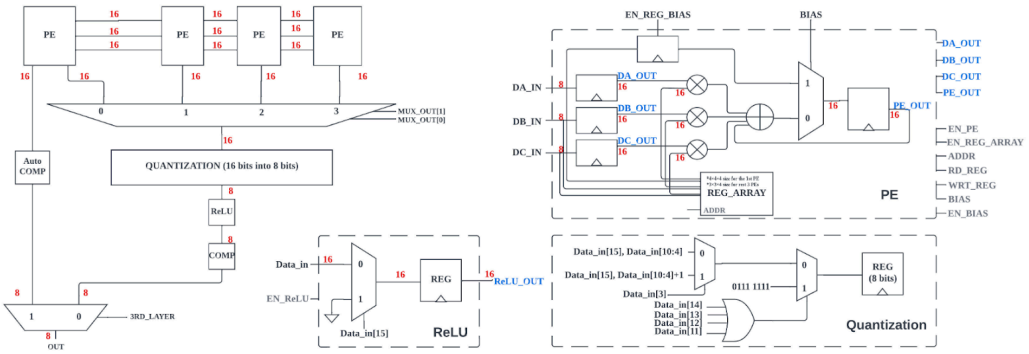


Figure 4 Block Diagram for NPU

7.1 PE Module

Four input channels are initially connected to four PE modules separately. There are a total of four parallel PE modules, each consisting of 3 8-bit multipliers, a 16-bit adder, and a register array, as shown in the top right corner of Figure 4. Each module performs the functionality of multiply, add, and store.

7.2 ReLU Module

The function of the ReLU module is to activate the result of the MAC module. It outputs zero when the input is less than zero, otherwise, it outputs the input itself. Its structure is shown in the bottom right corner of Figure 4, consisting of a MUX and a register.

7.3 Comparator Module

The function of the comparator is to compare the input with the value stored in the comparator and update the internally stored data to the larger of the two. The initial value of the 16-bit comparator is 16'h8000. This module is used for max pooling.

7.4 Auto-Comparator Module

The difference between an automatic comparator and a standard comparator lies in the addition of a counter to the automatic comparator. When the automatic comparator is enabled, the counter increments by one to record the index of the input data. This feature is crucial for tracking the sequence of inputs during operations. Apart from this distinctive mechanism, the core functioning principle of the automatic comparator remains similar to that of a standard comparator: both incorporate a storage register designed to hold the greater value after comparing the current input

with the previously stored value in the register. This setup ensures that the comparator continuously updates its stored value to reflect the highest input value encountered up to that point.

7.5 Register Array

The first register array consists of 4*4*4 registers, and the rest three each consist of 3*3*4 registers. This is because we just need the first PE for Layer 3. Its functionality and control are both modeled after SRAM. Selection is done using address signals.

7.6 Quantization Module

Our approach to quantization is as follows: the input is 16-bit with the decimal point in the middle, and the output is 8-bit, also with the decimal point in the middle. We perform quantization through a process of rounding, either rounding up or down as appropriate. Signed numbers using 2's complement representation are both implemented before and after quantization; if a 16-bit number exceeds the value of 0111.1111 (indicating an overflow), then the output is clamped to 0111.1111. Conversely, if there is an underflow, meaning the value falls below the minimum representable, the output is set to 1000.0001. This process ensures that the numerical range is strictly bounded.

8. FSM Specification

FSM (Finite State Machine) is used to automatically generate control signals to drive NPU Core. The programmable nature of the FSM allows it to adapt to various accumulations through external input, ensuring compatibility with any number of them. The block diagram of FSM is shown in Figure 5.

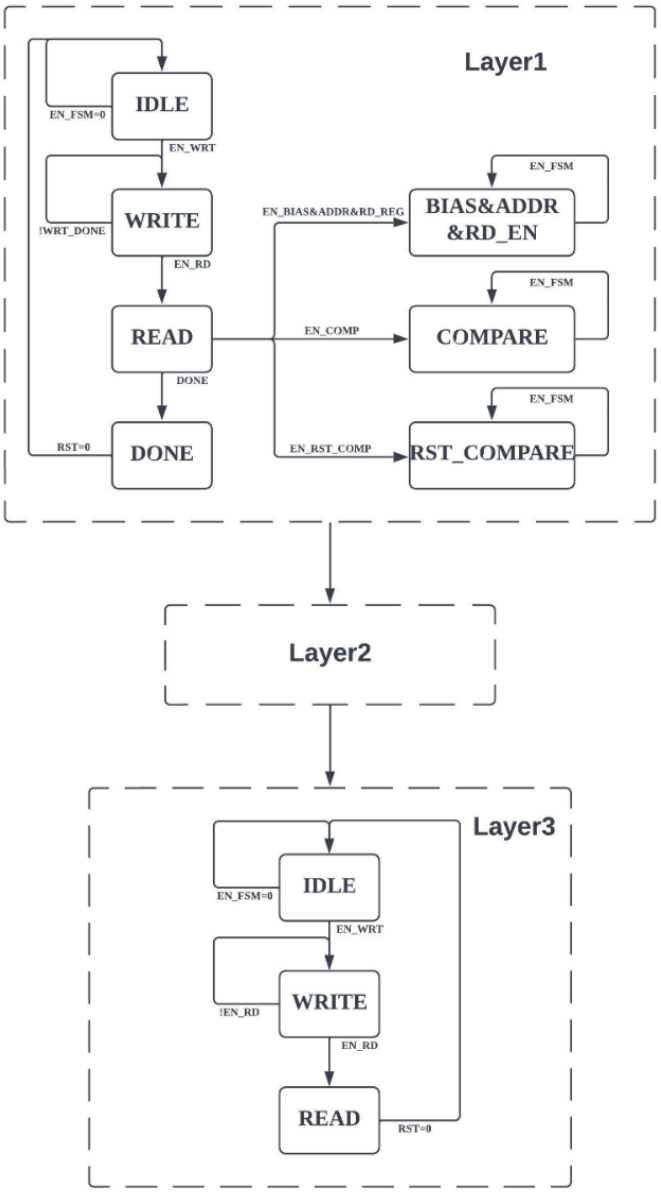
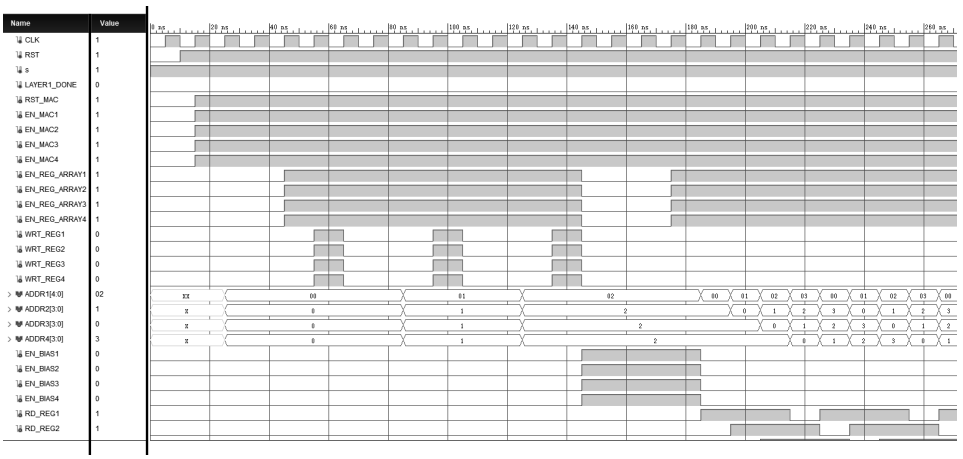


Figure 5 FSM

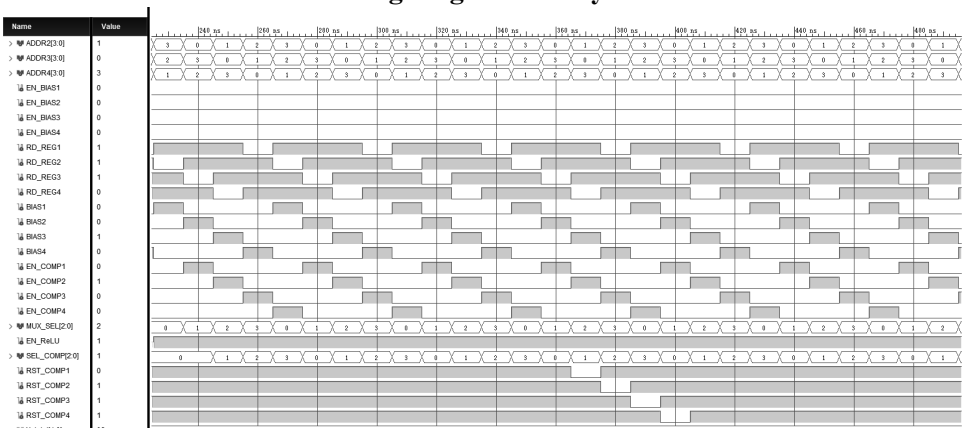
There are three main blocks in FSM,controlling each layer respectively. A break signal is implemented as the flag signal that can move the FSM to the next layer. More details are covered in the Timing Diagram section.

For the first layer, it has 4 main states. weights and bias are stored in WRITE state. Convolution starts at READ state, after 1 kernel calculation, it triggers the FSM of BIAS, ADDR, RD_EN, COMPARE, RST_COMPARE. Until a break signal comes, it moves to the FSM of the next layer. The whole process repeats once during the second layer. When it comes to the third layer, it only has 3 states. Pixels are stored in the WRITE state, and it starts to calculate and compare in the READ state.

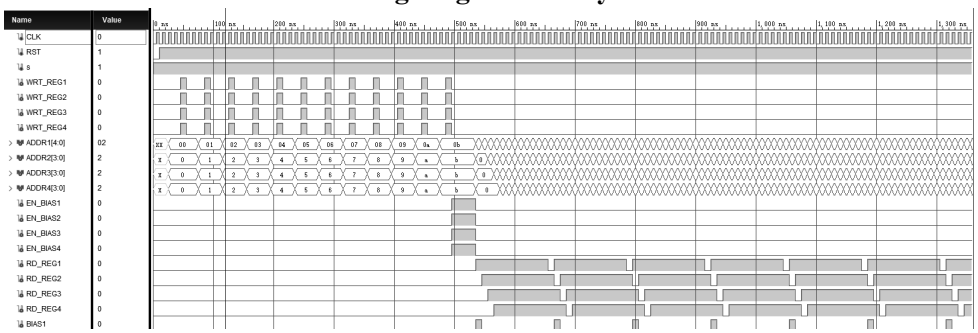
The state transition diagram and table of FSM_Layer1, FSM_Layer2 and FSM_Layer3 are shown below.



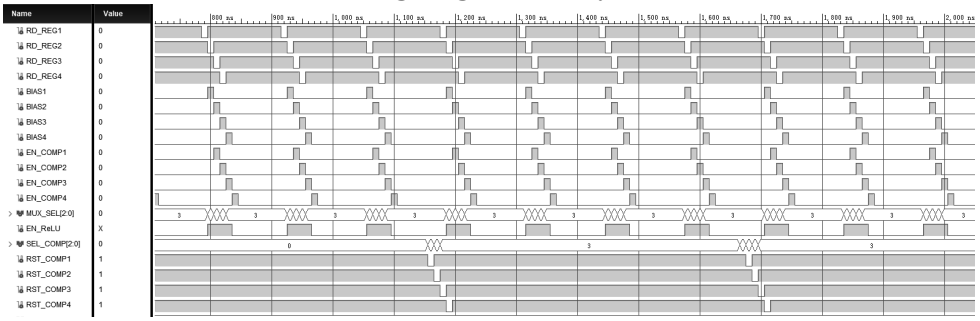
Timing diagram for layer 1



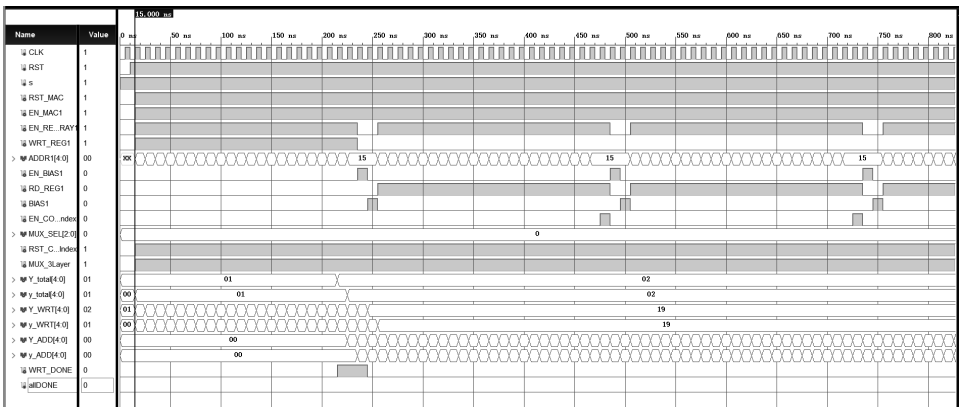
Timing diagram for layer 1



Timing diagram for layer 2



Timing diagram for layer 2



Timing diagram for layer 3

Figure 6 Timing diagram for FSM

9. DFT Specification

Our debug module, designed for post-tape-out verification, employs a methodology inspired by scan chain technology. It samples intermediate results at various points within the NPU and sequentially outputs these values for analysis. To utilize this unit for debugging purposes, operators should sequentially activate EN_DEBUG, Signal 2, and Signal 3, controlling the module to efficiently identify and address potential issues within the system. This structured approach ensures a thorough and manageable debugging process, enhancing the reliability and performance of the NPU.

10. Timing Diagram

For the first layer, the first 16 clock cycles aim to store the weights (12) and Bias (4). Input pixels start to come at the 17th time cycle. The convolution result needs to be calculated in 4 cycles once a time, using the 1st cycle to load bias and 3 cycles to do the convolution. Before the first result comes, COMP_REG stores 8000, which is the minimum value. The first Compared result comes at the 33rd time cycle, 4 compared results from 4 channels come out in sequence, and then need to wait for 12 cycles to come next time. The whole process for the first layer needs 1045 cycles.

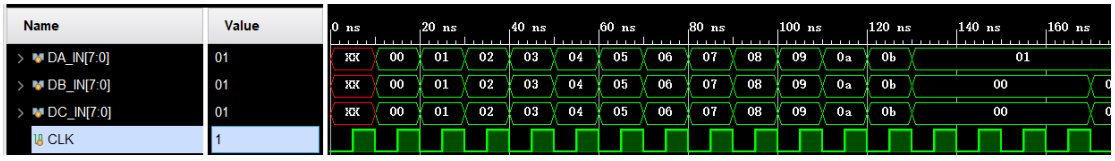


Figure 10.1 Timing diagram for storing weights and bias

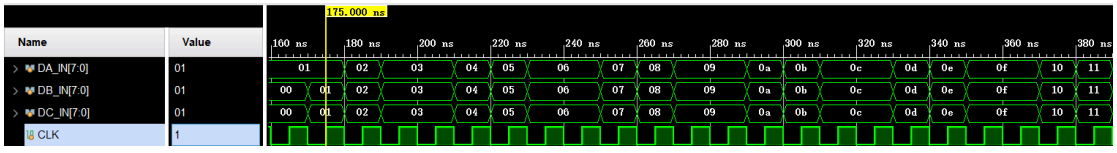


Figure 10.2 Timing diagram for coming pixels

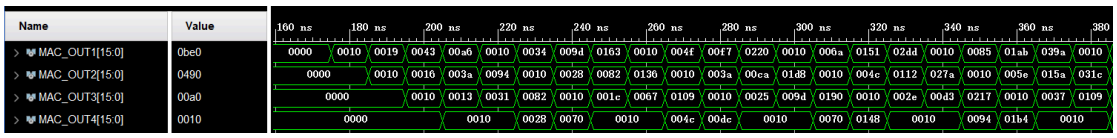


Figure 10.3 Timing diagram for convolution results

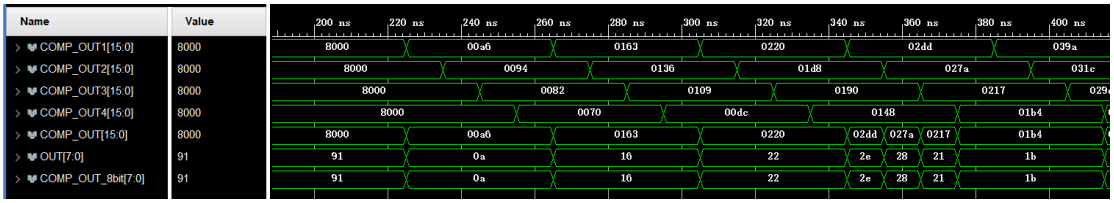


Figure 10.4 Timing diagram for final result of layer1

For the next layer, it takes 52 clock cycles to store weight (48) and bias (4). Input pixels start to come at the 53rd time cycle. The convolution result needs to be calculated in 13 cycles at once a time, using the 1st cycle to load bias and 12 cycles to do the convolution. First Compared result comes at the 67th time cycle, 4 compared results from 4 channels come out in sequence, and then need to wait for 49 cycles to come next time. The whole time cycle for comparing is 52 (3*14). The whole process for the second layer needs 850 cycles.

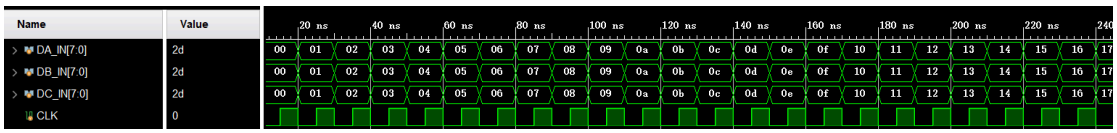


Figure 10.5 Timing diagram for storing weights and bias

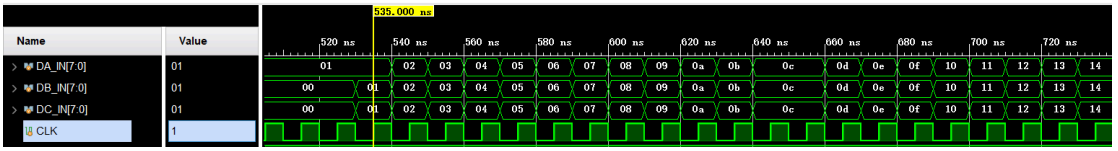


Figure 10.6 Timing diagram for coming pixels

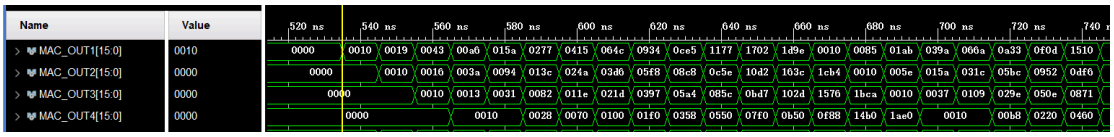


Figure 10.7 Timing diagram for convolution results

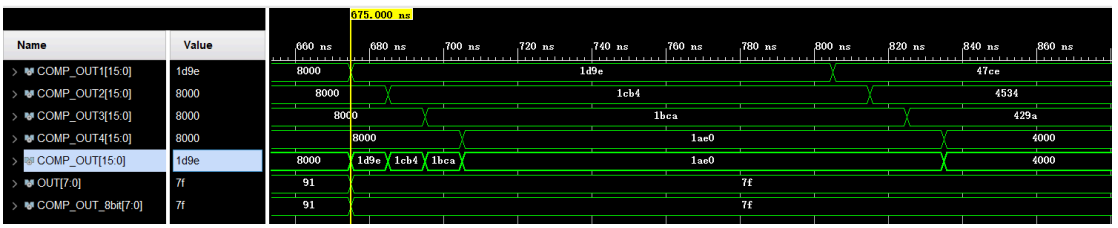


Figure 10.8 Timing diagram for final results

For the last layer, the first 22 clock cycle is aiming to store the pixels and then it takes 2 time cycles to input and load Bias. Input weights start to come at the 25th time cycle. The convolution result needs to be calculated in 23 cycles once a time, using the 1st cycle to load bias and 22 cycles to do the convolution. Before the first result

comes, Comp_Index stores 8000, which is the minimum value. First Compared result comes at the 47th time cycle, it is bigger than 8000,so index is 0 and this data is stored in the Comp_Index ; when the second result comes and is bigger than the data stored in Comp_Index, Index becomes 1. The whole process for the last layer needs 245 cycles.

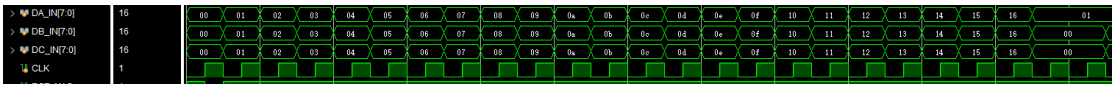


Figure 10.9 Timing diagram for storing pixels

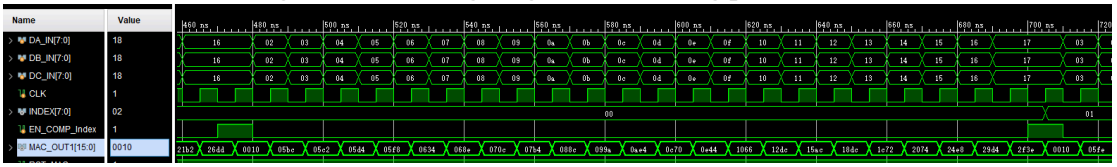


Figure 10.10 Timing diagram for the result

Reference

1.OpenAI. "ChatGPT: Optimizing Language Models for Dialogue." Accessed in 2023.

2.Team 08, EE6350 Spring 2023, Columbia University. "Neural Processing Unit Project Website." Accessed in 2023. https://www.ee.columbia.edu/~kinget/EE6350_S23/team08_npu_website/architecture.html.1.