Lab7: CNN Processing System

Instructor: Lih-Yih Chiou

Speaker: May

Date: 2020/4/15

EE

LPHPLAB VLSI Design LAB

Outline

□ Application

□ Network Architecture

□ Hardware Architecture

□ Results Demo

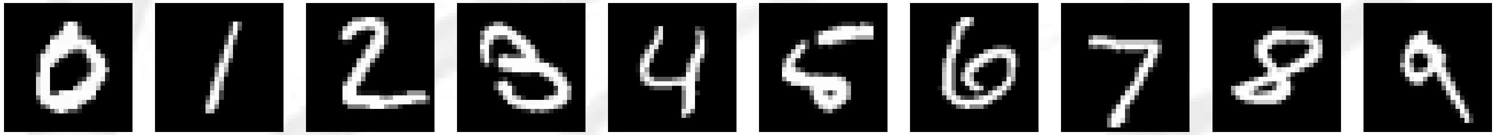
□ Homework Requirements

□ Appendix

Application

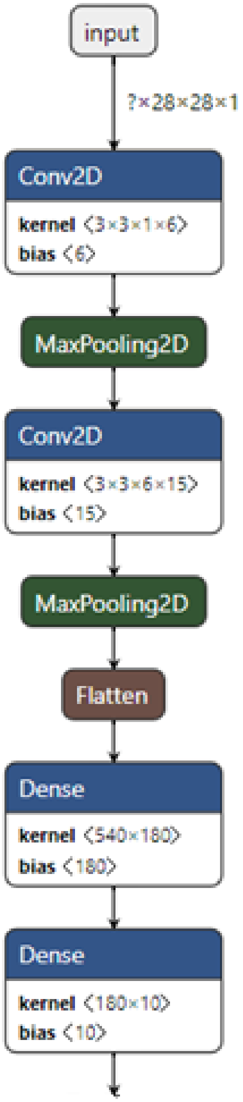
Application

- Use trained neural network to do hand-written digits classification.
- Dataset: MNIST
  - Image size: 28x28x1
  - Categories: digit 0 to 9



Network Architecture

Network Architecture (1/2)



28*28*1 =>
30*30*1(zero-padding)

28*28*6

14*14*6

12*12*15

6*6*15

540

180

10

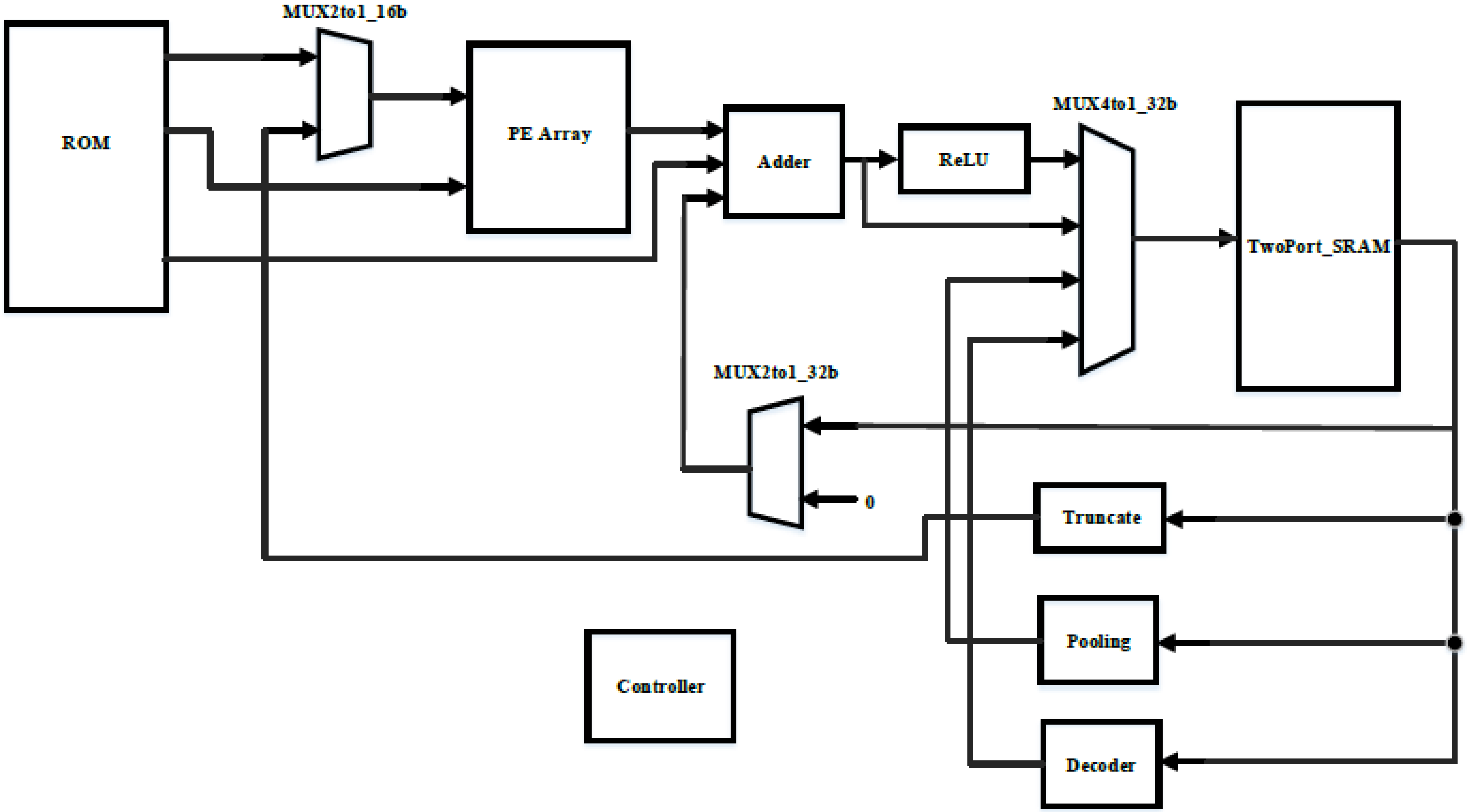
Network Architecture (2/2)

Layer name	Input size	Parameters	Output size
Conv0	30*30*1	Filters: 3*3*1, 6 Bias: 6	28*28*6
Pooling1	28*28*6		14*14*6
Conv2	14*14*6	Filters: 3*3*6, 15 Bias: 15	12*12*15
Pooling3	12*12*15		6*6*15
FC4	540	Weights: 540*180 Bias: 180	180
FC5	180	Weights: 180*10 Bias: 10	10

The layer marked as red will apply ReLU activation function

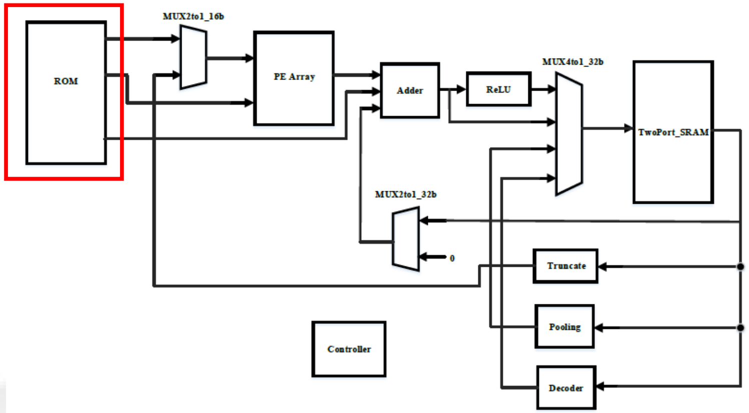
Hardware Architecture

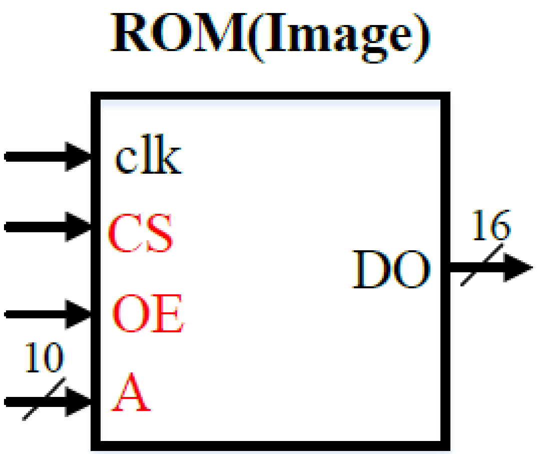
Hardware Architecture – Simple Block Diagram



Hardware Architecture – ROM(Image) (1/2)

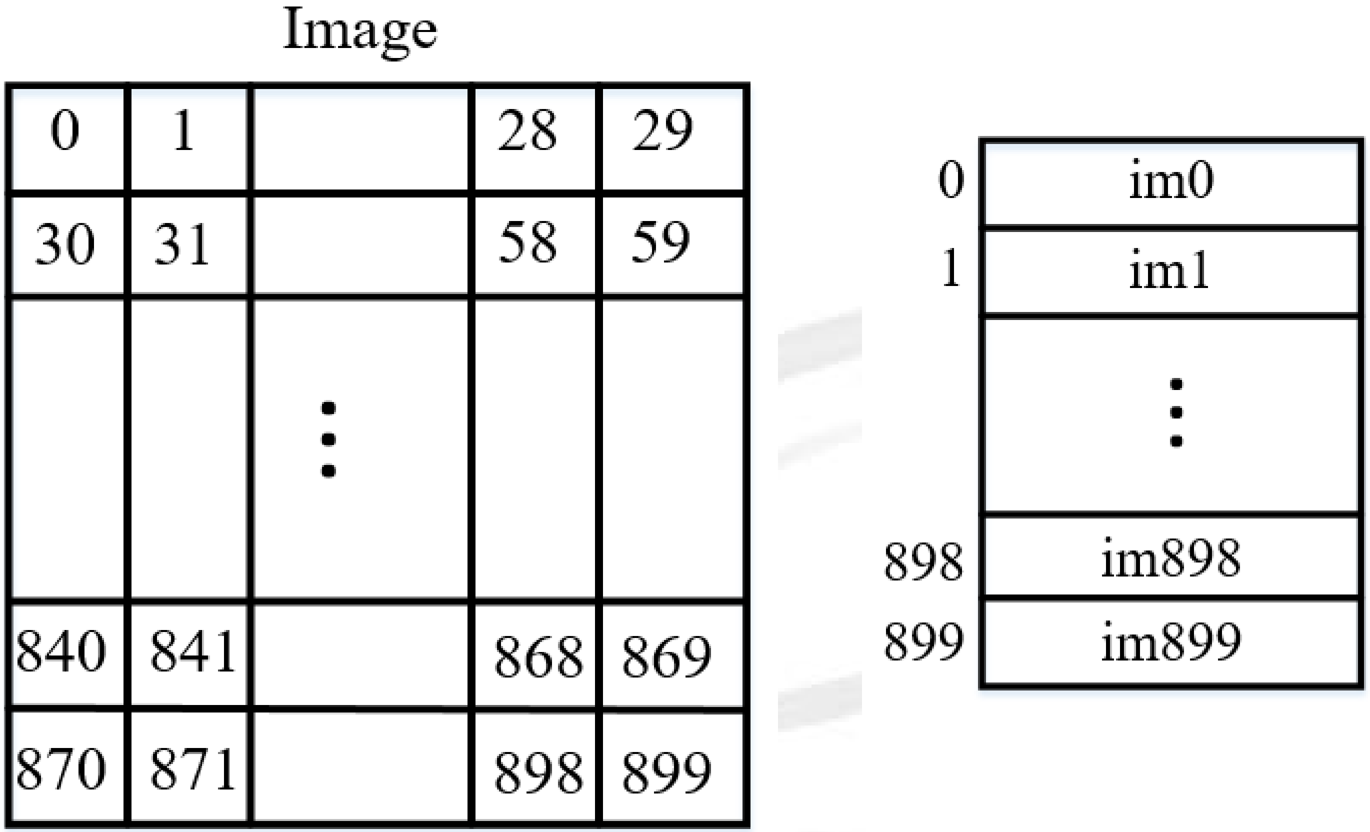
- Store 30*30*1 image(with zero padding)
- Size: 16bits*1024





Port	Description
clk	Clock
CS	Chip select
OE	Output enable
A	Address

Hardware Architecture – ROM(Image) (2/2)



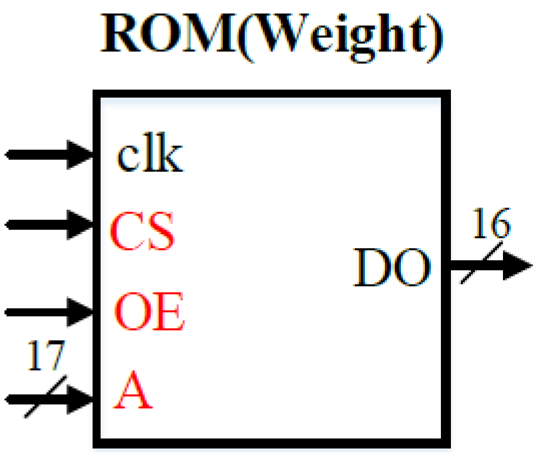
Image

0	1		28	29
30	31		58	59
		⋮		
840	841		868	869
870	871		898	899

0	im0
1	im1
	⋮
898	im898
899	im899

Hardware Architecture – ROM(Weights) (1/5)

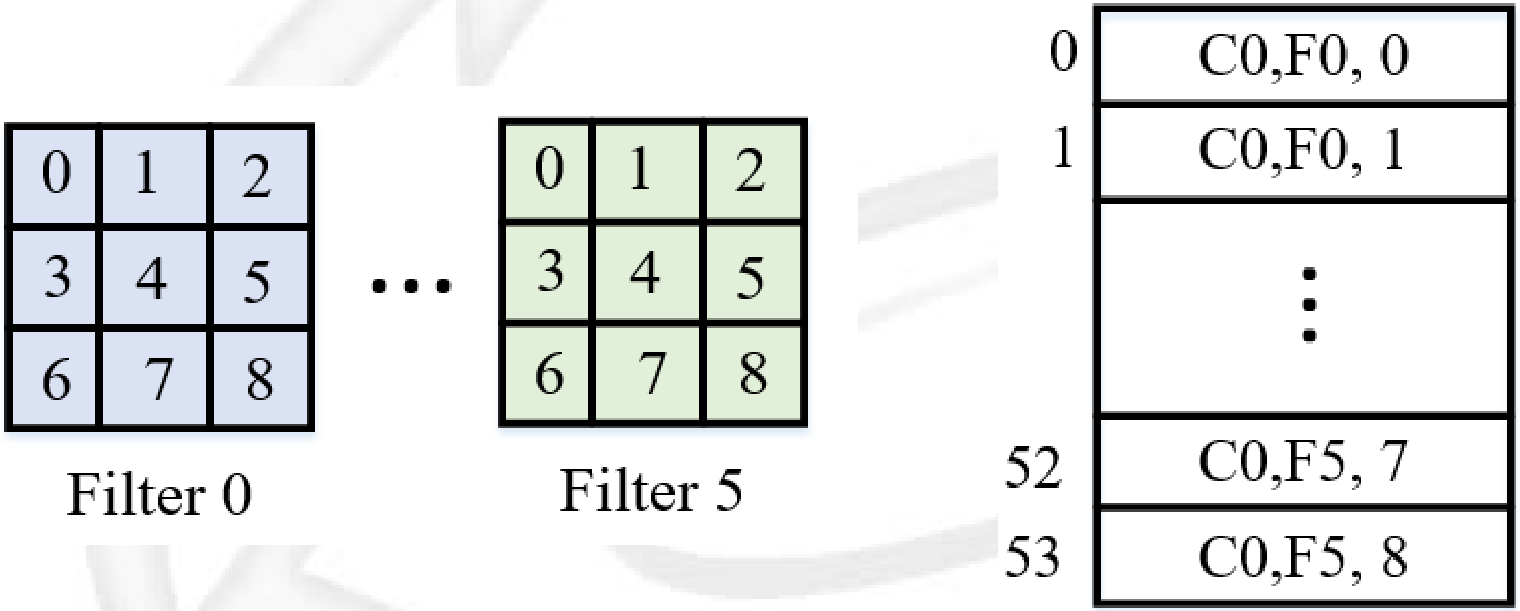
- Store network weights
- Size: 16bits*131072



Port	Description
clk	Clock
CS	Chip select
OE	Output enable
A	Address

Hardware Architecture – ROM(Weights) (2/5)

- Convolution layer 0 (Conv0)
- Filters: 3*3*1, 6 (total = 54)



0	1	2
3	4	5
6	7	8

Filter 0

…

0	1	2
3	4	5
6	7	8

Filter 5

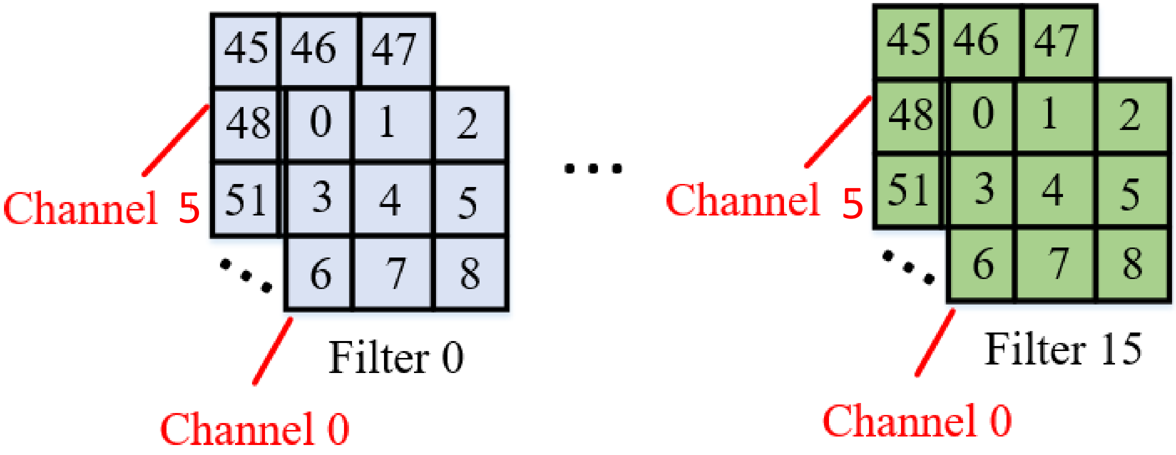
0	C0,F0, 0
1	C0,F0, 1
	⋮
52	C0,F5, 7
53	C0,F5, 8

C: Convolution

F: Filter

Hardware Architecture – ROM(Weights) (3/5)

- Convolution layer 2 (Conv2)
- Filters: 3*3*6, 15 (total = 810)



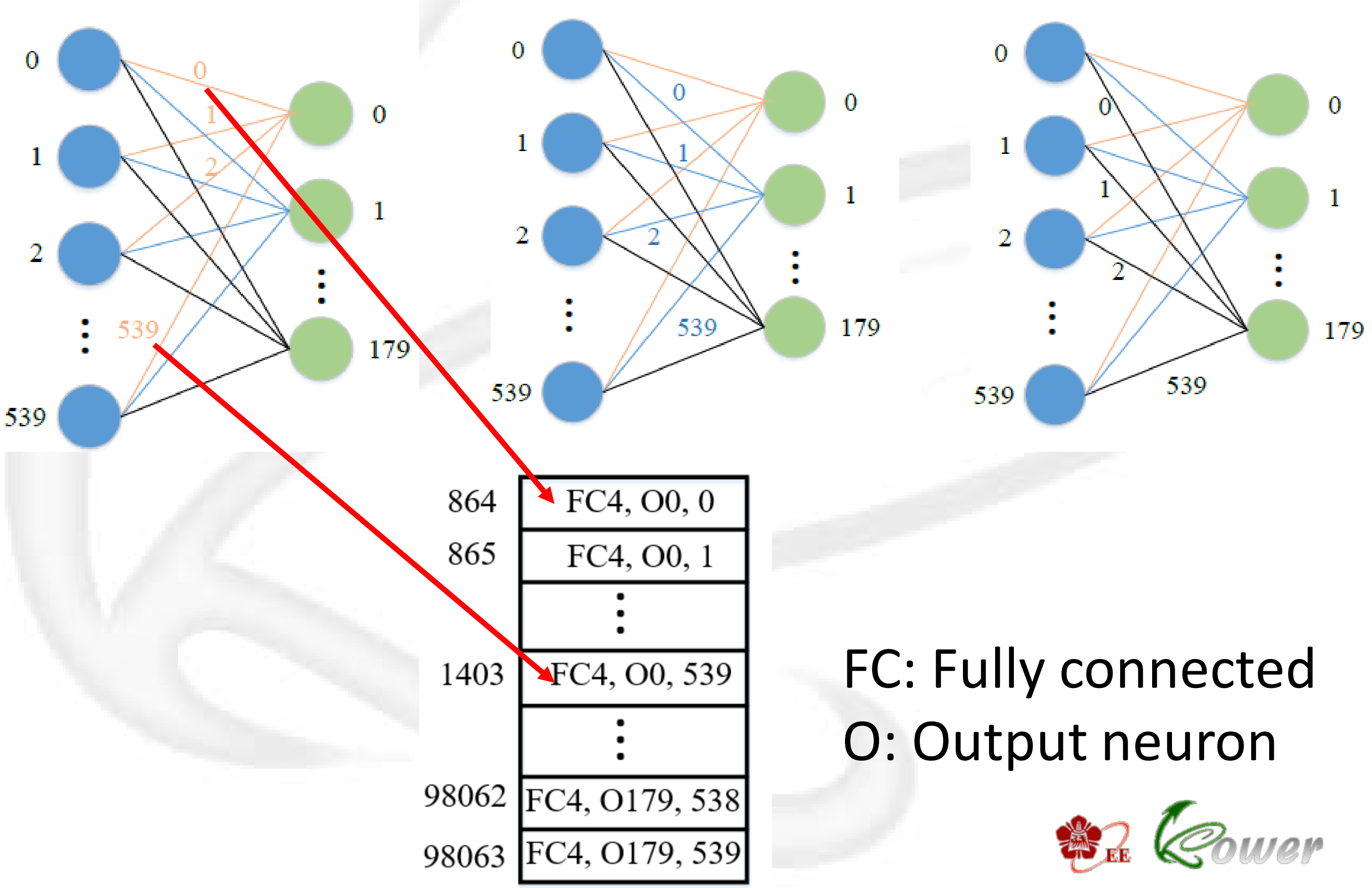
54	C2,F0, 0
55	C2,F0, 1
	⋮
863	C2,F15, 52
863	C2,F15, 53

C: Convolution

F: Filter

Hardware Architecture – ROM(Weights) (4/5)

- Fully connected layer 4 (FC4)
- Weights: 540*180 (total = 97200)

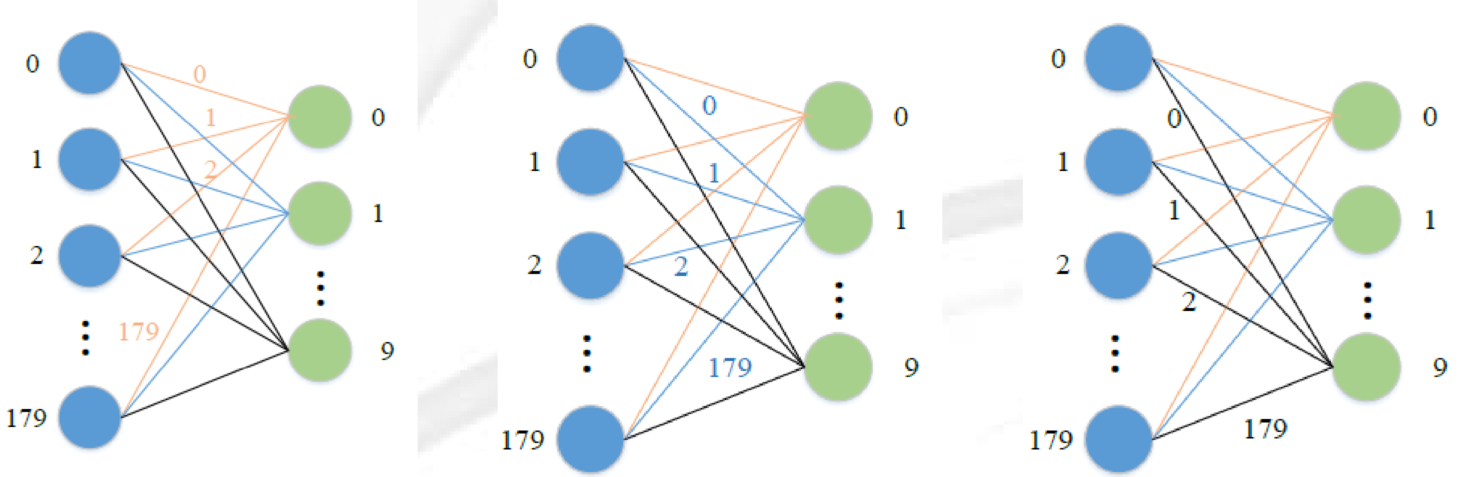


864	FC4, O0, 0
865	FC4, O0, 1
	⋮
1403	FC4, O0, 539
	⋮
98062	FC4, O179, 538
98063	FC4, O179, 539

FC: Fully connected
O: Output neuron

Hardware Architecture – ROM(Weights) (5/5)

- Fully connected layer 5 (FC5)
- Weights: 180*10 (total = 1800)

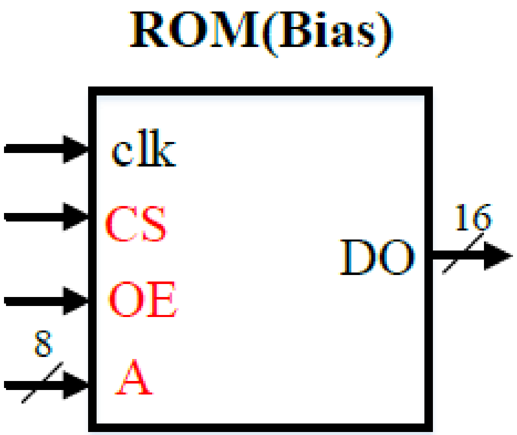


98064	FC5, O0, 0
98065	FC5, O0, 1
	⋮
98243	FC5, O0, 179
	⋮
99862	FC5, O9, 178
99863	FC5, O9, 179

FC: Fully connected
O: Output neuron

Hardware Architecture – ROM(Bias) (1/2)

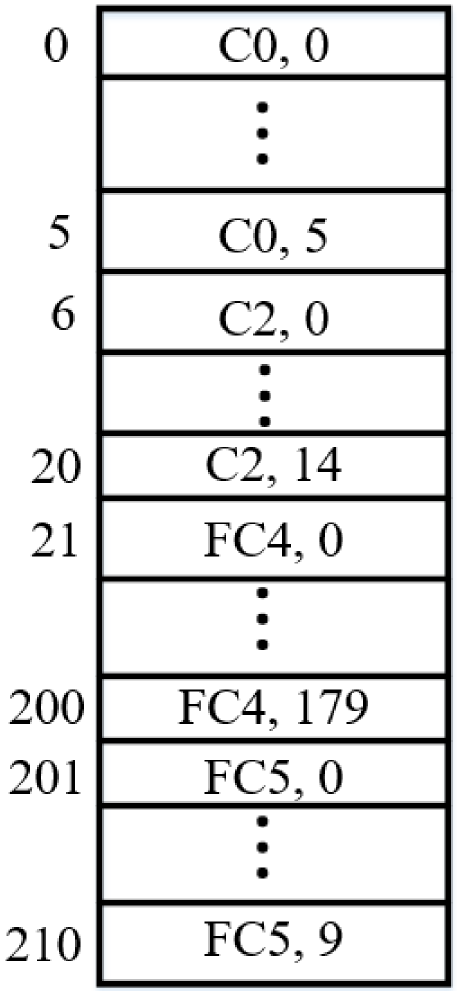
- Store network bias
- Size: 16bits*256



Port	Description
clk	Clock
CS	Chip select
OE	Output enable
A	Address

Hardware Architecture – ROM(Bias) (2/2)

Layer	# of bias
Conv 0	6
Conv 2	15
FC 4	180
FC 5	10



0 C0, 0

:

5 C0, 5

6 C2, 0

:

20 C2, 14

21 FC4, 0

:

200 FC4, 179

201 FC5, 0

:

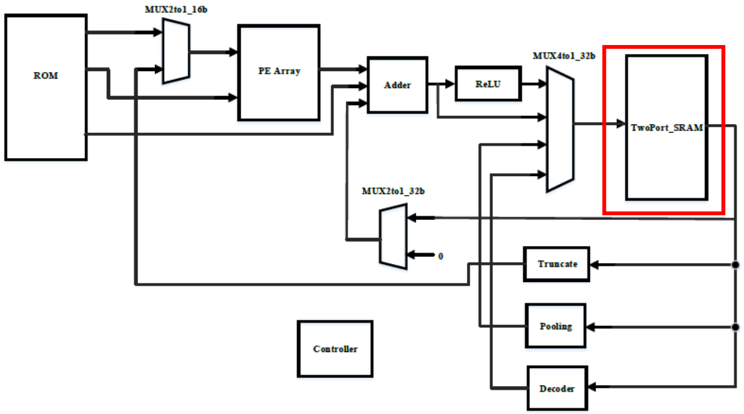
210 FC5, 9

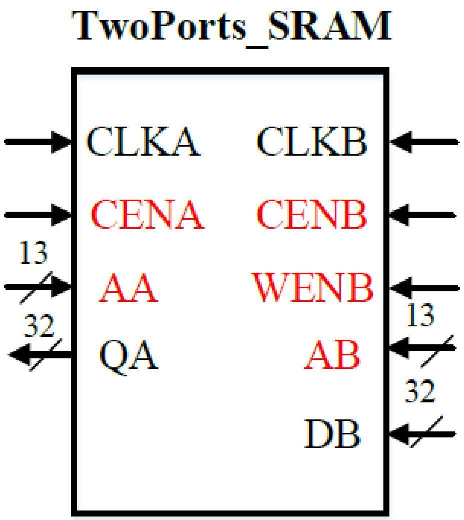
C: Convolution

FC: Fully connected

Hardware Architecture – Two Port SRAM (1/3)

- Store the partial sum or final result for each layer.
- Size: 32bits*8192

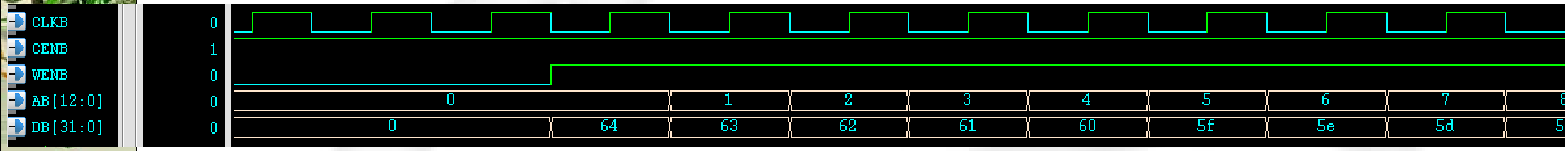




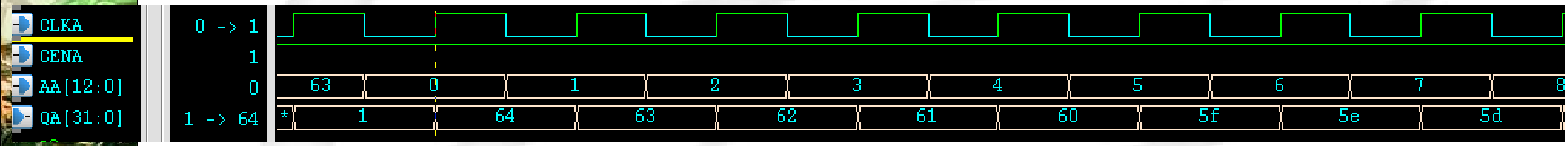
Port	Description
CLKA	Clock signal A
CLKB	Clock signal B
CENA	Chip enable A
CENB	Chip enable B
AA	Address A
AB	Address B
WENB	Write enable B
QA	Data output A
DB	Data input B

Hardware Architecture – Two Port SRAM (2/3)

- Port A only can read.
- Port B only can write.



(Write data into memory)



(Read data from memory)

Hardware Architecture – Two Port SRAM (3/3)

- Hint:
  - Divide two port SRAM into two parts.
  - First part (mem[0]-mem[4703]): Store the partial sum or result of Conv0, Conv2, FC1, Decode
  - Second part (mem[4704]-mem[8191]): Store the partial sum or result of Pooling1, Pooling3, FC2

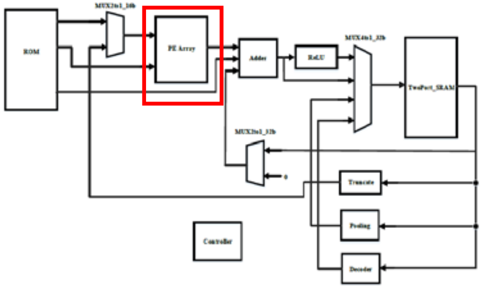
Layer	Mem address
Conv0	mem[0] – mem[4703]
Pooling1	mem[4704] – mem[5879]
Conv2	mem[0] – mem[2159]
Pooling3	mem[4704] – mem[5243]
FC4	mem[0] – mem[179]
FC5	mem[4704] – mem[4713]
Decode	mem[0]

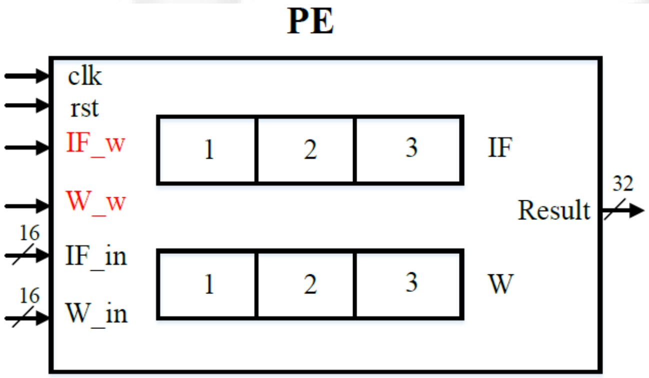
Hardware Architecture – PE (Processing Element) (1/2)

- PE(MAC in Lab5) has two shift registers (IF and W) store input feature maps and weights respectively.
- Perform 3 multiplications and 2 additions

  (Signed arithmetic, 2’s complement for negative value)

  ➔ Result = IF(1)*W(1)+IF(2)*W(2)+IF(3)*W(3)
- You can not consider the problem of overflow.

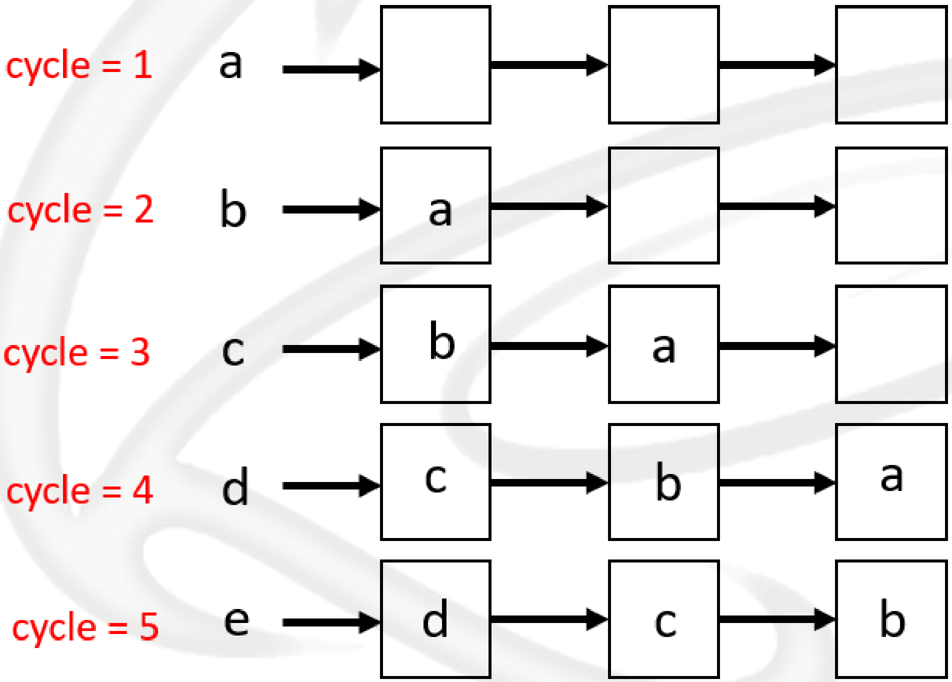




If not write enable, shift registers keep its old value.

Port	Description
clk	Clock signal
IF_w	Input features shift registers write enable
W_w	Weights shift registers write enable
IF_in	Input feature map
W_in	Weight
Result	Result

Hardware Architecture – PE (Processing Element) (2/2)



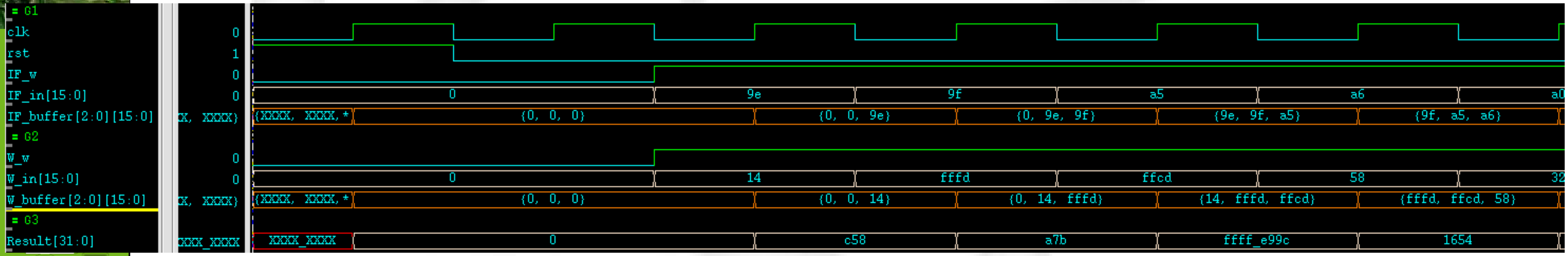
cycle = 1 a

cycle = 2 b a

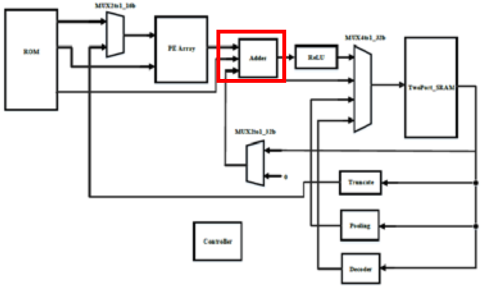
cycle = 3 c b a

cycle = 4 d c b a

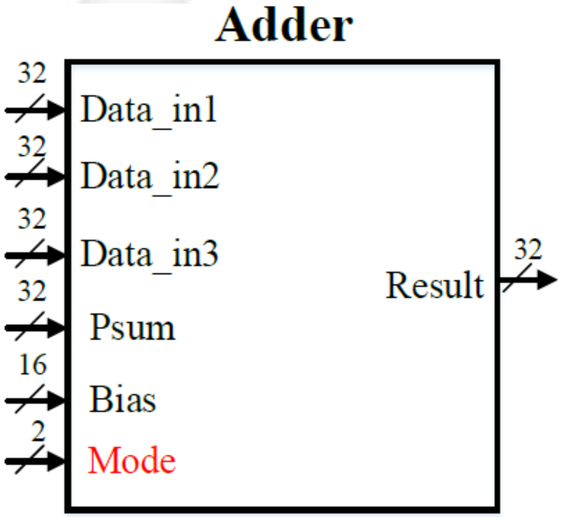
cycle = 5 e d c b



Hardware Architecture – Adder



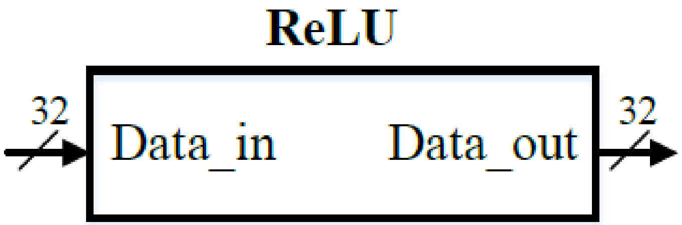
- Do add operation (signed arithmetic)
- Adder has 3 modes:
  - Mode 0: Data_in1+Data_in2+Data_in3
  - Mode 1: Data_in1+Data_in2+Data_in3 +partial sum
  - Mode 2: Data_in1+Data_in2+Data_in3 +partial sum+bias

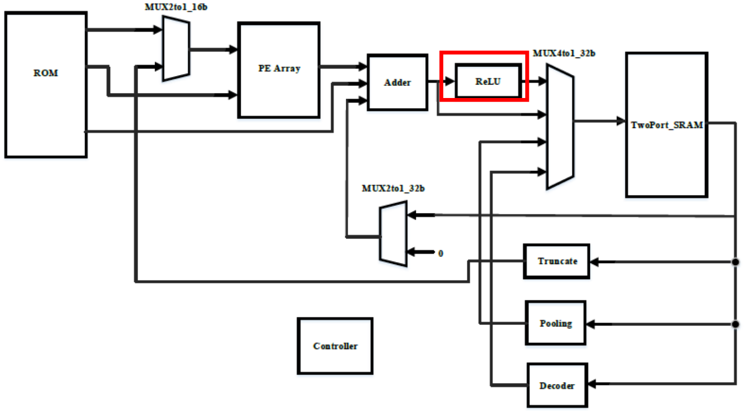


Port	Description
Data_in1	Data input1
Data_in2	Data input2
Data_in3	Data input3
Psum	Partial sum
Bias	Bias
Mode	Adder mode
Result	Result

Hardware Architecture – ReLU

- Implement ReLU function (signed arithmetic)
- $$\begin{cases} Data\_out = 0, \; Data\_in < 0 \\ Data\_out = Data\_in, \; Data\_in \geq 0 \end{cases}$$

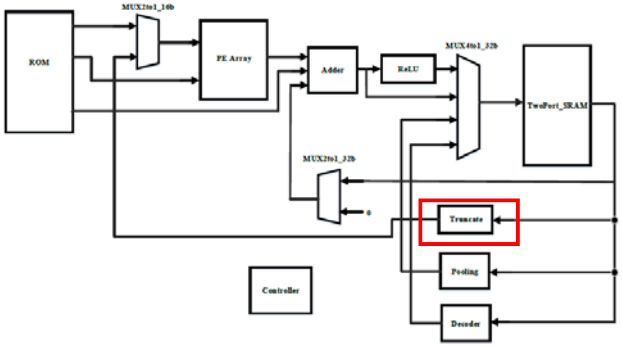


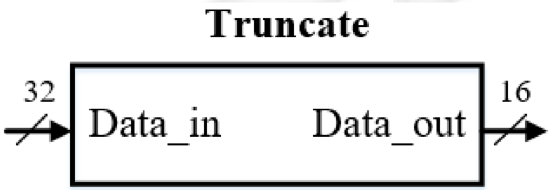


Port	Description
Data_in	Data input
Data_out	Data output

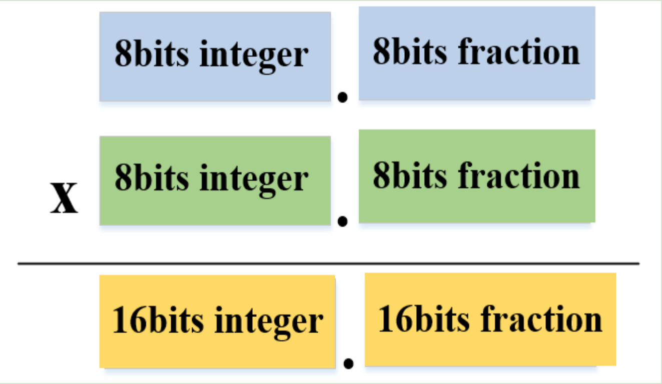
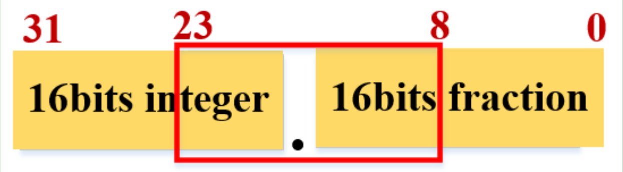
Hardware Architecture – Truncate

- Truncate 32bits data to 16bits data.



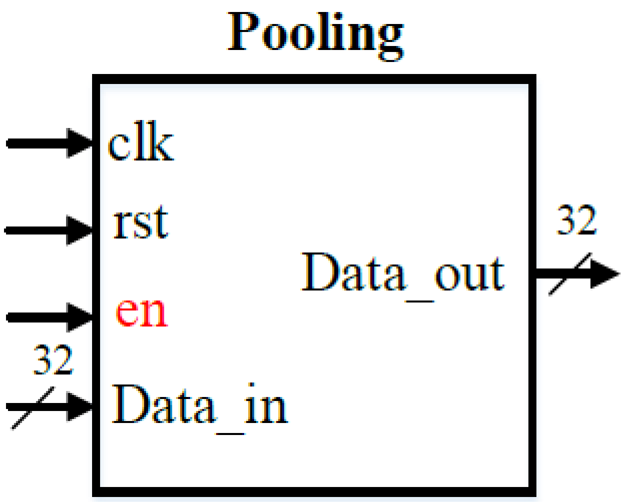


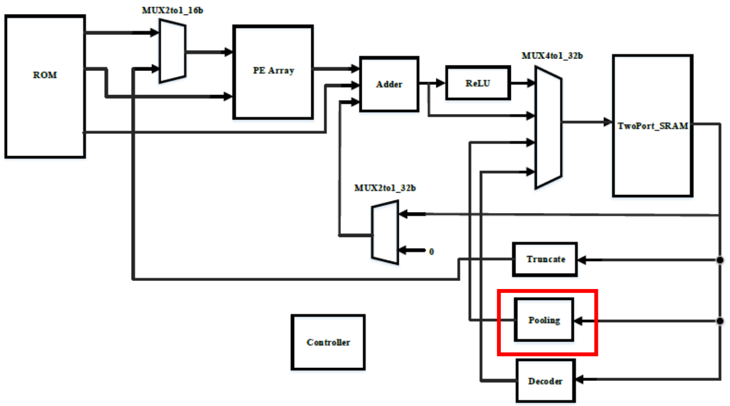
Port	Description
Data_in	Data input
Data_out	Data output

PE	Truncate
	

Hardware Architecture – Pooling

- Do max pooling operation. (signed arithmetic)
- Read 4 input feature map data, then write the maximal value back to two port SRAM.

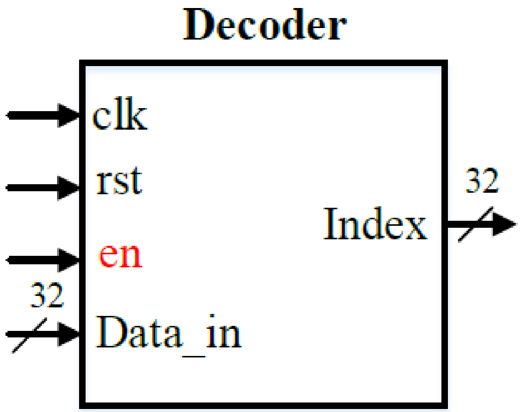


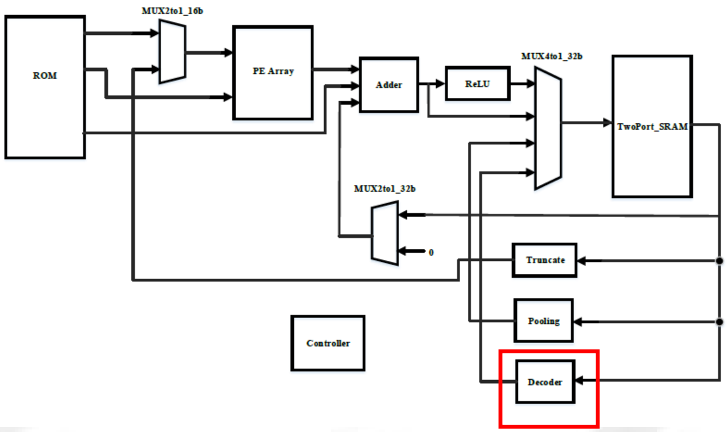


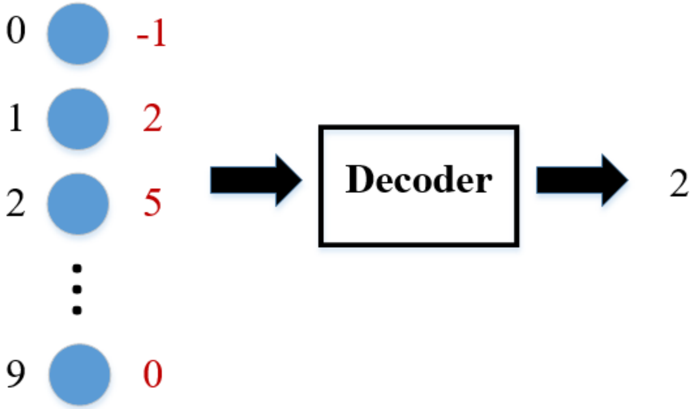
Port	Description
clk	Clock signal
rst	Reset signal
En	Max pooling enable
Data_in	Data input
Data_out	Data output

Hardware Architecture – Decoder

- Read FC5 10 results, then find which output neuron has maximum value, write its index back to two port SRAM[0].



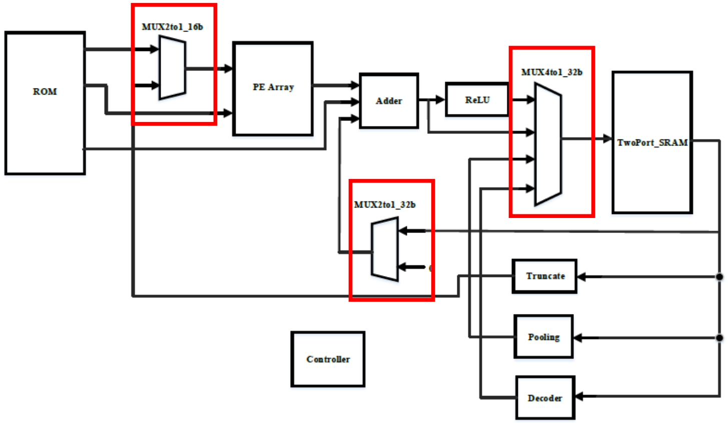


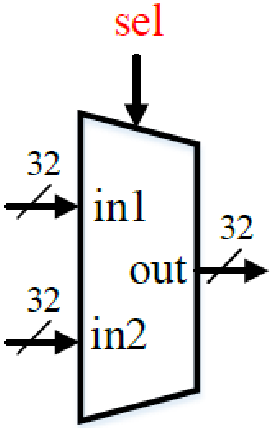


Port	Description
clk	Clock signal
rst	Reset signal
en	Decode operation enable
Data_in	Data input
Index	Data output

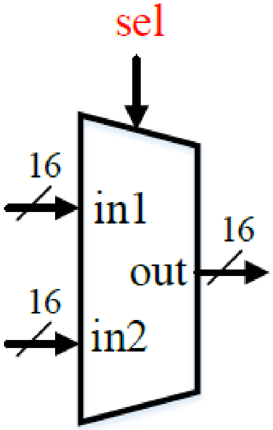
Hardware Architecture – Multiplexer

- There are 3 multiplexers in the design
  - 16bits MUX2to1
  - 32bits MUX4to1
  - 32bits MUX2to1

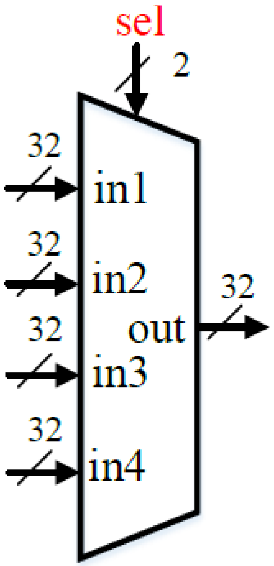




MUX2to1_32b

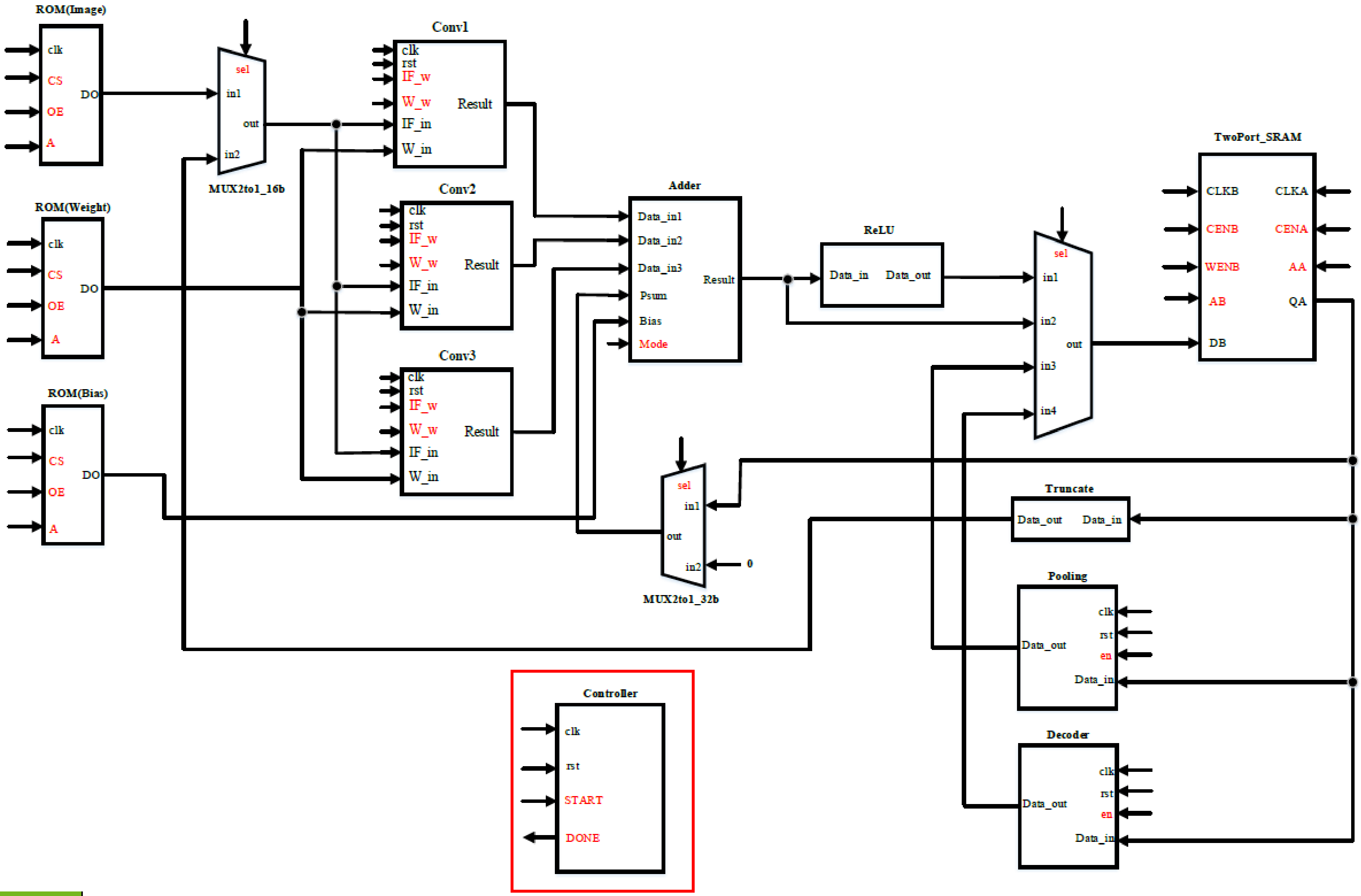


MUX2to1_16b



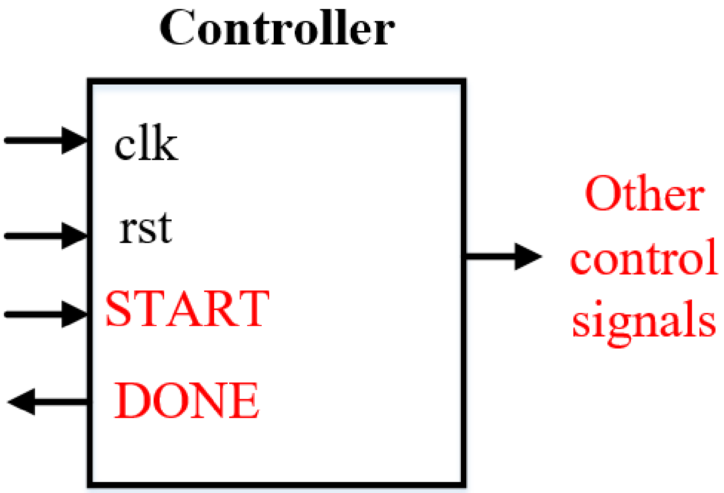
MUX4to1_32b

Hardware Architecture – Overall



Hardware Architecture – Controller

- Generate all control signals (The signals that marked as red)
- Special signals:
  - START: Testbench will give controller a START signal to start CNN Processing System.
  - DONE: When CNN Processing System finish detecting an image, controller will give testbench a DONE signal.

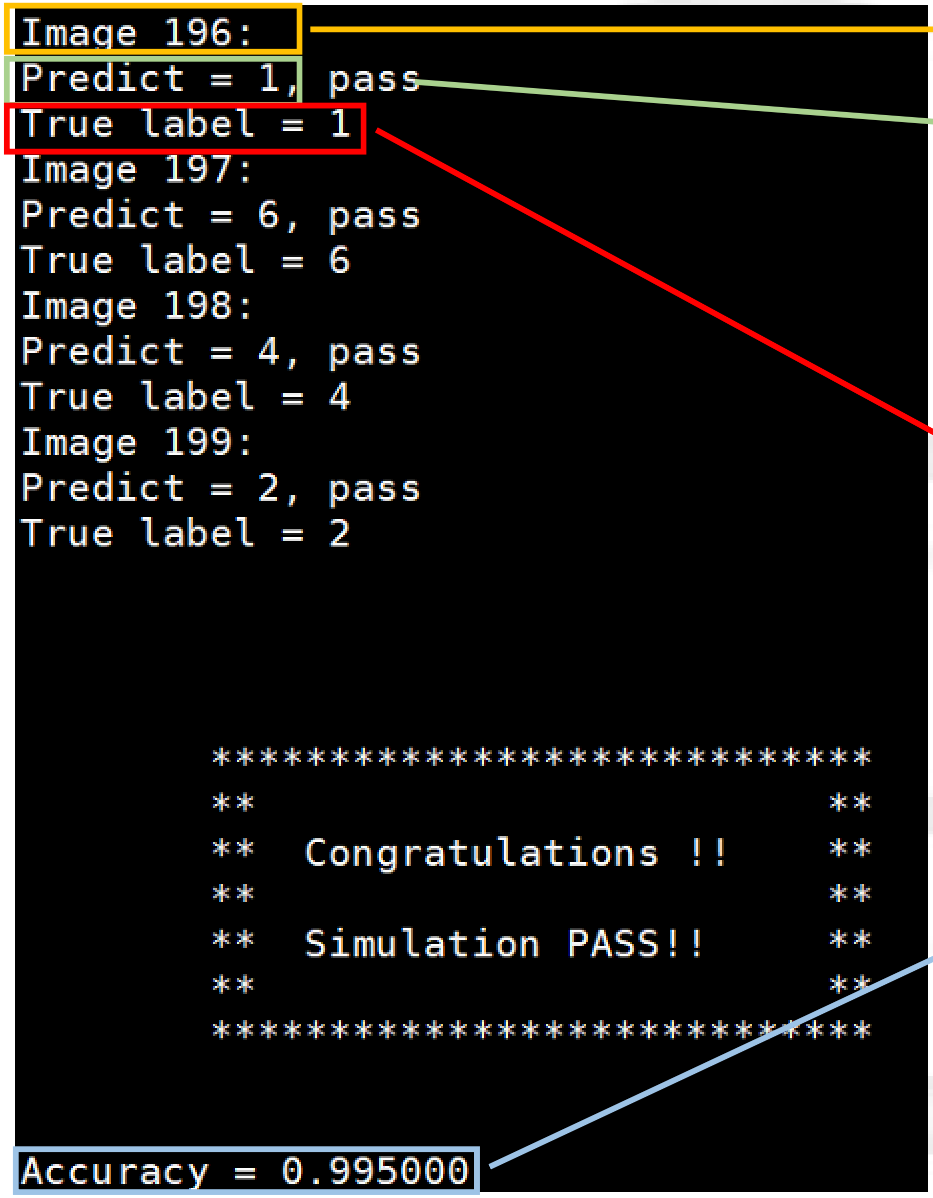


Results Demo

Results Demo (1/2)

- □ In “./images”, there are digit images that you can take a look at it.
- □ In “./data”, there are the content of images in .txt format.
- □ In “./parameters”, there are weights and bias in .txt format for CNN layers.
- □ In “./golden”, there are four files:
  - ➔ conv0.txt: the golden data of convolution layer 0 for image 0.
  - ➔ pool1.txt: the golden data of pooling layer 1 for image 0.
  - ➔ prediction.txt: the golden data of predictions that neural network predicts.
  - ➔ true_labels.txt: the golden data of labels that actually the images belong to.

Results Demo (2/2)



```
Image 196:
Predict = 1, pass
True label = 1
Image 197:
Predict = 6, pass
True label = 6
Image 198:
Predict = 4, pass
True label = 4
Image 199:
Predict = 2, pass
True label = 2



        ****************************
        **                        **
        **   Congratulations !!   **
        **                        **
        **   Simulation PASS!!    **
        **                        **
        ****************************



Accuracy = 0.995000
```

代表第幾張模擬圖片

Predict代表硬體跑完神經網路的預測結果，需與助教提供的prediction.txt一致

True label代表圖片真正對應的label

200張圖片經神經網路辨識完後所計算出的準確率

Homework Requirements

Homework Requirements (1/4)

- Two people for each group.(optional)
- Deadline: 5/3 (Sun.) 23:50
- Demo time: 5/4(Mon.) - 5/8(Fri.)
- Requirements:
  - Finish a CNN processing system based on Lab7's architecture.
  - Your design should be synthesized.
- Remember to fill out the Demo timetable on moodle before 4/26 (Sun.) 23:50. So we can make sure that each group's demo time will not conflict.

Homework Requirements (2/4)

- Grading
  - Pre-sim (50%)
    - Pass Conv0 layer for image 0 (10%)
    - Pass Pooling1 layer for image 0 (10%)
    - Pass all layers for 200 images (15%)
    - Pass all layers for hidden testbench (15%)
  - Post-sim (30%)
    - Pass all layers for 10 images (15%)
    - Pass all layers for hidden testbench (15%)
  - Report (10%)
  - Demo (10%)

Make sure your design can run under SoC Lab's environment !

Homework Requirements (3/4)

- Report
  - Don’t paste your code in the .docx file.
  - 如果兩個人一組，請寫出貢獻度。
    - Ex: A(E240XXXXX) 50% , B(E240XXXXX) 50%
  - 盡量在報告裡描述設計想法

Homework Requirements (4/4)

- Modules
  - ROM.v
  - TwoPort_SRAM.v
  - MUX2to1_16b.v
  - MUX2to1_32b.v
  - MUX4to1_32b.v
  - PE.v
  - Adder.v
  - ReLU.v
  - Truncate.v
  - Pooling.v
  - Decoder.v
  - Controller.v
  - top.v
  - top_tb.v

1. 紅色標示的檔案助教已經定義好，請勿更動裡面的內容(除了top_tb.v可視自己模擬的需求更改clock time及cycle數)
2. 所有modules皆勿更動inputs、outputs的宣告
3. File hierarchy 請參考Word檔

Appendix

Appendix (1/2)

- Simulation commands

Command	Description
make rtl_conv0 FSDB=1	Run RTL simulation of convolution layer 0 for the image 0.
make rtl_pool1 FSDB=1	Run RTL simulation of pooling layer 1 for the image 0.
make rtl_full FSDB=1	Run RTL simulation of the CNN for 200 images.
make syn_full FSDB=1	Run post-synthesis simulation of the CNN for 10 images.

FSDB=1 means dump the waveform

The waveform file will be in “build” directory

Appendix (2/2)

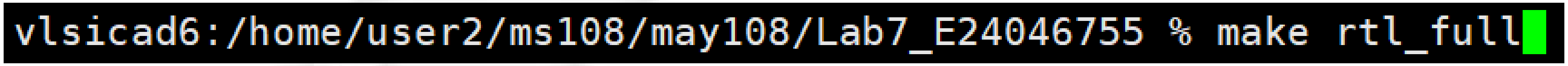
- Run utilities commands

Command	Description
make nWave	Open nWave without file pollution
make superlint	Open Superlint without file pollution
make synthesize	Synthesize RTL code without file pollution (You can change clock period in ./script/DC.sdc)
make tar	Compress homework to tar format
make clean	Delete “build” dictory (built files for simulation, synthesis, or verification)

請在你學號那一層的資料夾下執行這些指令（有makefile的那一層）

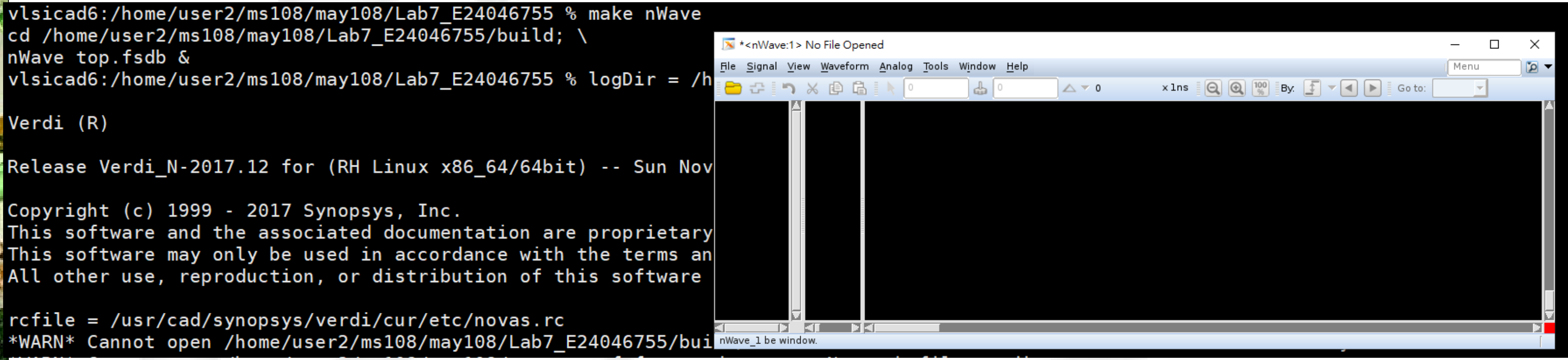
懶人包指令操作說明

- make rtl_full



```
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % make rtl_full
```

- make nWave



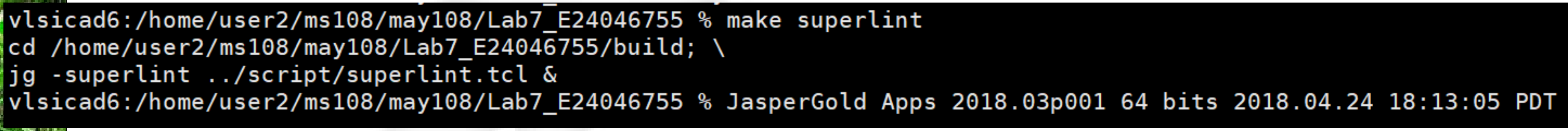
```
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % make nWave
cd /home/user2/ms108/may108/Lab7_E24046755/build; \
nWave top.fsdb &
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % logDir = /h

Verdi (R)

Release Verdi_N-2017.12 for (RH Linux x86_64/64bit) -- Sun Nov

Copyright (c) 1999 - 2017 Synopsys, Inc.
This software and the associated documentation are proprietary
This software may only be used in accordance with the terms an
All other use, reproduction, or distribution of this software

rcfile = /usr/cad/synopsys/verdi/cur/etc/novas.rc
*WARN* Cannot open /home/user2/ms108/may108/Lab7_E24046755/bui
```

*<nWave:1> No File Opened

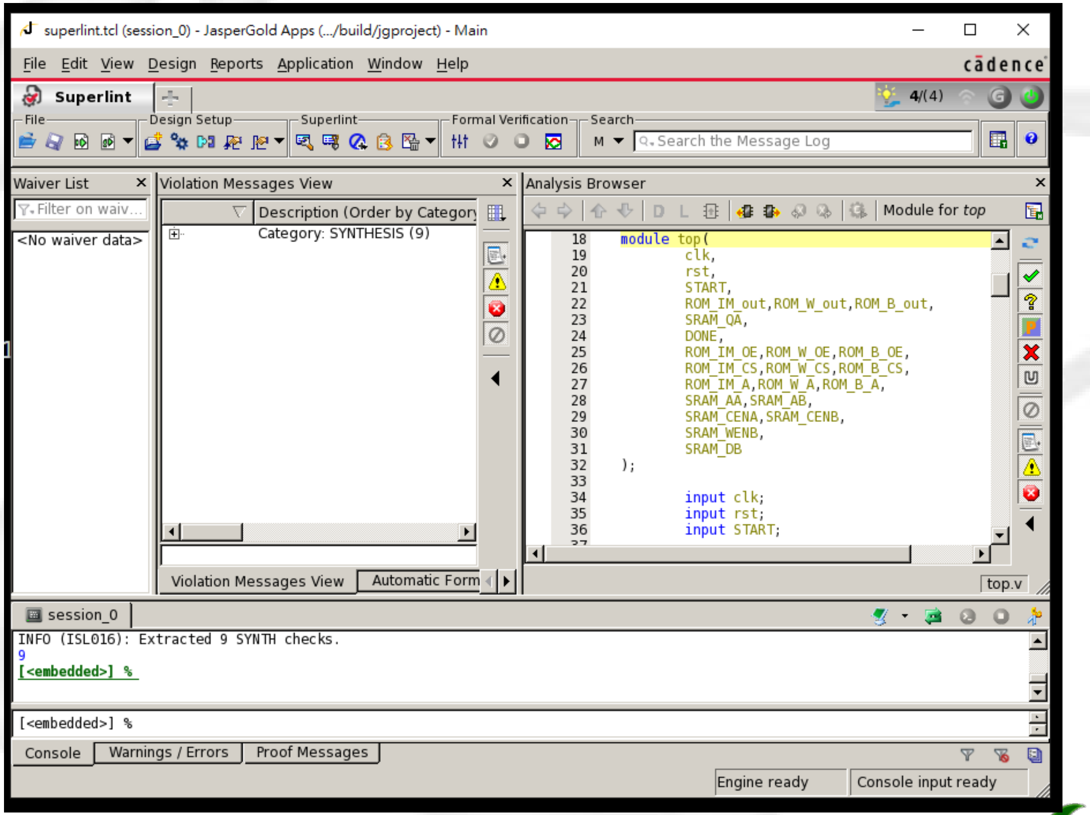
nWave_1 be window.

懶人包指令操作說明

- make superlint



```
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % make superlint
cd /home/user2/ms108/may108/Lab7_E24046755/build; \
jg -superlint ../script/superlint.tcl &
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % JasperGold Apps 2018.03p001 64 bits 2018.04.24 18:13:05 PDT
```



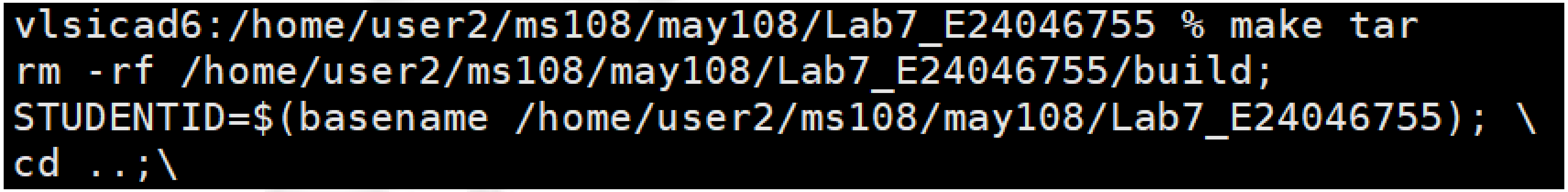
```
module top(
        clk,
        rst,
        START,
        ROM_IM_out,ROM_W_out,ROM_B_out,
        SRAM_QA,
        DONE,
        ROM_IM_OE,ROM_W_OE,ROM_B_OE,
        ROM_IM_CS,ROM_W_CS,ROM_B_CS,
        ROM_IM_A,ROM_W_A,ROM_B_A,
        SRAM_AA,SRAM_AB,
        SRAM_CENA,SRAM_CENB,
        SRAM_WENB,
        SRAM_DB
);

        input clk;
        input rst;
        input START;
```

```
INFO (ISL016): Extracted 9 SYNTH checks.
9
[<embedded>] %
```

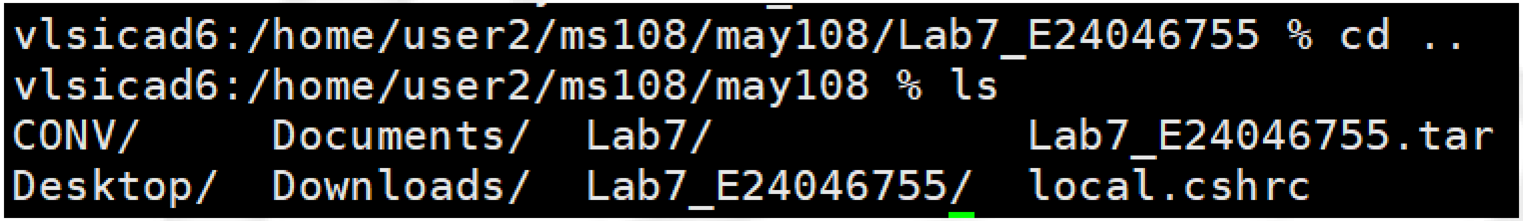
```
[<embedded>] %
```

懶人包指令操作說明

- make tar



```
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % make tar
rm -rf /home/user2/ms108/may108/Lab7_E24046755/build;
STUDENTID=$(basename /home/user2/ms108/may108/Lab7_E24046755); \
cd ..;\
```



```
vlsicad6:/home/user2/ms108/may108/Lab7_E24046755 % cd ..
vlsicad6:/home/user2/ms108/may108 % ls
CONV/     Documents/  Lab7/             Lab7_E24046755.tar
Desktop/  Downloads/  Lab7_E24046755/   local.cshrc
```