**National Cheng Kung University**

**Department of Electrical Engineering**

***Introduction to VLSI CAD (Spring 2020)***

**Lab Session 7**

**Design of CNN Processing System**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | | |
| 陳映余 | H54071156 | | |
| Practical | | Points | Marks |
| Part 1 | | 10 |  |
| Part 2 | | 10 |  |
| Part 3 | | 15 |  |
| Part 4 | | 15 |  |
| Part 5  (Word整體報告內容詳細程度) | | 10 |  |
| Notes | |  |  |
|  | | | |

**Due: 23:50 May 3, 2020@ moodle**

**Deliverables**

1. All Verilog codes including testbenche should be uploaded.

NOTE: Please **DO NOT** include source code in the paper report!

1. All homework requirements should be uploaded in this file hierarchy.
2. Please **DO NOT** upload waveforms (.fsdb or .vcd)!
3. **If you upload a dead body which we can’t even compile, you will get NO credit!**
4. **All Verilog file should get at least 90% SuperLint Coverage.**
5. All homework requirements should be uploaded in this file hierarchy or you will not get full credit.

**File Hierarchy**

* Lab7\_E240XXXXX\_E240XXXXX.tar(Lab7\_E240XXXXX.tar) (Don’t add other text in file name)
  + Lab7\_E240XXXXX\_E240XXXXX (Main folder of this project)
    - Lab7\_E240XXXXX\_E240XXXXX.docx (Your homework report)
    - Makefile (You shouldn’t modify it)
* data (Images data in .txt format)
* test\_im\*.txt
* golden (Golden hexadecimal data)
* \*.txt
* images (Hand-written digit images)
* \*.png
* Include (RTL parameters definition file)
* def.v
* parameters (The weights and bias of the neural network)
  + layer\*\_w.txt
  + layer\*\_b.txt
* script (Any scripts of verification and synthesis)
  + - * Script files (DC.sdc, \*.tcl)
* sim
* top\_tb.v
* tsmc13\_neg.v
  + - * ROM (ROM behavior model)
        + ROM.v
        + ROM\_tb.v
      * TwoPortSRAM (Two-Port SRAM behavior model)
        + TwoPortSRAM.v
        + TwoPortSRAM\_tb.v
* src (RTL code)
* Adder.v
* Controller.v
* Decoder.v
* MUX2to1\_16b.v
* MUX2to1\_32b.v
* MUX4to1\_32b.v
* PE.v
* Pooling.v
* Relu.v
* Truncate.v
* top.v
* syn (Synthesized code and timing file)
* top\_syn.v
* top\_syn.sdf

*Part1*

* Objective:

Build a CNN Processing System as the architecture of Fig.1. The system can do inference of CNN as Fig.2 and classify hand-written digit images.

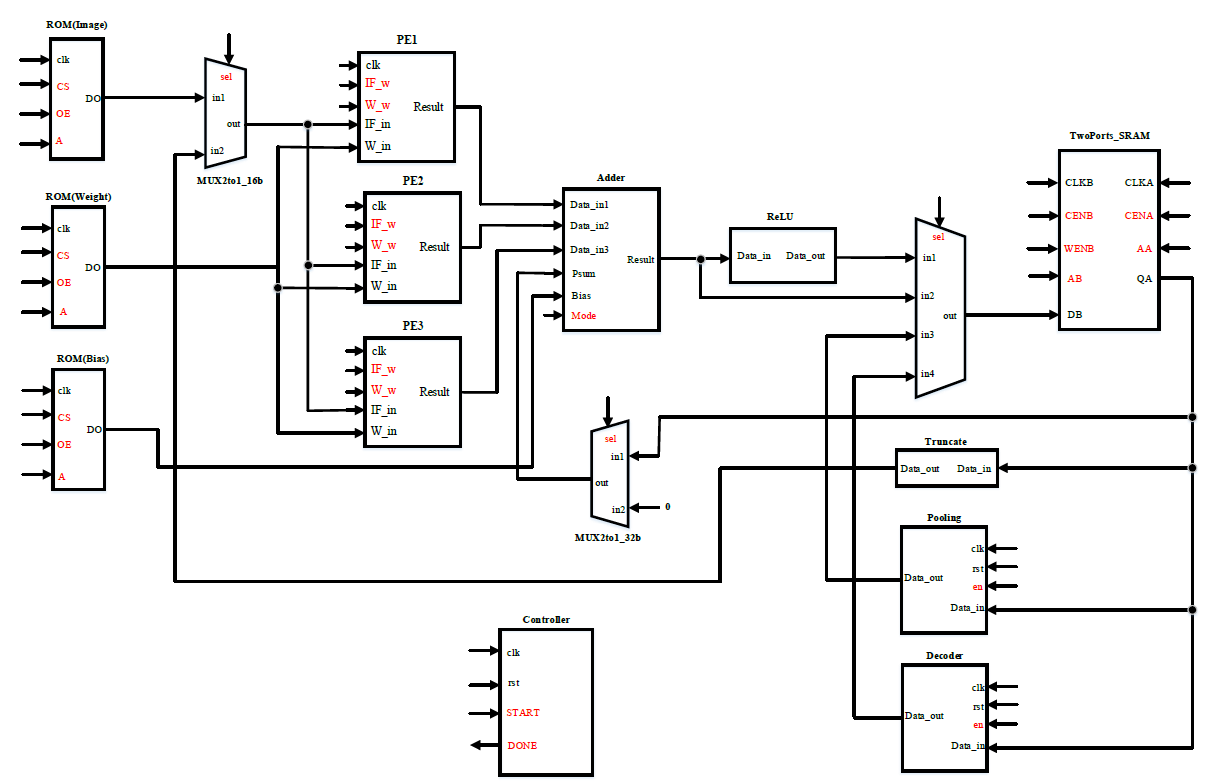


Fig.1 The architecture of CNN Processing System

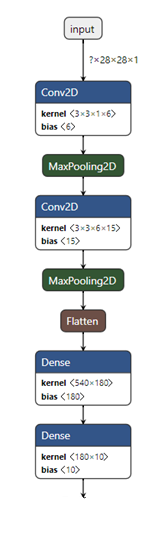


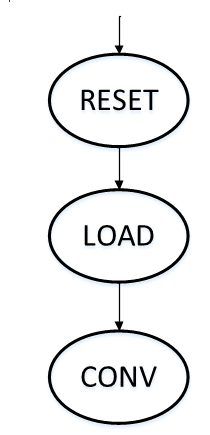
Fig.2 CNN NetworkArchitecture

* Please follow the tutorial of PowerPoint, and build a CNN Processing System by yourself. Run the system and let it pass the convolution layer 0 (Conv0) result for the image 0.

Simulation command: make rtl\_conv0

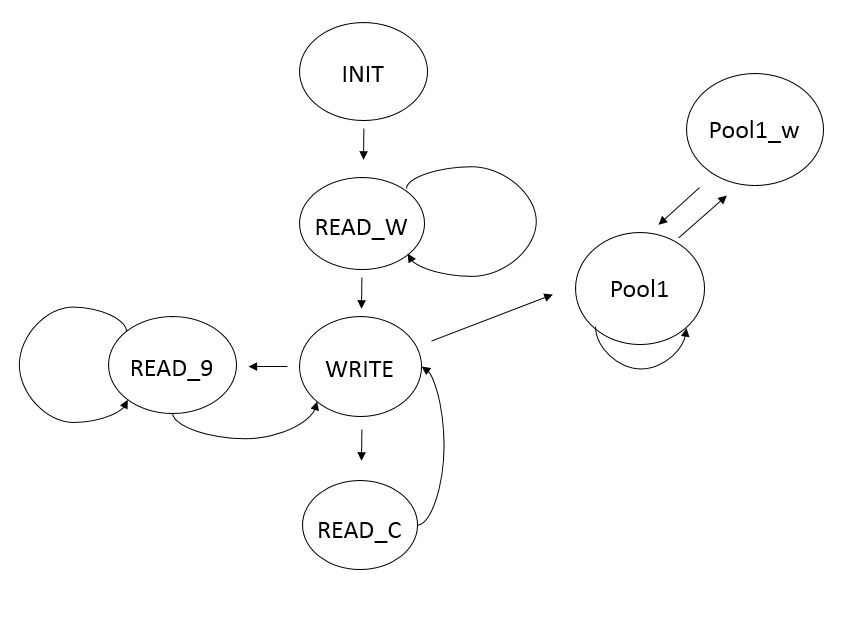
* Show your finite state machine (FSM) of the controller. Describe how your controller works.

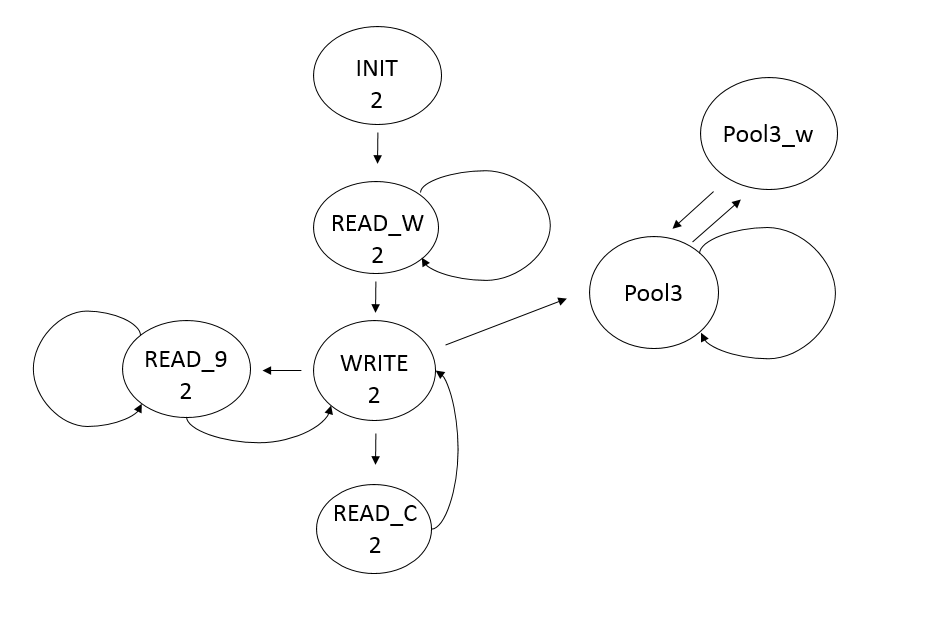
Note: Please use software tools to clearly draw the FSM.

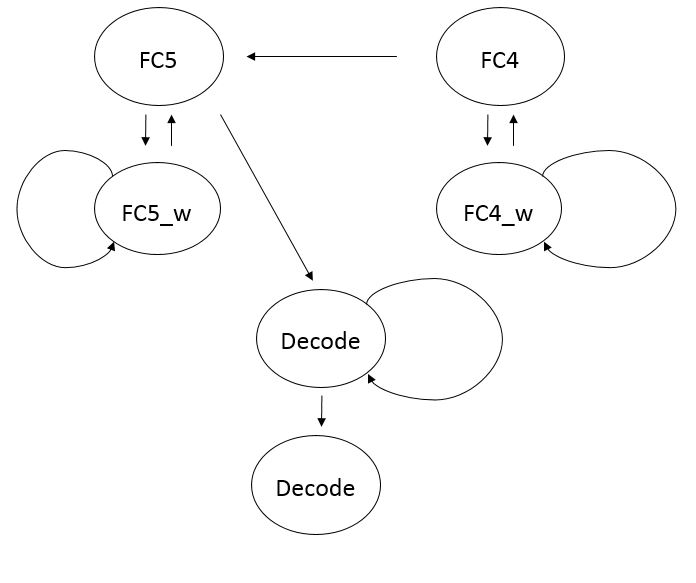
EX:

當Controller收到rst訊號時會進入RESET state，收到START訊號後會進入LOAD state。在LOAD state會去記憶體讀取convolution運算的資料(權重、partial sum、input feature map)，讀取完資料後進入CONV state做convolution運算。

|  |
| --- |
| Your FSM Design |
| 在下一頁，解釋在圖下面一頁。 |









Conv0:

一開始會在INIT state初始所有的暫存器，接著來到Read\_w，從ROM寫入9個feature與接下要用到的9個weight，再來寫入SRAM裡面，再到Read\_c寫入三個數字，再回到Write，直到一整個column寫滿再到Read9寫入9個feature，重複這樣的動作直到將一張圖的input feature map 全部寫入為止，再到下一階段Pool1。

由於要加上6個不同bias，因此完成讀入一張圖後，要重新回到INIT去重新讀入feature map與寫入新的bias。

Pool1:

當寫完第一張圖並做完convolution後，從SRAM裡面將值讀出來做pooling運算，也就是一次取四個值選取最大的那個重新讀入SRAM裡面，由於要讀4次，因此Pool1會多待幾個cycle，再到Pool1\_w作寫入，再回Pool1取值一直到將Conv0所存的所有值取出運算後，再到下一階段Conv2的INIT2。

Conv2: 一開始會在INIT2 state初始所有的暫存器，接著來到Read\_w2，從SRAM寫入9個feature與接下要用到的9個weight，再來寫入SRAM裡面，再到Read\_C2寫入三個數字，再回到Write，直到一整個column寫滿再到Read92寫入9個feature，重複這樣的動作直到將一張圖的input feature map 全部寫入為止，再到下一階段Pool3。

這裡比較不一樣的是一個是從ROM取值，一個是回SRAM取值，而一張圖也有許多filter要加，而Bias只能加一次，需要特別注意。

由於要加上15個不同的bias，因此完成讀入一張圖後，要重新回到INIT去重新讀入feature map與寫入新的bias。

Pool3:

這裡的行為與Pool1相同，從SRAM裡面將值讀出來做pooling運算，也就是一次取四個值選取最大的那個重新讀入SRAM裡面，由於要讀4次，因此Pool1會多待幾個cycle，再到Pool1\_w作寫入，再回Pool1取值一直到將Conv2所存的所有值取出運算後，再到下一個階段FC4。

FC4:

這裡要將Pool3裡面的值取出來並乘上相對應的weight，我的做法是一次取出9個值再寫入，因此再讀的時候會用到比較多的cycle，等寫完一次同個Bias之後，再回到FC4做同樣的動作直到取到最後一個Bias，再到FC5。

FC5:

這裡的做法與FC4相同，不同的是一個是從Pool3所存，一個是從FC4所存，不過做的動作一樣，直到取到最後一個Bias為，再到Decode。

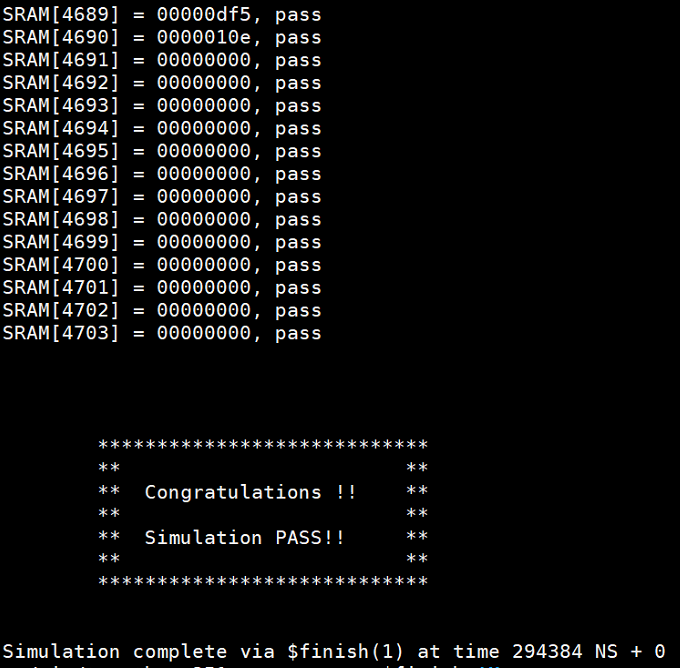
Decode:

這裡就是將FC5最後存入SRAM的10個值挑出最大的那個，並取出其對應的索引值(index)，由這裡值可以判斷哪一個為最相關的數字，由於要取10個值因此需要做幾個cycle才能結束到DONE結束整個流程。

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

EX:



|  |
| --- |
| Simulation Result |
|  |

*Part2*

* Run the system and pass the pooling layer 1 (Pooling1) result of the image 0.

Simulation command: make rtl\_pool1

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part3*

* Run the RTL code. Let the CNN Processing System classify 200 images.

Simulation command: make rtl\_full

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part4*

* Run the post-synthesis simulation. Let the CNN Processing System classify 10 images.

Simulation command: make syn\_full

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part5*

* Show your Superlint coverage

Note: Use the following command to get the lines of your code.

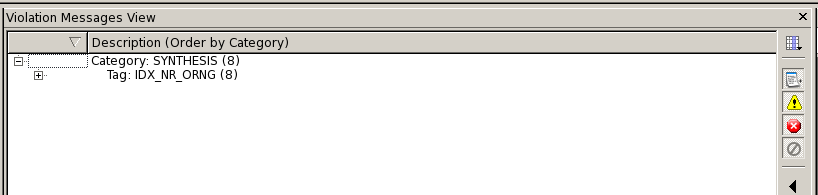
Command: wc –l ./src/\*

EX:

Number of lines of RTL codes:

Total lines = 1458

Number of warnings or errors on Superlint: 8



Coverage:

Coverage = 99.45%

|  |
| --- |
| Superlint Coverage |
| Number of lines of RTL codes:  Total lines = 1822  Number of warnings or errors on Superlint: 6    Coverage:  Coverage = 99.67% |

* Lesson Learned

這次的小專題內容比較大，耗費了三周才能完成，我認為自己完成比較能夠理解所有架構，因為每個人想法、風格不同，因此最好是討論彼此的架構，然後自己實際去做coding。

* Contribution

EX: A 50 %, B 50 %

A:100%。

*Appendix*

1. Simulation Requirements

You should make sure that your code can be simulated with specified commands in Table A-1. TA will use the same commands to check your design under SoC Lab environment. If your code can’t be recompiled by TA, you will get no credit.

TA will also see how many problems do you finish to decide which command TA will run. For example, if you only finish convolution layer 0, TA will run “make rtl\_conv0”. However, if you finish all layers, then TA will run “make rtl\_full”.

Table A-1: Simulation Commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| make rtl\_conv0 | Run RTL simulation of convolution layer 0 for the image 0. |
| make rtl\_pool1 | Run RTL simulation of pooling layer 1 for the image 0. |
| make rtl\_full | Run RTL simulation of the CNN for 200 images. |
| make syn\_full | Run post-synthesis simulation of the CNN for 10 images. |