**National Cheng Kung University**

**Department of Electrical Engineering**

***Introduction to VLSI CAD (Spring 2021)***

Lab Session 5

**Design of Finite State Machines and Memories**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | | |
| 陳旭祺 | E24099059 | | |
| Practical | | Points | Marks |
| Prob A | | 10 |  |
| Prob B | | 10 |  |
| Prob C | | 10 |  |
| Prob D | | 20 |  |
| Prob E | | 30 |  |
| Report | | 20 |  |
| Notes | |  |  |
|  | | | |

**Due: 15:00 April 14, 2021@ Moodle**

**Deliverables**

1. All Verilog codes including testbenches for each problem should be uploaded.

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

1. Noted! TA will use commands in Appendix A to check your design in SoC Lab. If TA cannot compile your code with the commands, you will not get full credit.
2. **If you upload a dead body which we cannot even compile, you will get NO credit!**
3. **All Verilog files should get at least 90% SuperLint Coverage.**
4. All homework requirements should be uploaded in this file hierarchy or you will not get full credit.

NOTE: Please **DO NOT** upload waveforms!

1. The name of uploaded file should be Lab5\_StudentID.tar, and the student ID must be uppercase. (Ex. Lab5\_E24081234.tar)

**Lab5\_studentID.tar**

**Lab5\_studentID**

**studentID.doc**

**ProbB**

**ProbD**

**mealy.v**

**mealy\_tb.v**

**ProbA**

**moore.v**

**moore\_tb.v**

**ProbC**

**mac.v**

**RAM.v**

**mac\_tb.v**

**RAM\_tb.v**

**mac\_syn.v**

**ROM.v**

**mac\_syn.sdf**

**ROM\_tb.v**

**ROM\_data.dat**

**ProbE**

**controller.v**

**input\_memory.v**

**grayscale.v**

**picture1.bmp**

**Picture2.bmp**

**output\_memory.v**

**top.v**

**top\_tb.v**

**top\_syn.v**

**Picture3.bmp**

**top\_syn.sdf**

Fig.1 File hierarchy for Homework submission

**Objectives:**

To learn how to design the finite state machine and the simple system.

Prob A: Moore Machine

1. Observe the given state diagram



|  |  |  |  |
| --- | --- | --- | --- |
| Current  State | Next State | | qout |
| din=0 | din=1 |
| S0=00 | S2 | S1 | 1 |
| S0=01 | S1 | S0 | 0 |
| S0=10 | S3 | S2 | 0 |
| S0=1 | S3 | S1 | 1 |

**Table1. Moore state table**

1. Write Verilog code for this FSM based on the **Moore Machine**. And you should name all your I/O ports according to the naming specification.

|  |  |  |
| --- | --- | --- |
| **Port name** | **input/output** | **# bit** |
| *rst* | input | 1 |
| *clk* | input | 1 |
| *din* | input | 1 |
| *qout* | output | 1 |

**Table2. I/O Naming Specification**

1. Compile the Verilog code to verify the operations of this module works properly.
2. Please attach your waveforms and specify your operations on the waveforms.

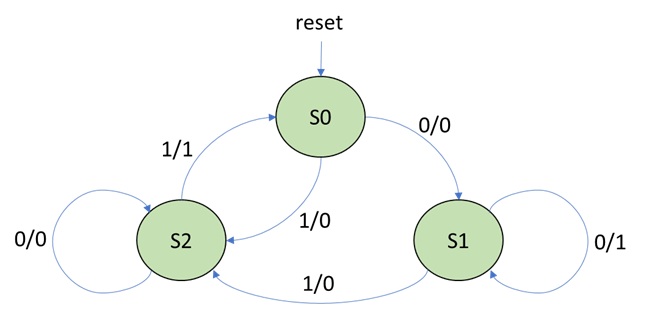
|  |
| --- |
| Simulation result on the terminal |
|  |
| Waveform |
|  |
| Explanation of waveform |
| 以下圖紅色為在clk的正緣觸發的狀態變化，共7個cycle   |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 2 |  | | 3 |  | 4 |  | | 5 |  | 6 |  | | 7 |  | 8 |  | |

1. Show SuperLint coverage (moore.v)

|  |
| --- |
|  |

Prob B: Mealy Machine

1. Observe the given state diagram



|  |  |  |
| --- | --- | --- |
| Current  State | Next State, output | |
| din=0 | din=1 |
| S0=00 | S1,0 | S2,0 |
| S1=01 | S1,1 | S2,0 |
| S2=11 | S2,0 | S0,1 |

**Table3. Construct the state table**

1. Write Verilog code for this FSM based on the **Mealy Machine**. You should name all your I/O ports according to the naming specification.

|  |  |  |
| --- | --- | --- |
| **Port name** | **input/output** | **# bit** |
| *rst* | input | 1 |
| *clk* | input | 1 |
| *din* | input | 1 |
| *qout* | output | 1 |

**Table4. I/O Naming Specification**

1. Compile the Verilog code and write a testbench to verify the operations of this module works properly.
2. Please attach your waveforms and specify your operations on the waveforms.

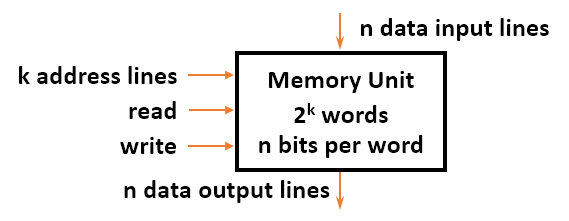
|  |
| --- |
| Simulation result on the terminal |
|  |
| Waveform |
|  |
| Explanation of waveform |
| 以下圖紅色為在clk的正緣觸發的狀態變化，共7個cycle   |  |  |  |  | | --- | --- | --- | --- | | 1 |  | 2 |  | | 3 |  | 4 |  | | 5 |  | 6 |  | | 7 |  | 8 |  | |

1. Show SuperLint coverage (mealy.v)

|  |
| --- |
|  |

Prob C: Memory

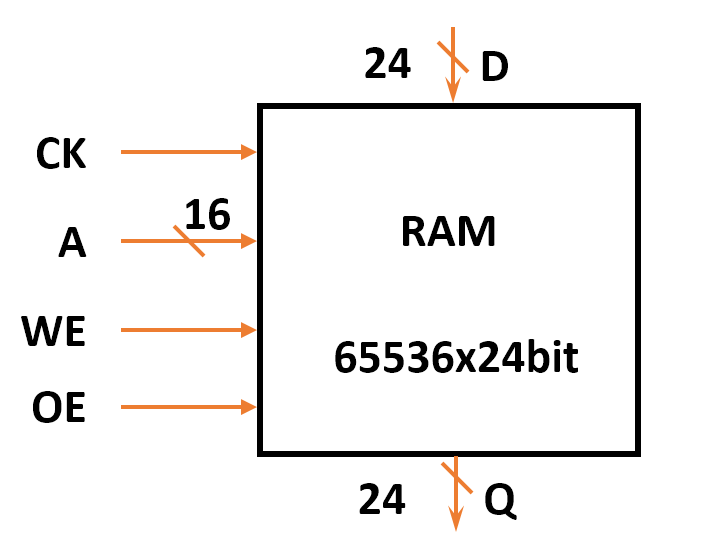
1. Understand the function of memory



Please fill the blanks :

1. Memory is a collection of binary storage cells together with associated circuits needed to transfer information
2. During readoperation, a specific word is selected by applying k-bit address and the selected word will be put on Data Output
3. During write operation, the input address selects the memory location to be written with the data appear at the Data Input
4. Design a 65536x24 bits random-access memory which consists of :

* Data depth = 65536, Data width = 24,
* Working at positive edge of clock
* When *WE* is high, *D* is stored into the memory as pointed by *A*. When *OE* is high, the data will be put on *Q* as pointed by *A*.



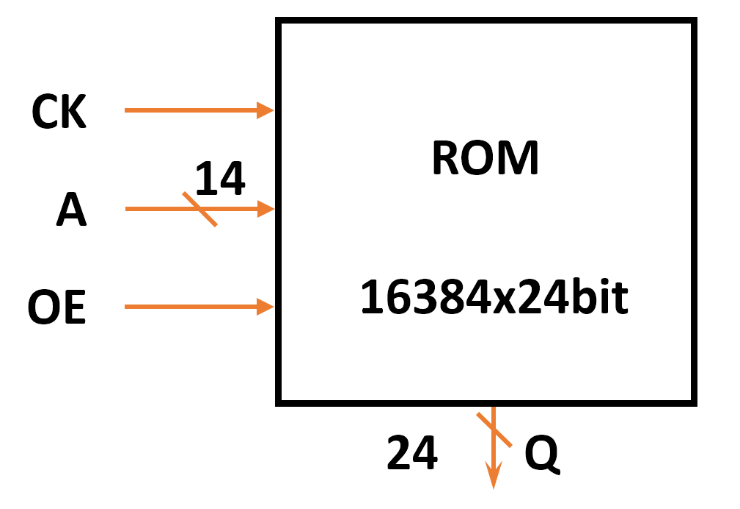
1. Compile the RAM code and write a testbench to observe the operation of this module using the following rules **(All data value is *hex* ):**
2. Write data (0x00\_0afc, 0x00\_b031, 0xf0\_0005, 0x00\_0246, 0x00\_00a9, 0x00\_006e, 0xff\_ffff, 0x00\_046a, 0xc1\_0dd3) into addr (0, 1887, 22453, 308, 74, 299, 51, 16388, 65535) (dec).
3. Read data from addr (70, 299, 0, 1888, 308, 51, 22453, 97) (dec).
4. Write data (0x00\_1fc3, 0xed\_2a24) into addr (74, 1888) (dec).
5. Read data from addr (16388) (dec).
6. Write data (0x00\_0123) into addr (65535) (dec).
7. Read data from addr (74) (dec).
8. Write data (0x00\_00cc) into addr (45294) (dec).
9. Read data from addr (65535, 45294) (dec).

1. Please attach your waveforms and specify your operations on the waveforms.

|  |
| --- |
| Simulation result on the terminal |
| 一張含有 桌 的圖片  自動產生的描述一張含有 桌 的圖片  自動產生的描述  一張含有 桌 的圖片  自動產生的描述 |
| Waveform |
|  |
| Explanation of waveform |
| 以address[0]驗證為例，在助教指定的測試A中write\_enable=1，故當clk一拉高寫入值0x00\_0afc到指定地址address[0]；測試B中read\_enable=1與write\_enable=0，故當clk一拉高讀出address[0]的值，觀察讀出值為0x00\_0afc代表值確實有寫入RAM裡面，而我另外用   |  | | --- | |  |   在終端機上print出來address 0到99的值，可觀察第一行RAM[0]=000afc1  一樣代表有寫入RAM裡面。 |

1. Design a 16384x24 bits read-only memory which consists of :

* Data depth = 16384, Data width = 24,
* Working at positive edge of clock
* When *OE* is high, the data will be put on *Q* as pointed by *A*.



1. Compile the ROM code and write a testbench based on the conditions to prove that the operations of this module work properly:

* A provided file named *ROM\_data.dat* and then load the *ROM\_data.dat* into ROM module. Please read data from A = 16368 to 16383.
* The data is equal to its address number. EX: (A,Data) = (0,0), (1,1), …, (16383,16383)

1. Please attach your waveforms and specify your operations on the waveforms.

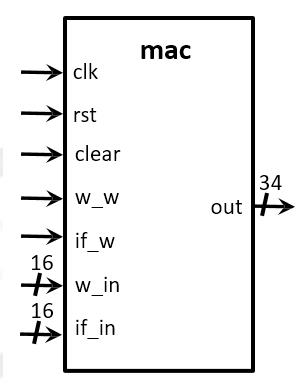
|  |
| --- |
| Simulation result on the terminal |
|  |
| Waveform |
|  |
| Explanation of waveform |
| 用Notrpad++打開ROM\_data.dat檔案可知data即為address 值，因此當clk拉高、正緣觸發時，data\_out值改變與address相同。 |

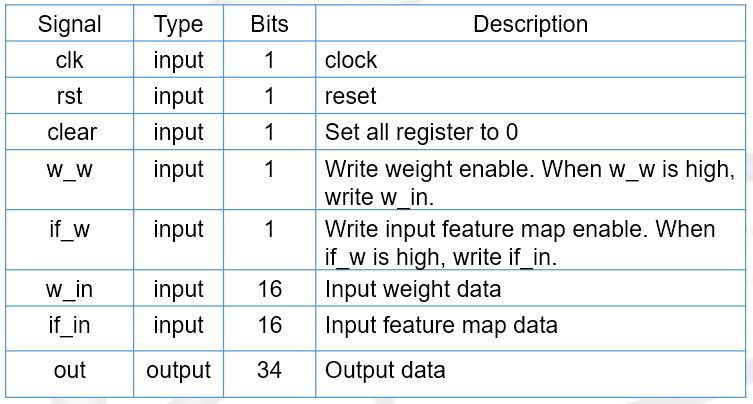
1. Show SuperLint coverage (RAM.v) (ROM.v)

|  |
| --- |
| SuperLint Coverage for RAM.v |
|  |
| SuperLint Coverage for ROM.v |
|  |

Prob D: Design a MAC using shift register

1. Design a MAC using shift register. The following is MAC specification.





1. **After synthesizing your design, you may have some information about the circuit. Please fill in the following form.**

|  |  |  |
| --- | --- | --- |
| Synthesis Report | | |
| 一張含有 文字 的圖片  自動產生的描述 | | |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
|  |  |  |

1. Please attach your waveforms.

|  |
| --- |
| Simulation result on the terminal |
|  |
| Waveform |
|  |
| Explanation of waveform |
| 一張含有 文字, 已掛上 的圖片  自動產生的描述與上次Lab4 ProbC大同小異，差別只減少input put，而改用邏輯設計課所學Serial-in to Parallel-out (SIPO) Shift Register，用non-blocking的寫法  實現 |
| SuperLint Coverage |
|  |
| Waveform after Synthesis |
|  |

Prob E: A Simple system

You are about to integrate all components that you have learned so far to form a simple system (*system*). The system will be able to change RGB pictures to grayscale pictures. The block diagram of system is as shown in Fig1. (Clock pin and reset pin is ignored in the graph, but you should implement it)

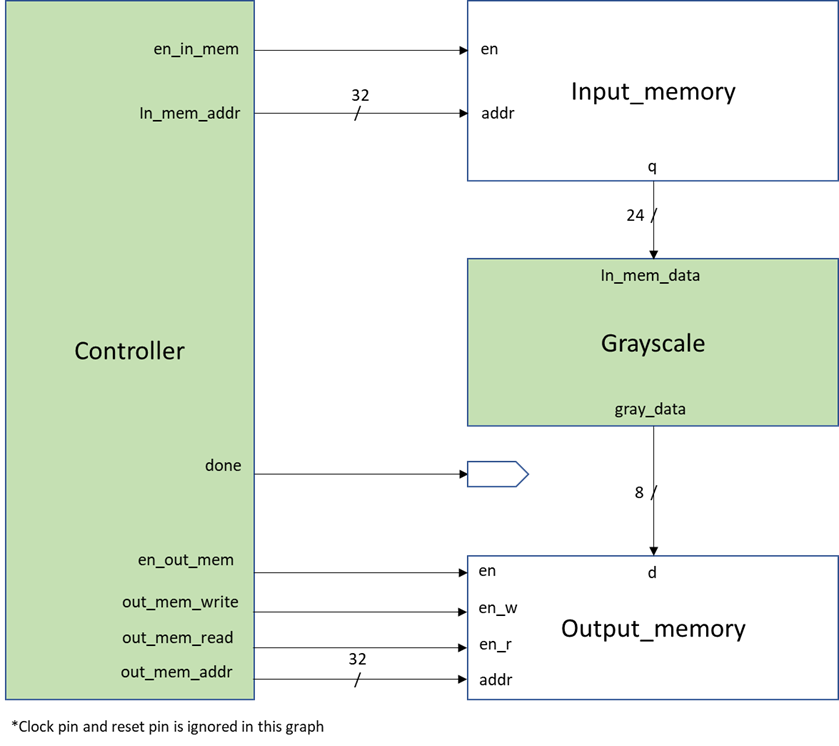


Fig1. The block diagram of system (external)

1. Understanding the function:

Once system is initialized, it

* 1. reads pixel from the input memory.
  2. compute new value of pixels
  3. writes the new value pixel back to the output memory.
  4. repeats the process step (a)-(c) until the last pixel of output memory is updated.
  5. flags “done” when step (d) is completed

1. Know the basic design rules
   * All operations initiated on the positive edge trigger of the clock
   * Control signals:
     + *en\_in\_mem*: enable input memory
     + *in\_mem\_addr*: input memory address
     + *en\_out\_mem*: enable output memory
     + *out\_mem\_read*: output memory read enable
     + *out\_mem\_write*: output memory write enable
     + *out\_mem\_addr*: output memory address
     + *done*: Stop the process
2. Draw your state diagram

|  |
| --- |
|  |

1. Please attach your screenshot of result.bmp.

|  |  |
| --- | --- |
| Original Image | Results |
|  |  |
|  |  |
|  |  |

1. Show SuperLint coverage (controller.v)

|  |  |  |
| --- | --- | --- |
| Simulation result on the terminal | | |
|  | | |
| Waveform | | |
|  | | |
| Explanation of waveform | | |
| 第一張圖為所有執行的波形圖，第二張為最一開始從，使in\_mem\_addr, out\_mem\_addr初始化從0開始加，en\_in\_mem與en\_out\_mem、out\_mem\_write隨clk交替拉高，進入讀入(S\_in\_mem)與讀出(S\_out\_mem)的狀態迴圈，一直到out\_addr到32'd479999時，就是把整張480000像素的圖片跑完就進入done =1卡在S\_done的單一狀態裡面，符合上面設計的state diagram的大致流程。 | | |
| SuperLint Coverage | | |
|  | | |
| Synthesis Report | | |
|  | | |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
|  |  |  |
| Waveform after Synthesis | | |
|  | | |

* Lessons learned from this lab

這次Lab5作業最大的難點在ProbE，由於Controller.v的input只有clk和rst，因此所有外接訊號都要自己生成而又要由自己生成的數值去設立state diagram 終止的條件，尤其是這兩個訊號in\_mem\_addr, out\_mem\_addr一直在想要怎麼產生，解決方法是**用clk來驅動，在跑完一輪in\_mem和out\_mem也就是我設定的2個cycle cycle後就會把address加一**，在第一個clk 拉起到第二個clk拉起中間組合電路的grayscale.v有充足的時間會做完，如此交替直到跑完480000像素做完。在搞懂上述觀念前也多次詢問助教，感謝助教能在百忙之中撥空指導，希望有一天也能具備像助教這樣coding的能力並搭配數位邏輯的概念。

※ NOTE:

Please compress all the following files into one compressed file (“.tar “ format) and submit through Moodle website:

|  |
| --- |
| Verilog codes, testbench and any other files to support your testbench |
| ProbA: moore.v, moore\_tb.v  ProbB: mealy.v, mealy\_tb.v  ProbC: RAM.v, RAM\_tb.v  ROM.v, ROM\_tb.v, ROM\_data.dat  ProbD: mac.v  mac\_syn.v  mac\_syn.sdf  mac\_tb.v  ProbE: controller.v  grayscale.v  input\_memory.v  output\_memory.v  top.v  top\_syn.v  top\_syn.sdf  top\_tb.v  picture1.bmp  picture2.bmp  picture3.bmp |

1. If there are other files used in your design, please attach the files too and make sure they’re properly included.
2. Simulation command

