# National Cheng Kung University

## Department of Electrical Engineering

# *Introduction to VLSI CAD (Spring 2024)*

## Lab Session 5

# FSM&Synthesis of Sequential Logic

| Name | Student ID | |
|---|---|---|
| 游宗謀 | E94106151 | |
| **Practical Sections** | **Points** | **Marks** |
| Lab in class | 15 | |
| Prob A | 20 | |
| Prob B | 10 | |
| Prob C | 15 | |
| Report | 35 | |
| File hierarchy, naming…etc. | 5 | |
| Notes: | | |

**Due Date: 14:59, April 4, 2024 @ moodle**

## Deliverables

1) All Verilog codes including testbenches for each problem should be uploaded.
   NOTE: Please **DO NOT** include source code in the paper report!
2) All homework requirements should follow the naming rule in this file hierarchy or you will not get the full credit.
   NOTE: Please **DO NOT** upload waveforms!
3) Important! TA will use the command in Appendix A to check your design under SoC Lab environment, if your code can not be recompiled by TA successfully using the commands, you will not get the full credit.
4) If you upload a dead body, which we cannot even compile, you will get **NO** credit!
5) All Verilog file before synthesizing should get at least 95% Superlint Coverage.
6) Lab5_Student_ID.tar (English alphabet of Student_ID should be **capital**.)

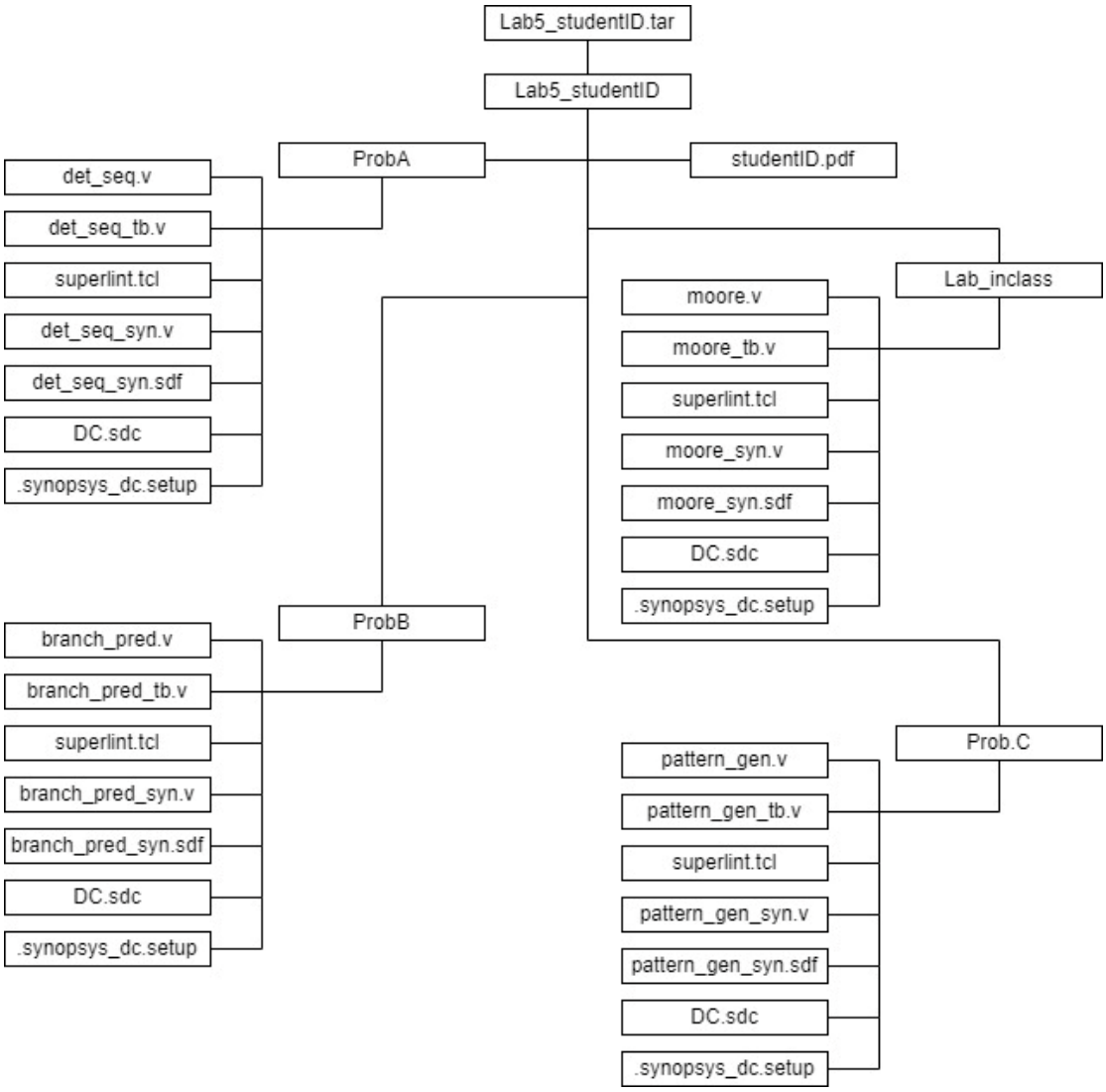
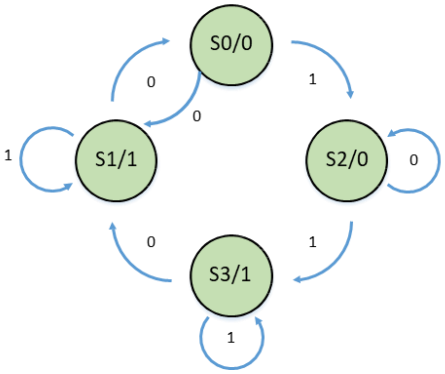



Fig.1 File hierarchy for Homework submission

1) **Design a Moore machine circuit that can be synthesized. The following is Moore machine module's specification. (Do NOT add or delete I/O ports, but you can change their behavior.)**




| Current State | Next State | | qout |
|---|---|---|---|
| | din=0 | din=1 | |
| S0=00 | S1 | S2 | 0 |
| S1=01 | S0 | S1 | 0 |
| S2=10 | S2 | S3 | 1 |
| S3=11 | S1 | S3 | 1 |

| Signal | Type | Bits | Description |
|---|---|---|---|
| clk | input | 1 | Clock pin. |
| rst | input | 1 | Reset pin. Reset all of the flip flops to zeros. |
| din | input | 1 | Control signal for fsm. |
| qout | output | 1 | 1:when current state==S1 or current state==S3<br>0: when current state==S0 or current state==S2 |

2) **Please describe your FSM in detail**

| Explanation about your FSM |
|---|
| This is a Moore machine that will decide its output based on its current state, and the changes of its next state will be depends on its input. So, as the plot and form above, there are four different cases below. First, if the current state is s0, it will output(qout) 0, and the next state will become s1 if the input(din) is 0 or s2 if din is 1. Second, if the current state is s1, it will output 0, and the next state will become s0 if din is 0 or s1 if din is 1. Third, if the current state is s2, it will output 1, and the next state will become s2 if din is 0 or s3 if din is 1. Last, if the current state is s3, it will output 1, and the next state will become s1 if din is 0 or s3 if din is 1. |

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

| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
|---|---|---|
| **0.32** | **4.147200 um$^2$** | **6.7439e-03 mW** |

### 4) Please attach your design waveforms.

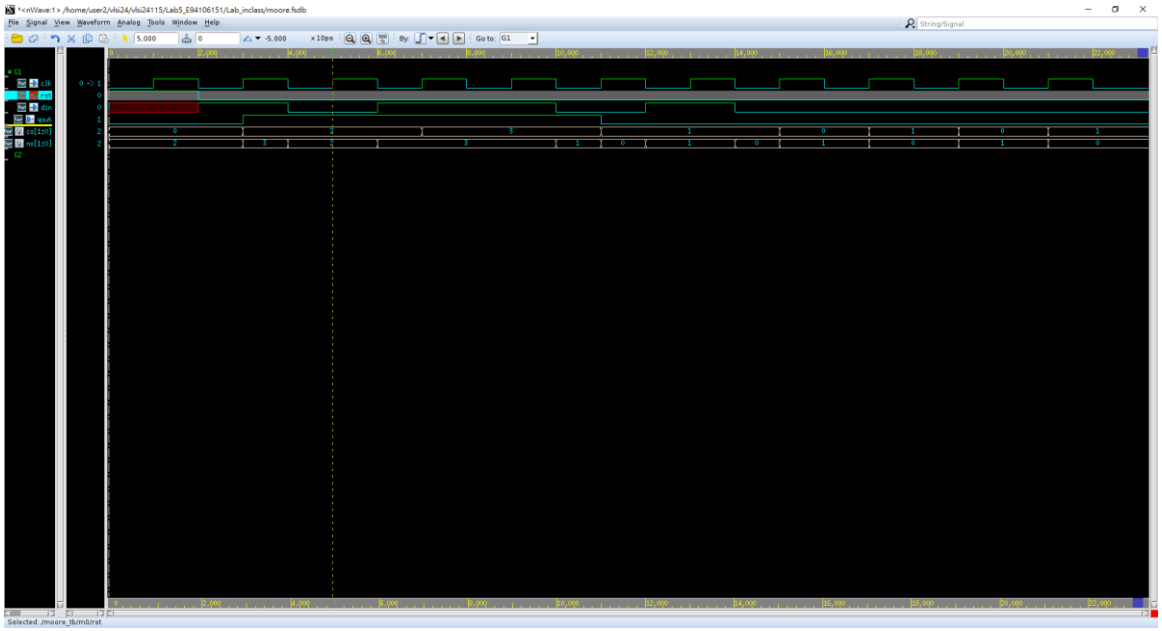
Your simulation result on the terminal.

RTL:

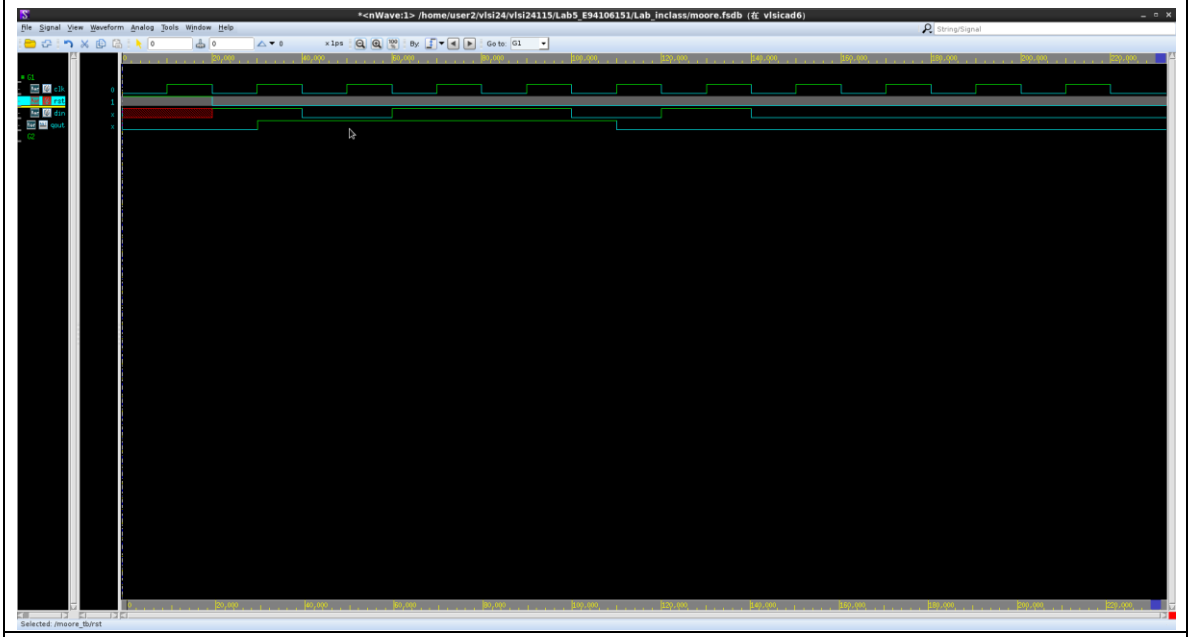

Gate:


Your waveform :

RTL:



Gate:

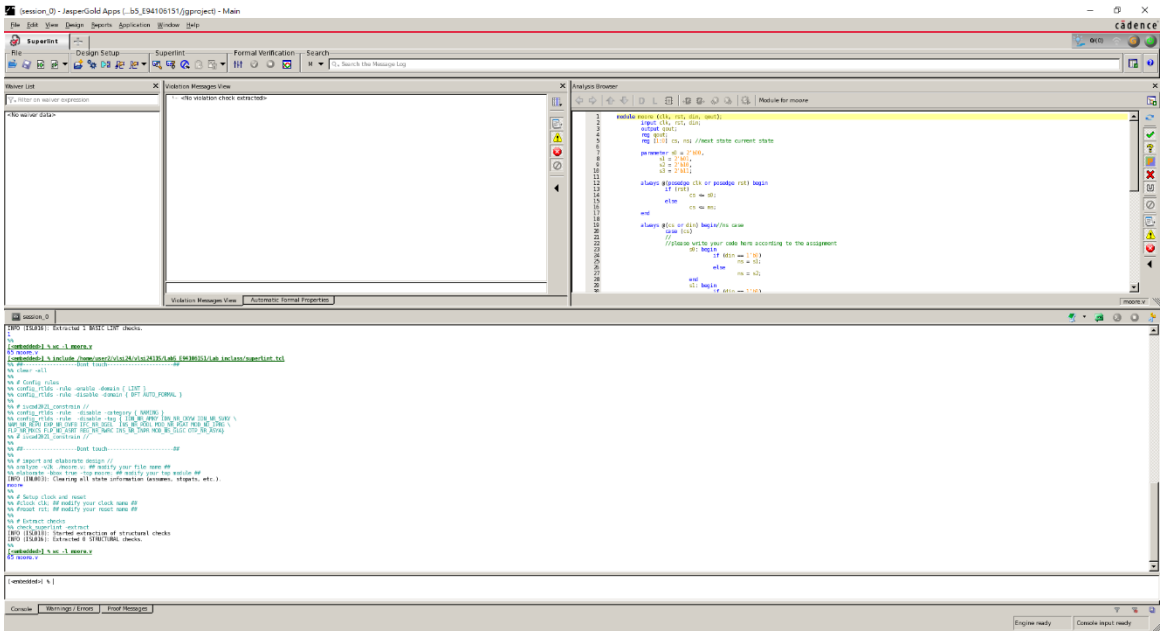


Explanation of your waveform :

以黃色標記指到的位置為例，前一個 clock cycle 的 state 為 s2(cs=2)，所以輸出 qout 為 1，而因為輸入 din 為 0，所以下一個 state 仍然是 s2(cs=2)，同理其他 clk 正緣也是跟這一樣可以用狀態機的圖表解釋。

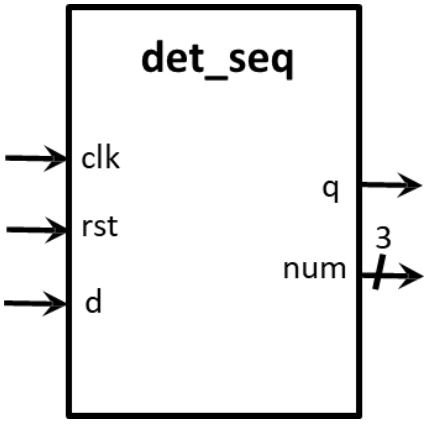
Superlint Coverage

Coverage = (1-(0/65))*100% = 100%



*ProbA: Design a circuit "detecting pattern 101011"*

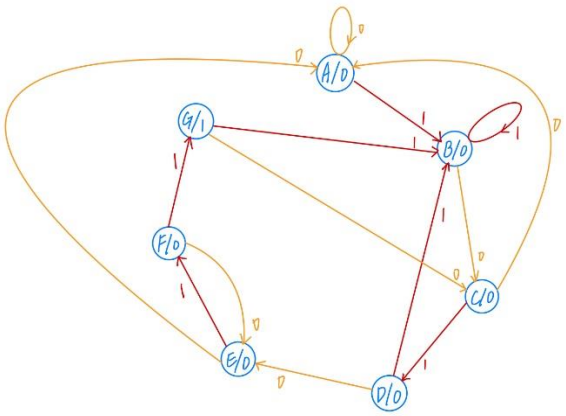
1) **Design a pattern seq-detecting circuit that can be synthesized with moore machine. The following is det_seq module's specification. (Do NOT add or delete I/O ports, but you can change their behavior.)**



| Signal | Type | Bits | Description |
|---|---|---|---|
| clk | input | 1 | clock |
| rst | input | 1 | reset, active high |
| d | input | 1 | pattern bit |
| q | output | 1 | When detect pattern 101011, q pulls to high at next clock posedge. Otherwise, q is low. |
| num | output | 3 | Count the number of pattern 101011 |

**2) Please describe your FSM in detail**

Explanation about your FSM




根據要偵測的 pattern: 101011，可以畫出上面的圖，表示出它輸入(d)、輸出(q)跟狀態之間的關係。總共有七個狀態，初始狀態為 A 當 d 一直改變時，狀態會根據上面的圖改變，直到狀態 G 就是代表偵測到 pattern 101011，就輸出(q)為 1 且 num 加 1。

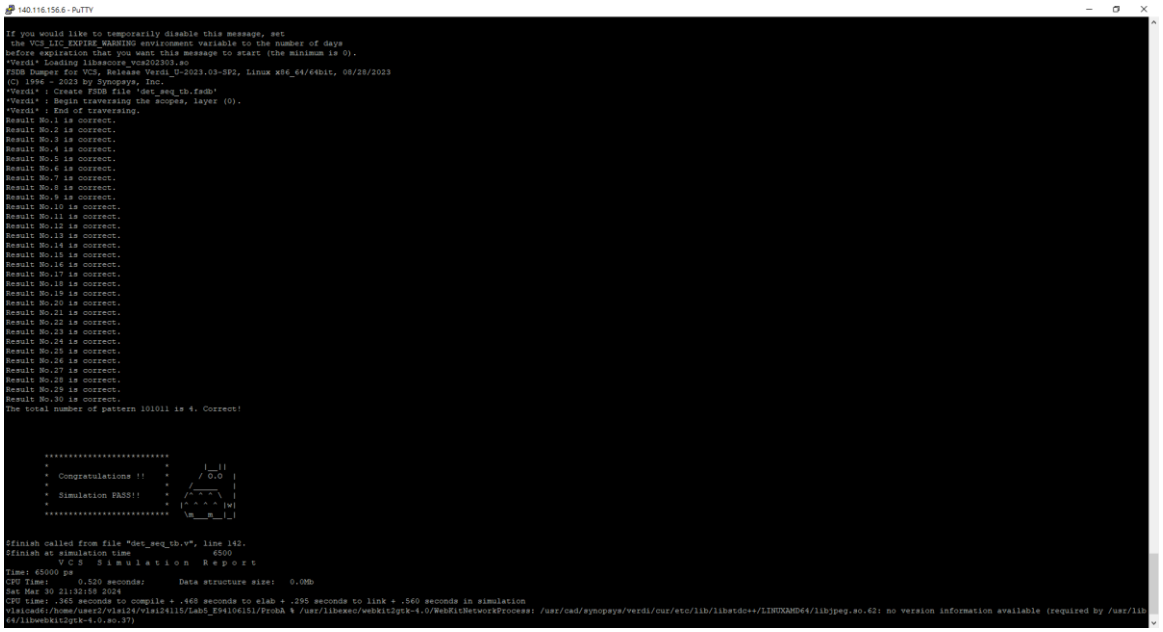
**3) After synthesizing your design, you may have some information about the circuit. Please fill in the following form.**

| Timing (slack) | Area (total cell area) | Power (total) |
|---|---|---|
| 0.30 | 12.286080 $um^2$ | 1.4213e-02 mW |

**4) Please attach your design waveforms.**

Your simulation result on the terminal.

RTL:

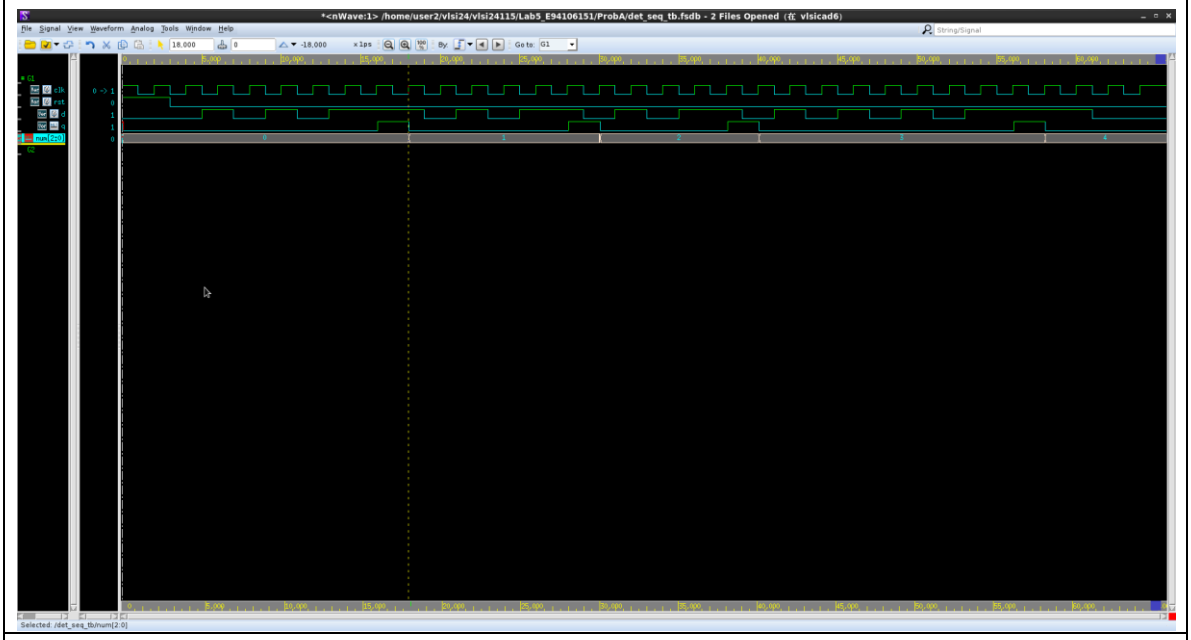



Gate:


Your waveform :

RTL:

Gate:

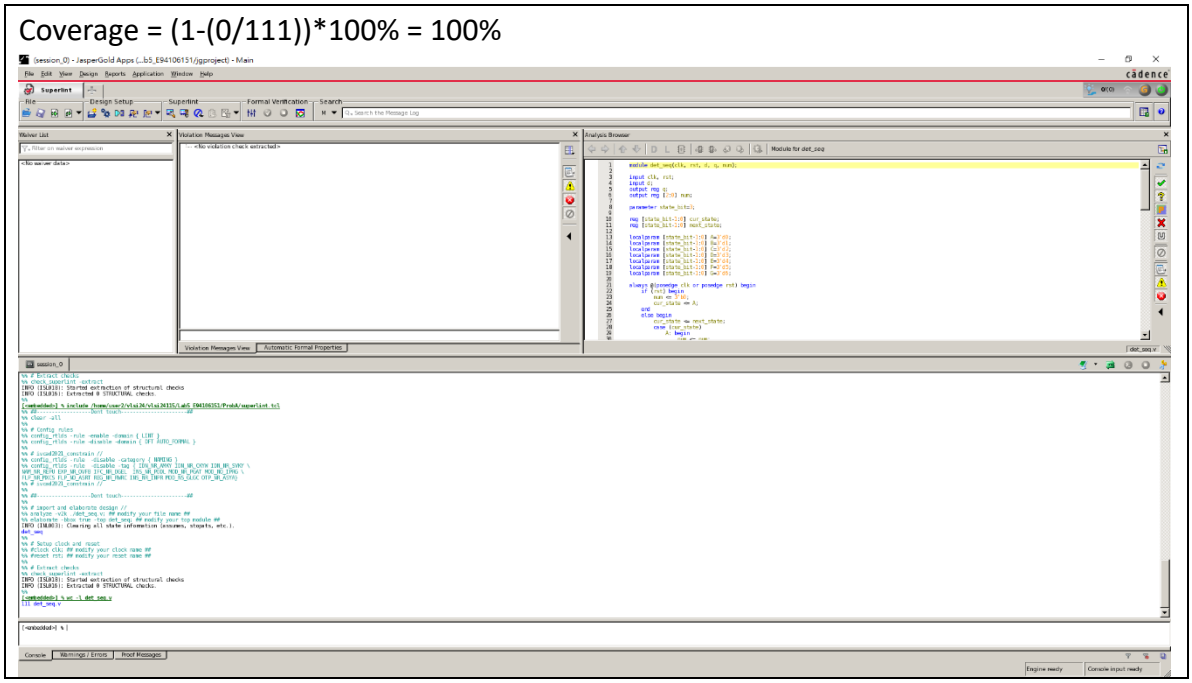


Explanation of your waveform :

以黃色標記指到的位置(RTL)為例，前一個 clock cycle 的 state 為 E(cur_state=4)，所以輸出 q 為 0、num 沒增加，而因為輸入 d 為 1，所以下一個 state 變成 F(cur_state=5)，同理其他 clk 正緣也是跟這一樣可以用狀態機的圖表解釋。

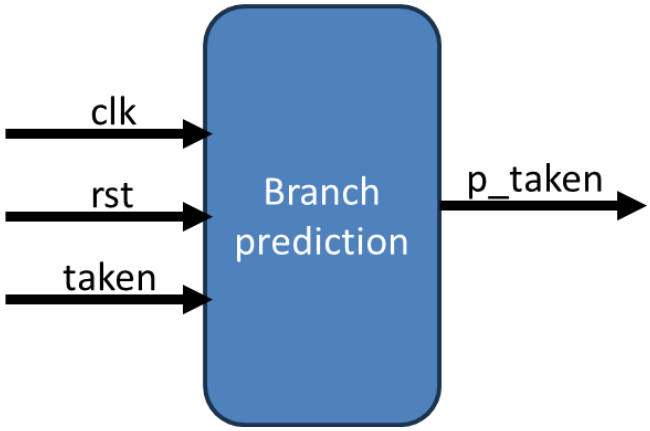
Superlint Coverage

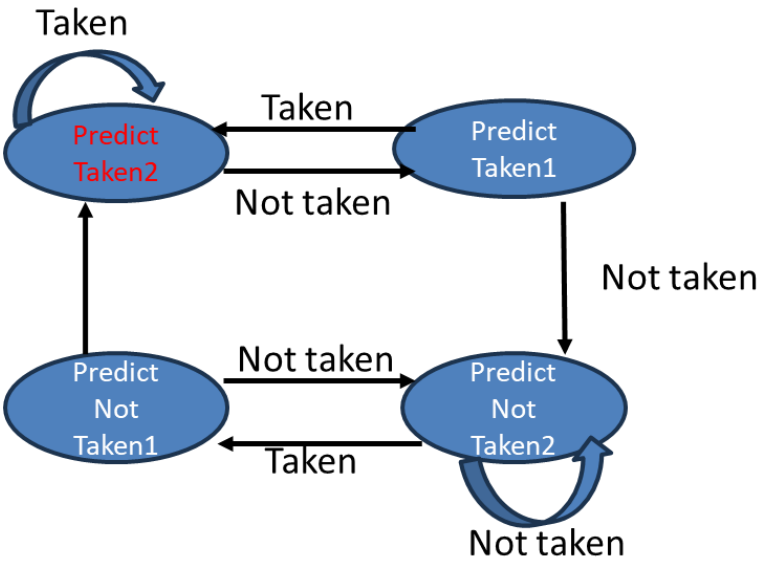
Coverage = (1-(0/111))*100% = 100%



*ProbB: Design a 2-bit branch prediction*

1) **Design a 2-bit branch prediction with moore machine. The following is 2-bit branch prediction module's specification. (Do NOT add or delete any I/O ports, but you can change their behavior.)**






**2) Please describe your FSM in detail.**

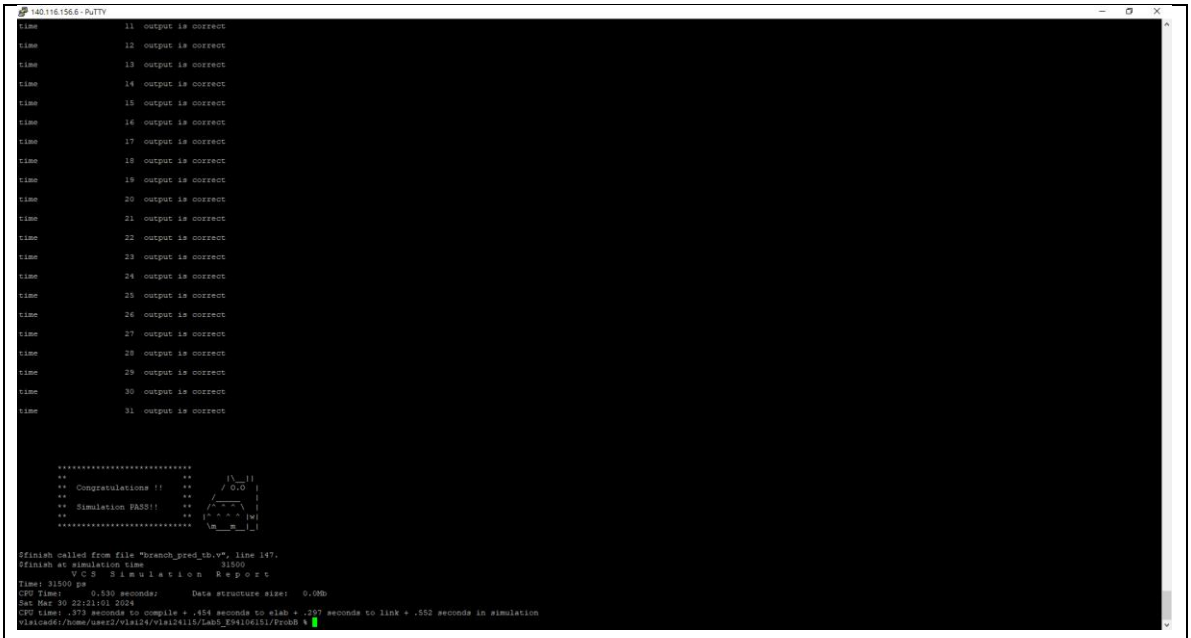
| Explanation about your FSM |
|---|
| 根據上圖我們可以設計四個 state 的 Moore machine。在 Predict Taken1 時，p_taken 為 1，若 taken 為 1 則下個 state 為 Predict Taken2、為 0 則下個 state 為 Predict Not Taken2。在 Predict Taken2 時，p_taken 為 1，若 taken 為 1 則下個 state 為 Predict Taken2、為 0 則下個 state 為 Predict Not Taken1。在 Predict Not Taken1 時，p_taken 為 0，若 taken 為 1 則下個 state 為 Predict Taken2、為 0 則下個 state 為 Predict Not Taken2。在 Predict Not Taken2 時，p_taken 為 0，若 taken 為 1 則下個 state 為 Predict Not Taken1、為 0 則下個 state 為 Predict Not Taken2。 |

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

| Timing (slack) | Area (total cell area) | Power (total) |
|---|---|---|
| **0.33** | **3.265920 $um^2$** | **5.4371e-03 mW** |

**4) Please attach your design waveforms.**

| Your simulation result on the terminal. |
|---|
| RTL: |

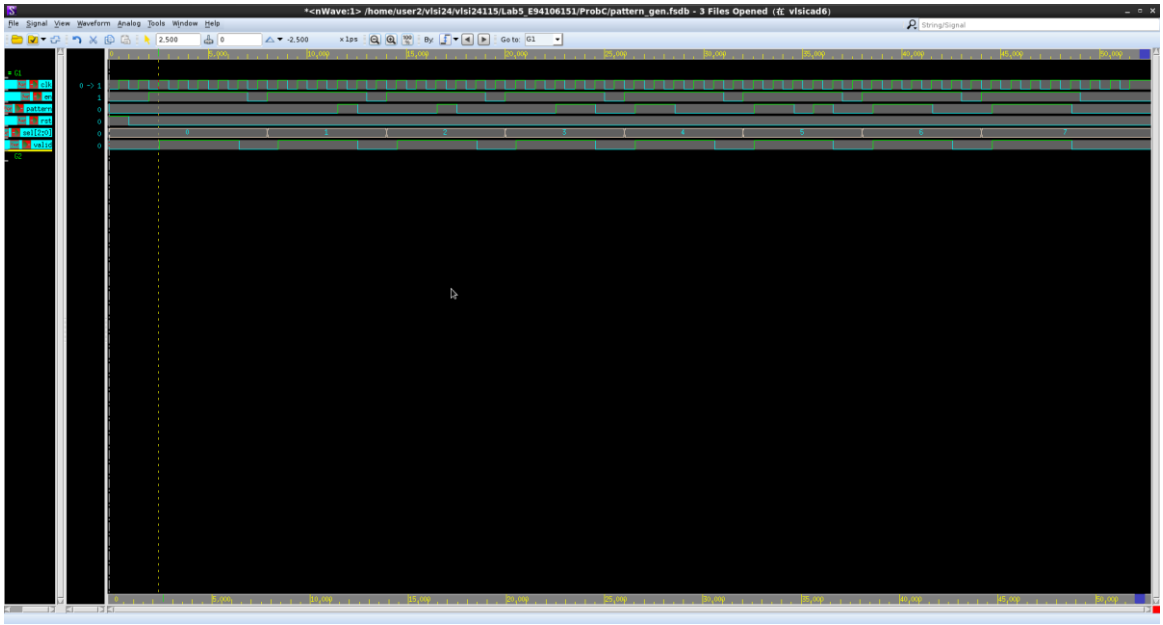



Gate:


Your waveform :

RTL:

Gate:

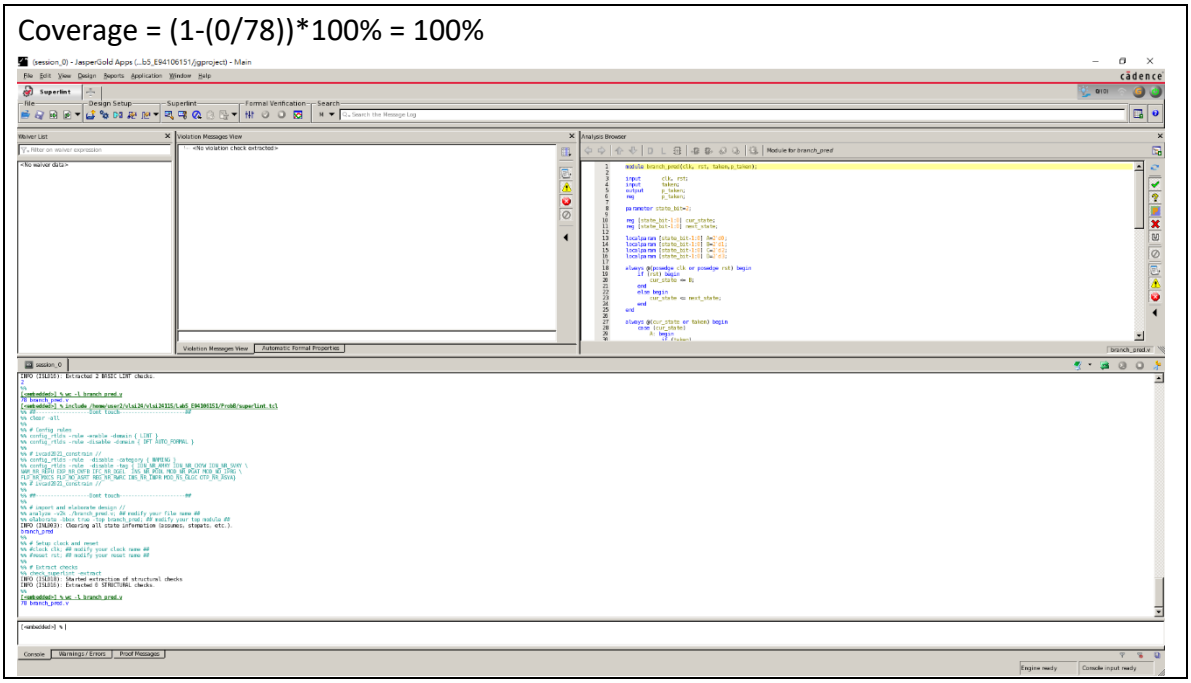


Explanation of your waveform :

以黃色標記指到的位置(RTL)為例，前一個 clock cycle 的 state 為 Predict Not Taken2(cur_state=3)，所以輸出 p_taken 為 0，而因為輸入 taken 為 1，所以下一個 state 變成 Predict Not Taken1(cur_state=2)，同理其他 clk 正緣也是跟這一樣可以用狀態機的圖表解釋。

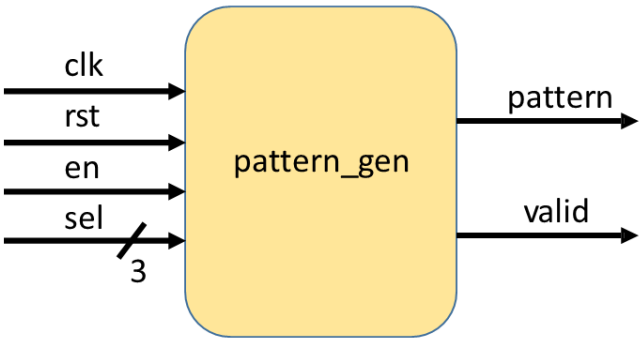
Superlint Coverage

Coverage = (1-(0/78))*100% = 100%



*ProbC: Design a pattern generator*

1) **Design a pattern generator which can create the following pattern and use mealy machine. The following is pattern generator specification.**



| sel [2:0] | pattern |
|---|---|
| 000 | 0000 |
| 001 | 0001 |
| 010 | 0010 |
| 011 | 0011 |
| 100 | 1100 |
| 101 | 1101 |
| 110 | 1110 |
| 111 | 1111 |

| Signal | Bits | Type | Description |
|---|---|---|---|
| clk | 1 | input | clock |
| rst | 1 | input | reset, active high |
| en | 1 | input | When en is high, the system will start to make pattern. The pattern will be created once. If the host want to create the next pattern, it should pull down the en to 0,then restart en. |
| sel | 3 | input | According different sel signal to make different pattern |
| pattern | 1 | output | Pattern output |
| valid | 1 | output | When valid is 1, pattern's value is valid. |

**2) Please describe your FSM in detail.**

| Explanation about your FSM |
|---|
| 根據上面的表格我設計了一個 5 個 state 的 Mealy machine。state 為 R 時，valid 為 0 輸出無效，當 en 為時下個 state 為 A。state 為 A 時，valid 為 1 輸出有效、pattern 根據上面藍色表格 sel 輸出 0 或 1，當 en 為時下個 state 為 B。 state 為 B 時，valid 為 1 輸出有效、pattern 根據上面藍色表格 sel 輸出 0 或 1，當 en 為時下個 state 為 C。 state 為 C 時，valid 為 1 輸出有效、pattern 根據上面藍色表格 sel 輸出 0 或 1，當 en 為時下個 state 為 D。 state 為 D 時，valid 為 1 輸出有效、pattern 根據上面藍色表格 sel 輸出 0 或 1，當 en 為時下個 state 為 R。 |

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

| Timing (slack) | Area (total cell area) | Power (total) |
|---|---|---|
| **0.30** | **10.264320 $um^2$** | **1.5215e-02 mW** |

**4) Please attach your design waveforms.**

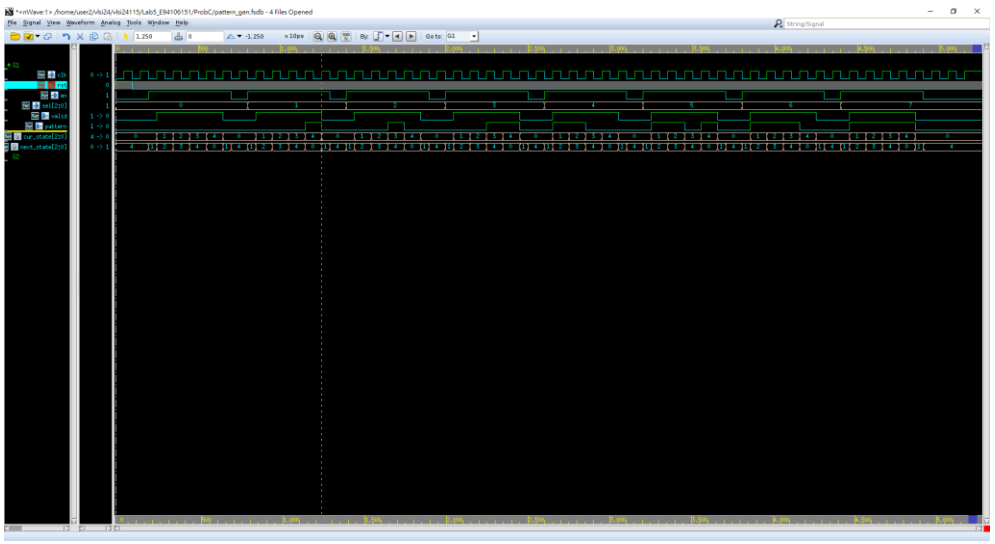
Your simulation result on the terminal.

RTL:

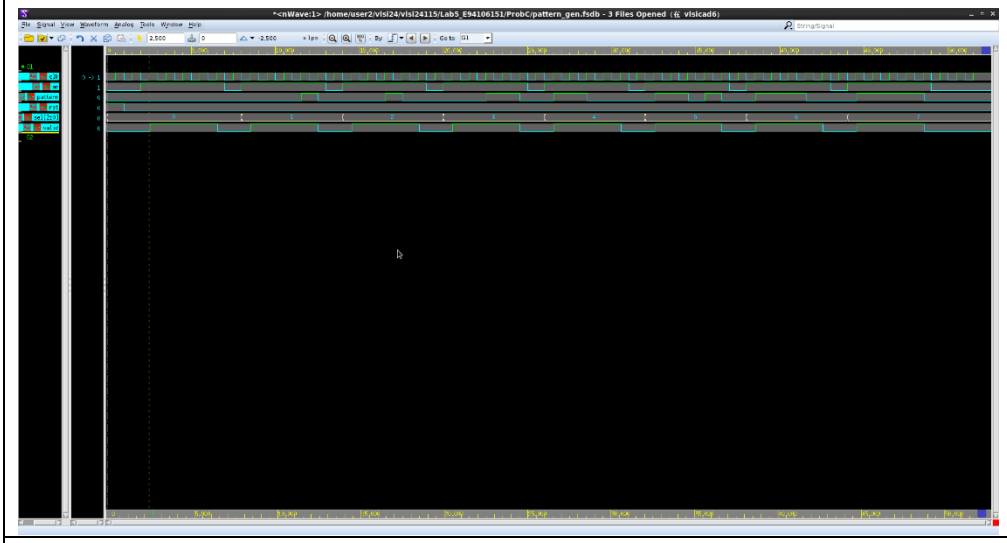

Gate:


Your waveform :

RTL:



Gate:

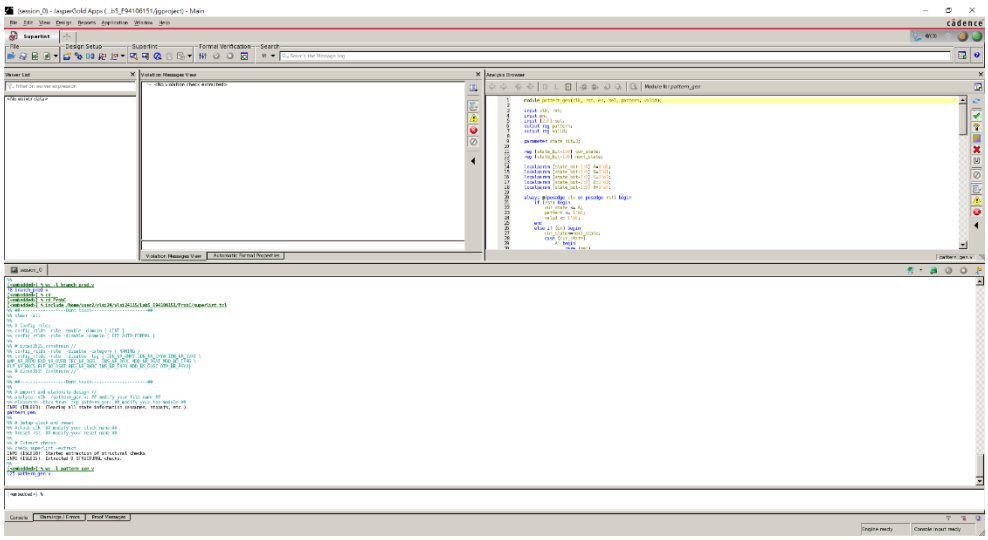


Explanation of your waveform :

以 1050ps 的位置(RTL)為例，前一個 clock cycle 的 state 為 C(cur_state=2)、sel 為 1，所以輸出有效(valid=1)、pattern 為 0、下一個 state 變成 D(cur_state=3)，同理其他 clk 正緣也是跟這一樣可以用狀態機的圖表解釋。

Superlint Coverage

Coverage = (1-(0/125))*100% = 100%



5) **At last, please write the lessons learned from this lab session, or some suggestions for this lab session. Thank you.**

**在這次的實驗中，我學到兩種有限狀態機要如何用 verilog 實現，還有如何用 design version 這個 tool 做時序電路的合成，感謝助教每次都將步驟講解得很清楚，讓我們不容易在過程中做錯，還有最感謝黃浚喆同學的解答，解決了我 Gate-level simulation 無法 compile 的問題。希望之後的實驗課時，講到 clock specification 時，助教能稍微提一下 set time 和 hold time 是什麼？讓我能更加理解，什麼情況下需要用到 add buffer on clock path。**

*Appendix A : Commands we will use to check your homework*

| Problem | | Command |
|---|---|---|
| Lab | Compile | % vcs -R moore.v -full64 |
| | RTL-sim | % vcs -R moore_tb.v -debug_access+all -full64 +define+FSDB |
| | Gate-sim | % vcs -R moore_tb.v -debug_access+all -full64 +define+FSDB+syn |

| Problem | | Command |
|---|---|---|
| ProbA | Compile | % vcs -R det_seq.v -full64 |
| | RTL-sim | % vcs -R det_seq_tb.v -debug_access+all -full64 +define+FSDB |
| | Gate-sim | % vcs -R det_seq_tb.v -debug_access+all -full64 +define+FSDB+syn |
| ProbB | Compile | % vcs -R branch_pred.v -full64 |
| | RTL-sim | % vcs -R branch_pred_tb.v -debug_access+all -full64 +define+FSDB |
| | Gate-sim | % vcs -R branch_pred_tb.v -debug_access+all -full64 +define+FSDB+syn |
| ProbC | Compile | % vcs -R pattern_gen.v -full64 |
| | RTL-sim | % vcs -R pattern_gen_tb.v -debug_access+all -full64 +define+FSDB |
| | Gate-sim | % vcs -R pattern_gen_tb.v -debug_access+all -full64 +define+FSDB+syn |