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	
劉冠妤	E24116152	
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**.)

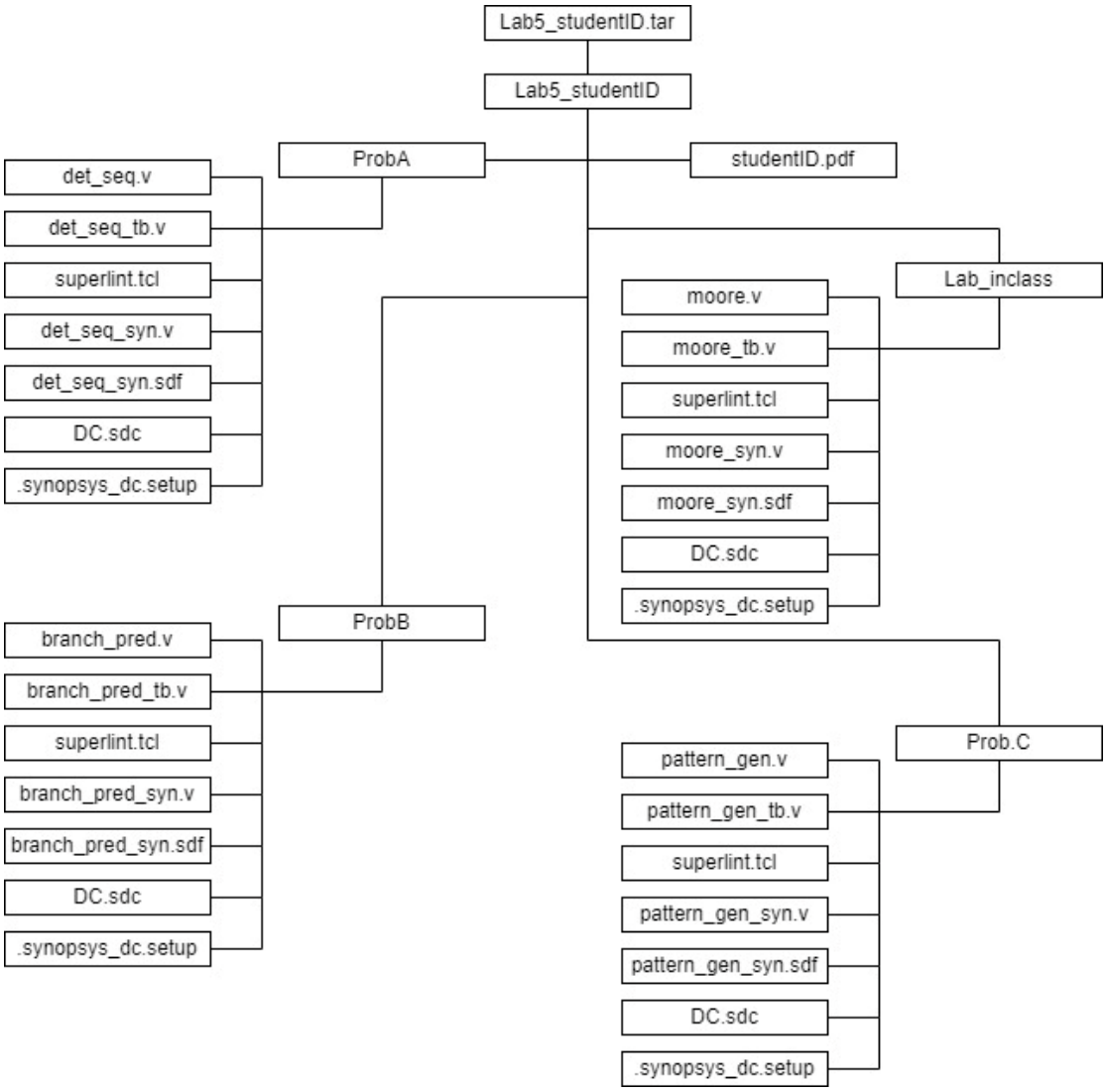
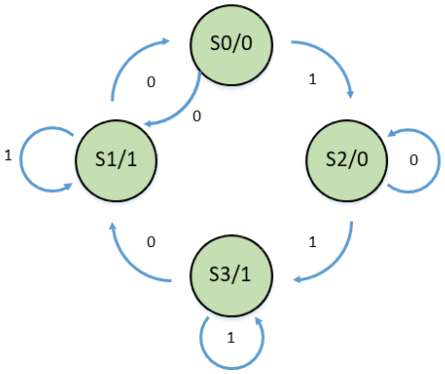


Fig.1 File hierarchy for Homework submission

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 0: when current state==S0 or current state==S2

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

Explanation about your FSM
In this FSM, there are two main parts: the combinatorial logic part and the sequential logic part. **Sequential Logic Part** The sequential logic part is defined by the always @(posedge clk or posedge rst) block. It handles the state transitions and the reset behavior of the FSM. If the reset signal (rst) is high, the current state (cs) is set to the initial state (s0). If the reset signal is not active, the current state (cs) is updated to the next state (ns) determined by the combinatorial logic. This block ensures that state transitions occur on the rising edge of the clock signal (clk). **Combinatorial Logic Part** 1. Next State Logic (always @(cs or din)) This block determines the next state (ns) based on the current state (cs) and the input (din). From s0 to s1 if din is 0, else to s2. From s1 to s0 if din is 0, else to s1. From s2 to s2 if din is 1, else to s3. From s3 to s1 if din is 0, else to s3. Default transition is to s0.

2. Output Logic (always @(cs))

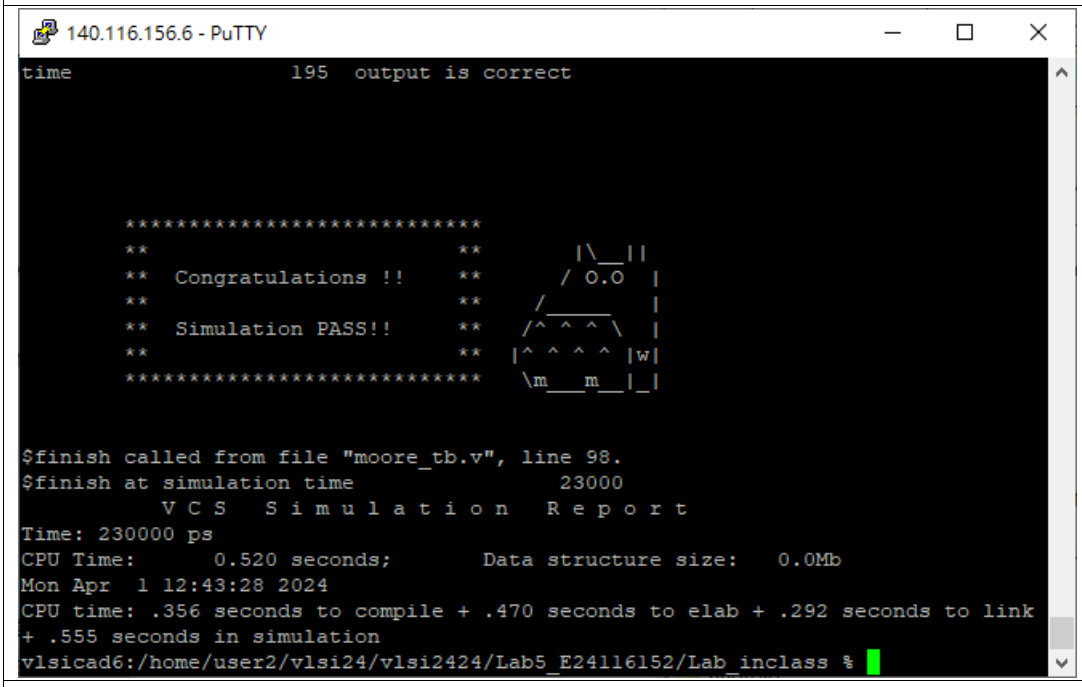
This block determines the output (qout) based on the current state (cs). Output is 0 in states s0 and s1. Output is 1 in states s2 and s3. Default output is 0.

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

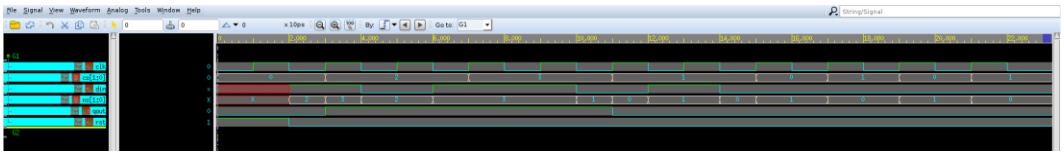
Timing (slack)	Area (total cell area)	Power (total)
0.32	**4.147200 μm^2**	**6.7439 μW**

4) Please attach your design waveforms.

Your simulation result on the terminal.



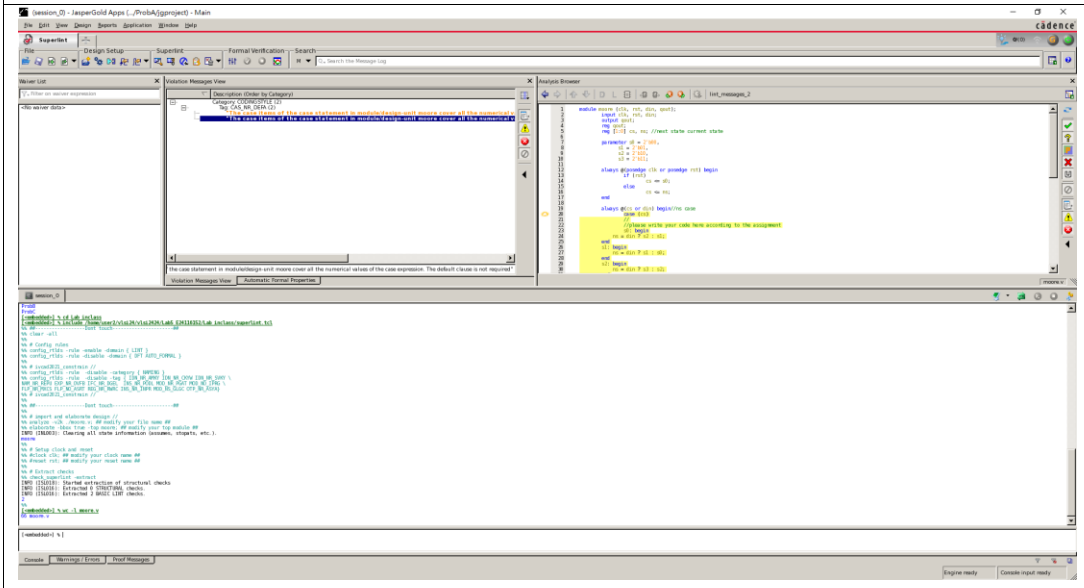
Your waveform :



Explanation of your waveform :

We can observe two things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates the Current state when posedge clk and passes in the Next state. The second is that the output qout is only related to the state at that time, because this is the design of Moore Machine. When state is 2b'10, 2b'11, qout is 1; when state is 2b'00, 2b'01, qout is 0.

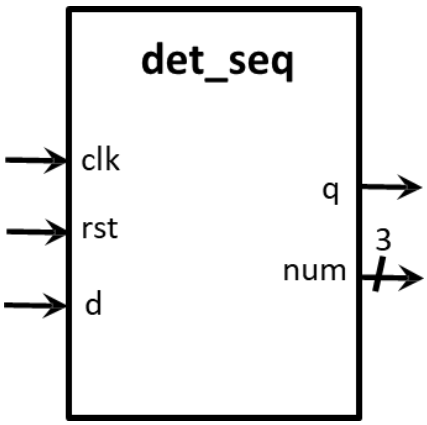
Superlint Coverage



Coverage = 96.97%

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
In this FSM, there are two main parts: the combinatorial logic part and the sequential logic part. **Sequential Logic Part:** The sequential logic part is defined by the always @(posedge clk or posedge rst)block. It handles the state transitions and the reset behavior of the FSM. If the reset signal (rst) is high, the current state (cs) is set to the initial state (one). If the reset signal is not active, the current state (cs) is updated to the next state (ns) determined by the combinatorial logic. This block ensures that state transitions occur on the rising edge of the clock signal (clk). In addition, it also includes a part that updates the cumulative value num, which increases num by one in the current state, and the rest does not change the value of num.

Combinatorial Logic Part:

1. Next State Logic (always @(*))

This block determines the next state (ns) based on the current state (cs) and the input (d). Each state (one, two, three, four, five, six, correct) has a unique behavior defined by the case statement. For example, in state one, if d is 1, the next state is two; otherwise, it remains in one. The default case sets the next state to one if the current state does not match any defined cases.

2. Output Logic (always @(*))

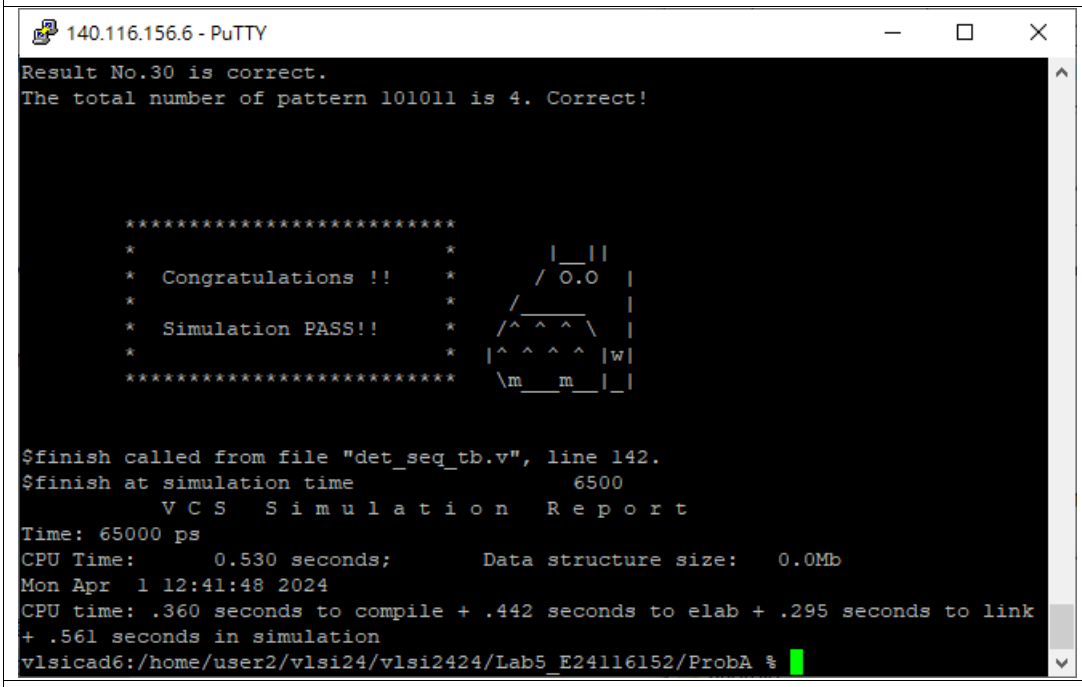
This block determines the output (q) based on the current state (cs). The output is 1 (1'd1) only when the current state is correct; otherwise, it is 0 (1'd0).

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

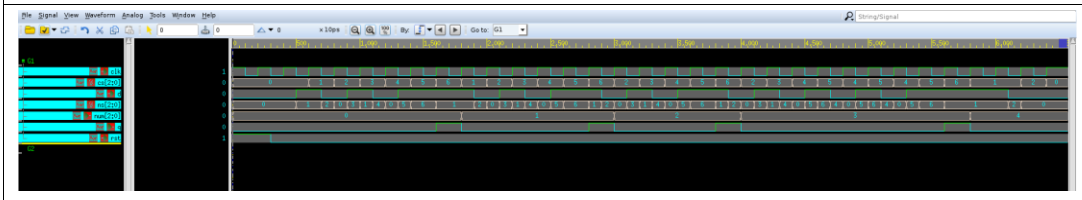
Timing (slack)	Area (total cell area)	Power (total)
0.29	**12.026880 μm^2**	**14.3890 μW**

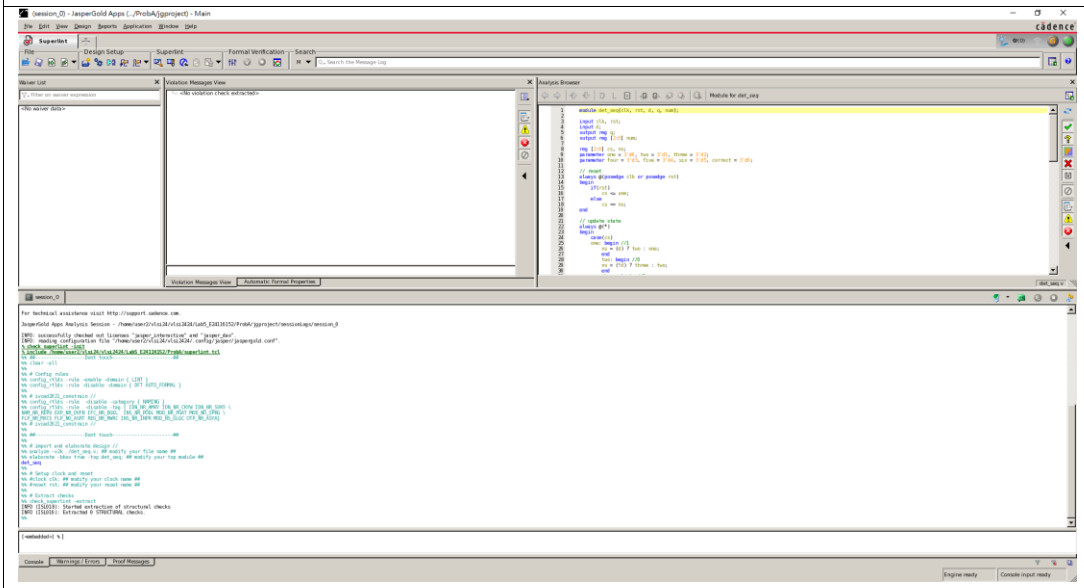
4) Please attach your design waveforms.

Your simulation result on the terminal.



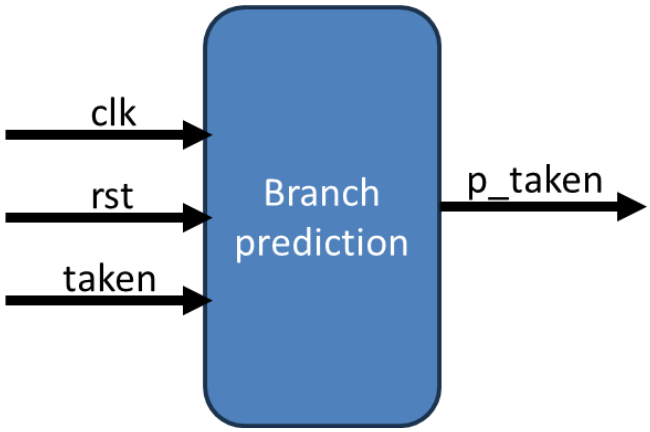
Your waveform :

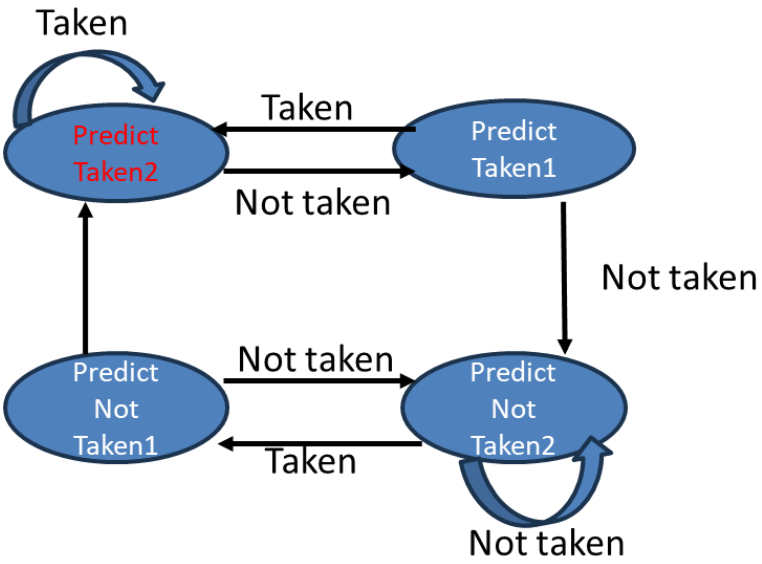


Explanation of your waveform :
We can observe three things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates the Current state when posedge clk and passes in the Next state. The second is that the output q is only related to the state at that time, because this is the design of Moore Machine. When state is correct, q is 1, otherwise, q is 0. The last part is the part that outputs num. When num is in the correct state, that is, when a set of target numbers 101011 appears completely, num is equal to itself plus one, and is triggered with the positive edge of the clock.
Superlint Coverage

Coverage = 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
In this FSM, there are two main parts: the combinatorial logic part and the sequential logic part. **Sequential Logic Part:** The sequential logic part is defined by the always @(posedge clk or posedge rst)block. It handles the state transitions and the reset behavior of the FSM. If the reset signal (rst) is high, the current state (cs) is set to the initial state (p0). If the reset signal is not active, the current state (cs) is updated to the next state (ns) determined by the combinatorial logic. This block ensures that state transitions

occur on the rising edge of the clock signal (clk).

Combinatorial Logic Part:

1. Next State Logic (always @(cs or taken))

   This block determines the next state (ns) based on the current state (cs) and the input (taken). Each state (pt1, pt2, pnt1, pnt2) has a unique behavior defined by the case statement. For example, in state pt1, if taken is 1, the next state is pt2; otherwise, it is pnt2. The default case sets the next state to pt1 if the current state does not match any defined cases.

2. Output Logic (always @(cs))

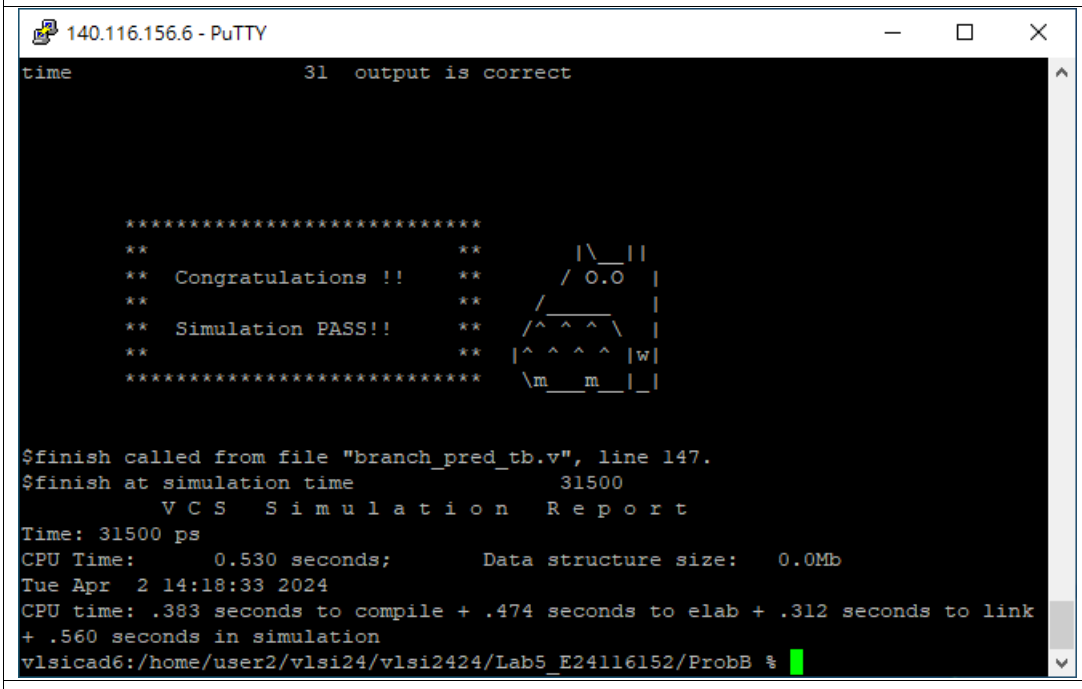
   This block determines the output (p_taken) based on the current state (cs). The output is 1 (1'd1) when the current state is pt1 or pt2; otherwise, it is 0 (1'd0).

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

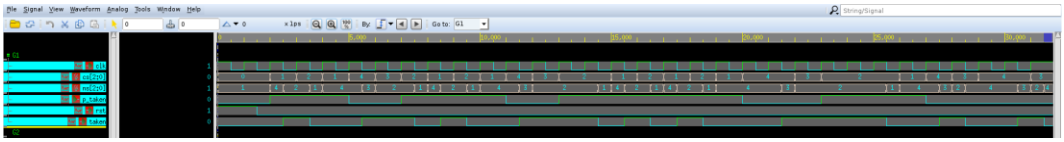
Timing (slack)	Area (total cell area)	Power (total)
0.30	7.413120 μm^2	9.3079 μW

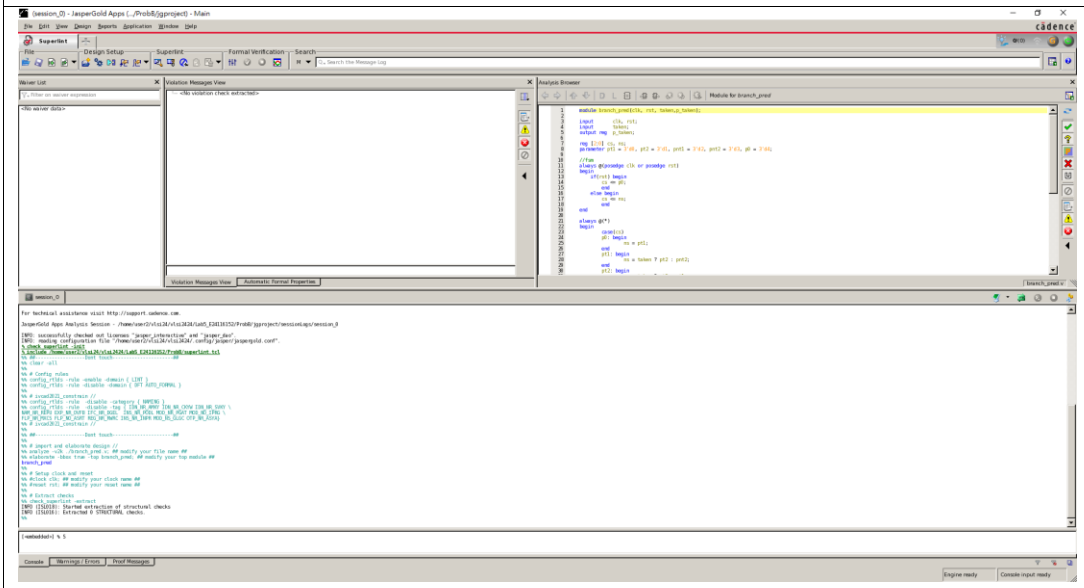
4) Please attach your design waveforms.

Your simulation result on the terminal.



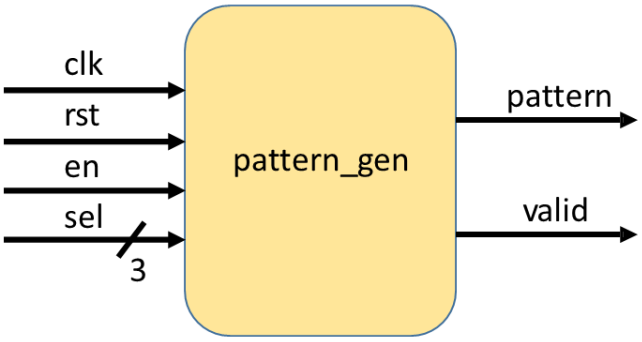
Your waveform :



Explanation of your waveform :
We can observe three things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates the Current state when posedge clk and passes in the Next state. The second is that the output p_taken is only related to the state at that time, because this is the design of Moore Machine. When state is predict_taken1 or 2, p_taken is 1, otherwise, p_taken is 0. The last part is in addition to the regular four states, I gave an additional starting state p0 to ensure that my state will start from predict_taken1.
Superlint Coverage

Coverage = 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
In this FSM, there are two main parts: the combinatorial logic part and the sequential logic part. **Sequential Logic Part:** The sequential logic part is defined by the always @(posedge clk or posedge rst)block. It handles the state transitions and the reset behavior of the FSM. If the reset signal (rst) is high, the current state (cs) is set to the initial state (idle). If the reset signal is not active, the current state (cs) is updated to the next state (ns) determined by the combinatorial logic. This block ensures that state transitions occur on the rising edge of the clock signal (clk). **Combinatorial Logic Part:** 1. Next State Logic (always @(cs or en)) This block determines the next state (ns) based on the current state (cs) and the enable signal (en). When in the idle state, if the enable signal is high, the next state is p0; otherwise, it remains in the idle state. The FSM transitions from p0 to p1, p1 to p2, p2 to p3, and p3 back to idle. 2. Output Logic (always @(cs)) This block determines the output (pattern) based on the current state (cs) and the select signal (sel). The output pattern is determined by the value of sel at different positions depending on the current state. The output valid is set to 1 when the FSM is not in the idle state and 0 otherwise.

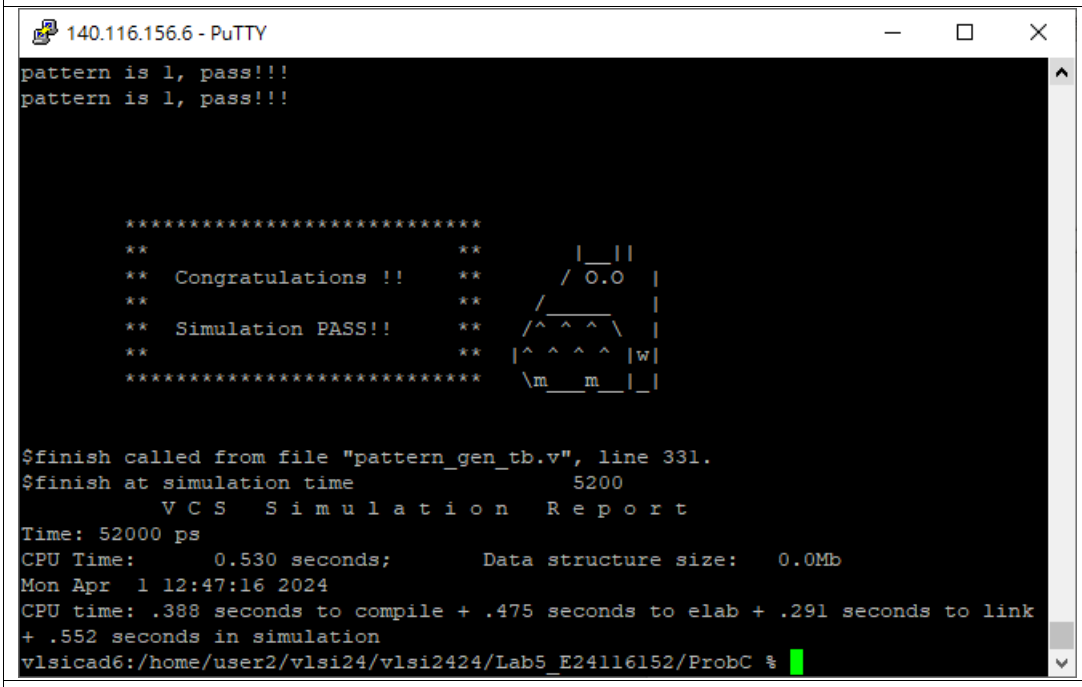
3) **After synthesizing your design, you may have some information about the circuit. Please fill in the following form.**

Timing (slack)	Area (total cell area)	Power (total)
8.76	**6.946560 μm^2**	**82.3884 nW**

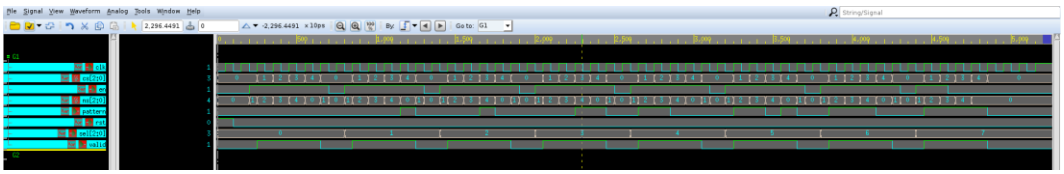
The value of timing is larger, but this is the result of synthesis according to the value on the handout (clock period is 10).

4) Please attach your design waveforms.

Your simulation result on the terminal.



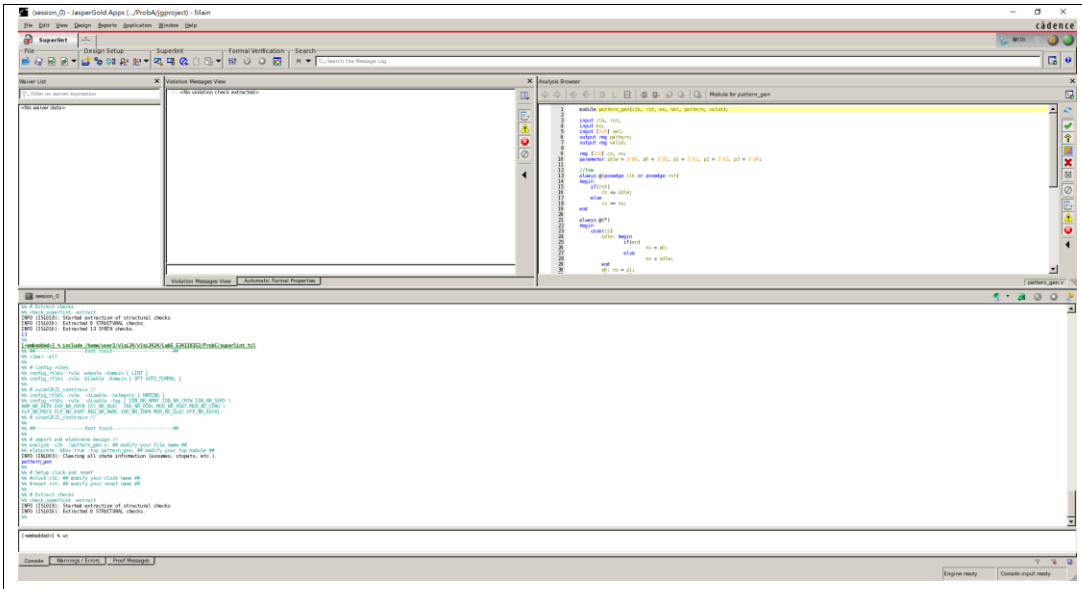
Your waveform :



Explanation of your waveform :

We can observe three things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates the Current state when posedge clk and passes in the Next state. The second is that the output sel is not only related to the state at that time but also related to the input, because this is the design of Mealy Machine. When state is p0, output pattern is sel [2]; when state is p1, output pattern is also sel [2]; when state is p2, output pattern is sel [1]; When state is p3, output pattern is sel [0]; when state is idle, output pattern is 0. The last part is when the en signal is one, the next state begins to change. If the en signal is zero, the next state is idle.

Superlint Coverage



Coverage = 100%

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

After this experiment, I learned more about the application and difference of Sequential and Combinatorial Logic, and also learned the difference and application methods of Moore and Mealy machine. I think the teacher is very attentive and serious in class. I have benefited a lot, but the synthesis method It's a bit complicated, I hope I can make more progress.

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