**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!

1. 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!

1. 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.
2. **If you upload a dead body, which we cannot even compile, you will get NO credit!**
3. **All Verilog file before synthesizing should get at least 95% Superlint Coverage.**
4. Lab5\_Student\_ID.tar (English alphabet of Student\_ID should be **capital**.)

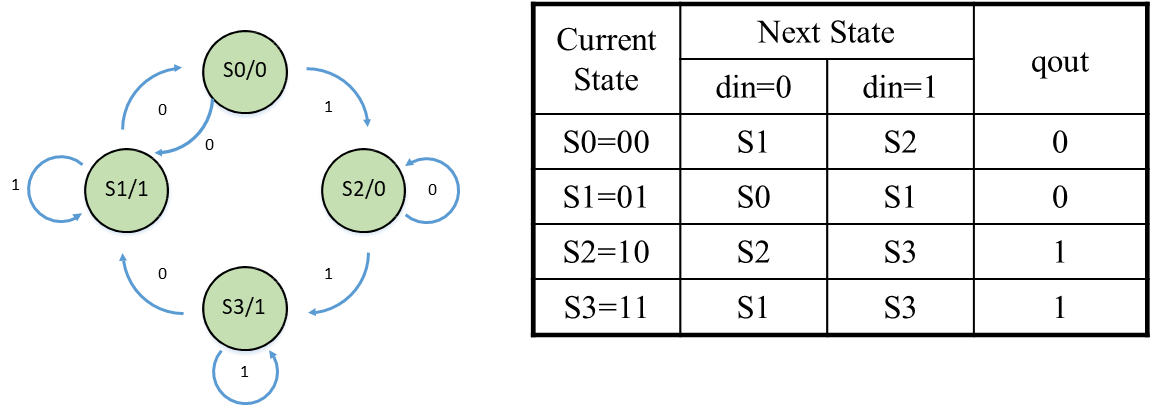
一張含有 文字, 圖表, 平行, 螢幕擷取畫面 的圖片

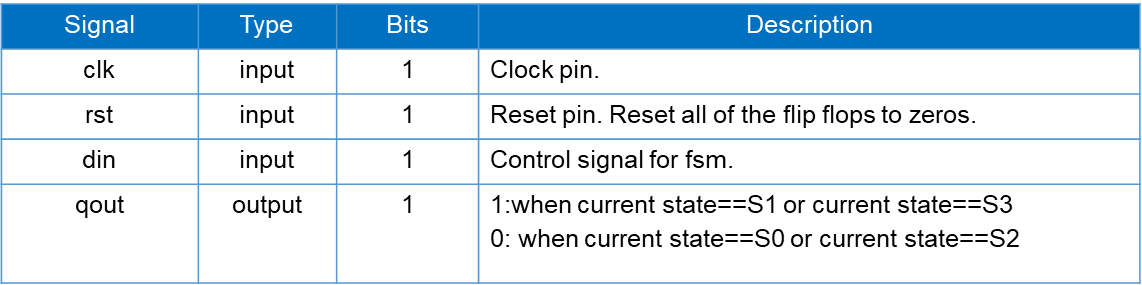
自動產生的描述

Fig.1 File hierarchy for Homework submission

Lab in class: Design a circuit “Moore machine”

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.)**





1. **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. |

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

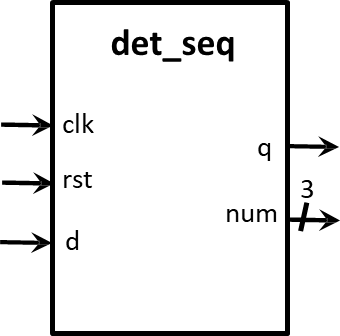
|  |  |  |
| --- | --- | --- |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
| **0.32** | **4.147200 𝜇𝑚2** | **6.7439 𝜇W** |

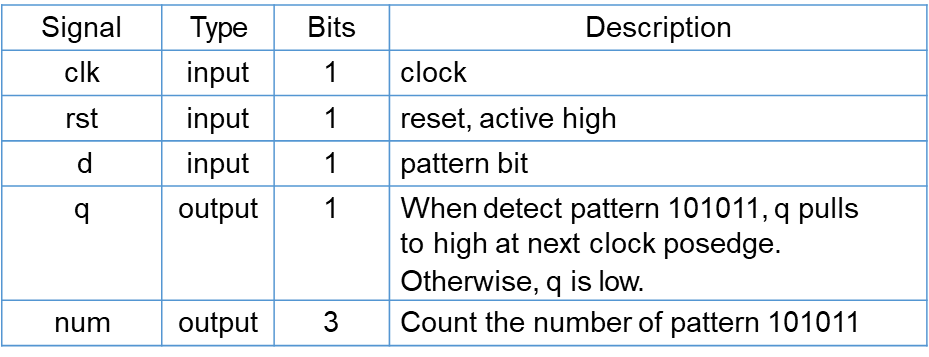
1. **Please attach your design waveforms.**

|  |
| --- |
| 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.)**





1. **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). |

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

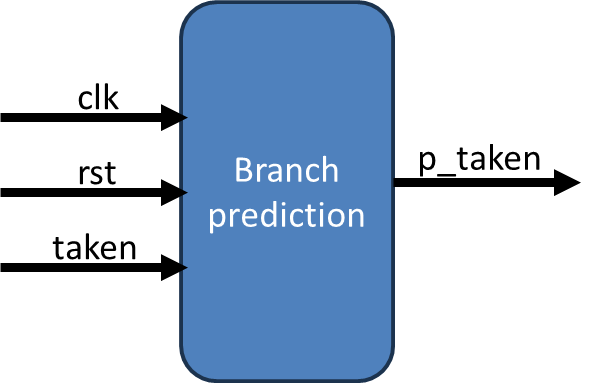
|  |  |  |
| --- | --- | --- |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
| **0.29** | **12.026880 𝜇𝑚2** | **14.3890 𝜇W** |

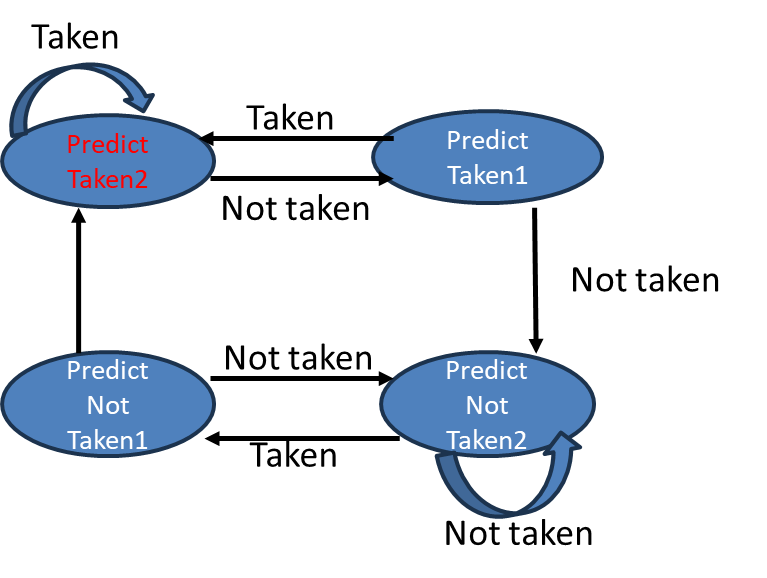
1. **Please attach your design waveforms.**

|  |
| --- |
| 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.)**





1. **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). |

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

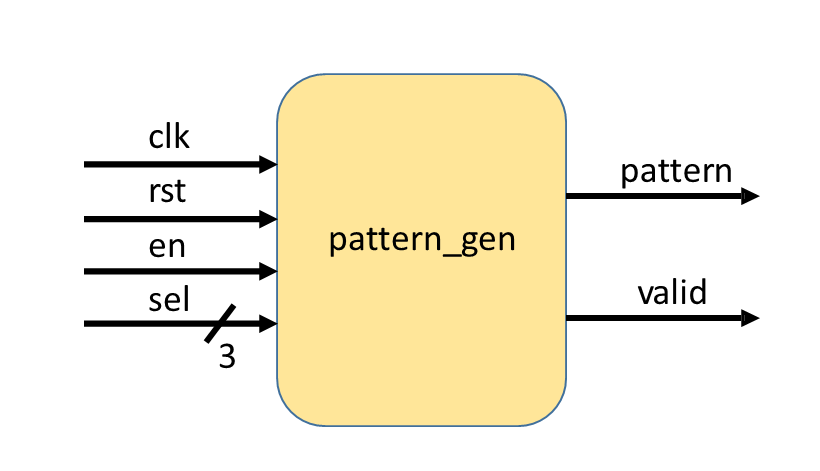
|  |  |  |
| --- | --- | --- |
| **Timing (slack)** | **Area (total cell area)** | **Power (total)** |
| **0.30** | **7.413120 𝜇𝑚2** | **9.3079 𝜇W** |

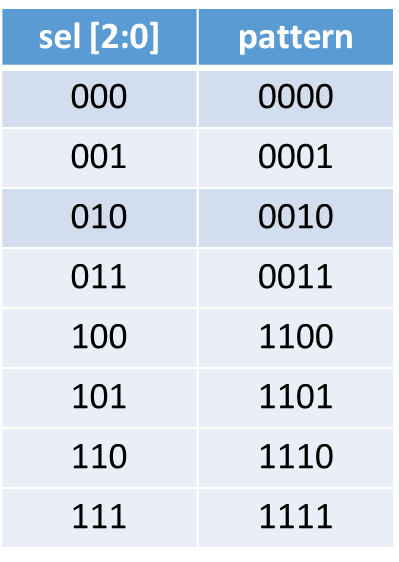
1. **Please attach your design waveforms.**

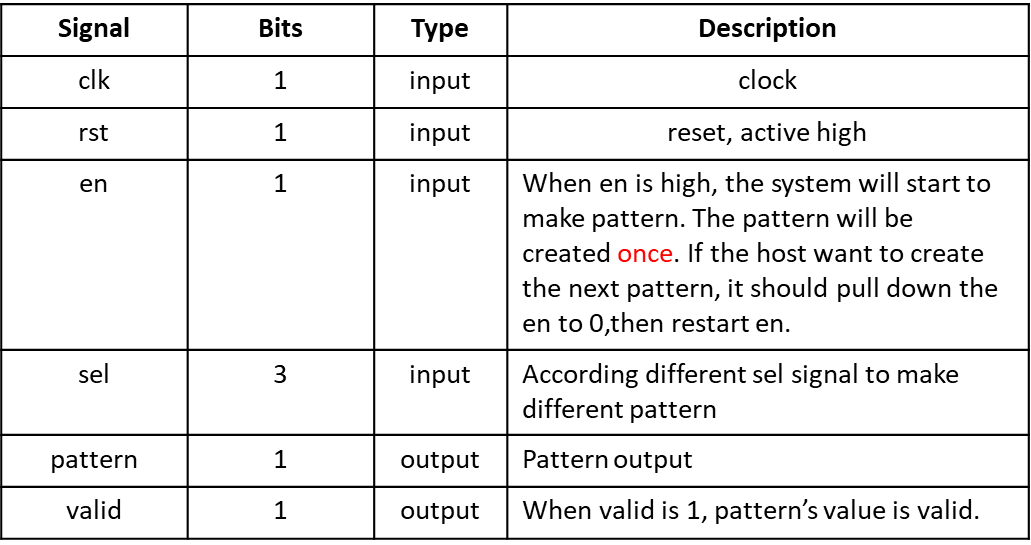
|  |
| --- |
| Your simulation result on the terminal. |
|  |
| Your waveform : |
|  |
| Explanation of your waveform : |
| We can observe three things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates 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.**

****

****

****

1. **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. |

1. **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 𝜇𝑚2** | **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).**

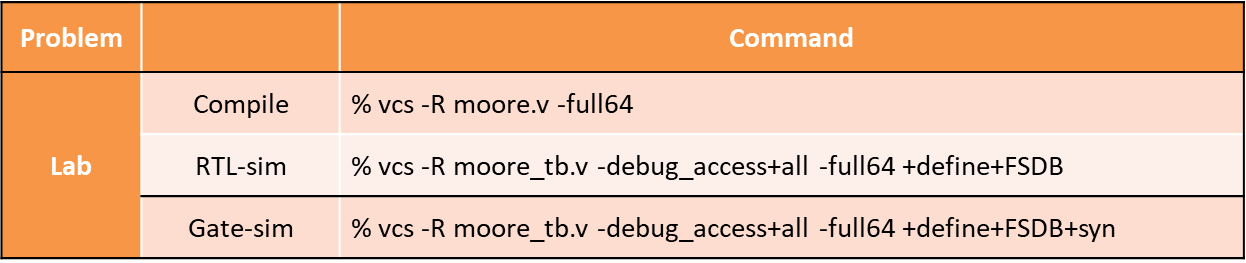
1. **Please attach your design waveforms.**

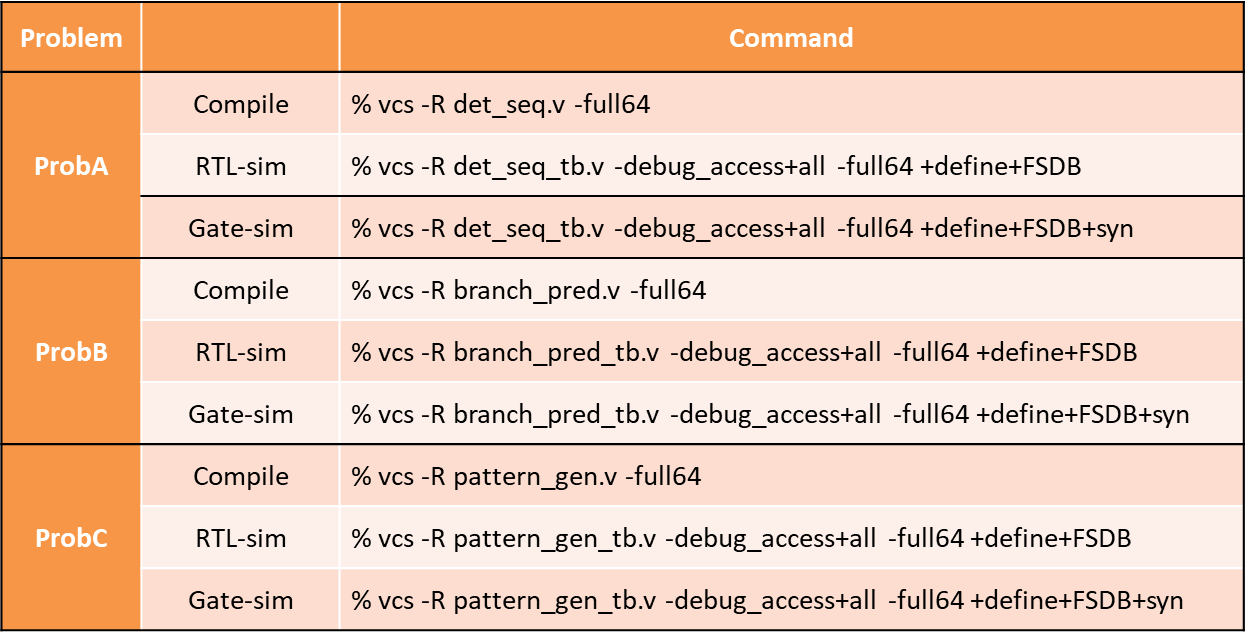
|  |
| --- |
| Your simulation result on the terminal. |
|  |
| Your waveform : |
|  |
| Explanation of your waveform : |
| We can observe three things from the figure: The first is that the Current state is triggered on the positive edge of the clock. This is because the design of this FSM only updates 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% |

1. **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



****