Welcome to The Hardware Design & Lab!

Fall 2023
Lab 4: Finite State Machines

Prof. Chun-Yi Lee

Department of Computer Science
National Tsing Hua University

Agenda

- Lab 4 Outline
- Lab 4 Basic Questions
- Lab 4 Advanced Questions

Lab 4 Outline

- Basic questions (1.5%)
  - Individual assignment
  - Due on 10/26/2023 (Thu). In class.

- Advanced questions (5%)
  - Individual assignment
  - Demonstration on your FPGA board (In class)
  - Lab report should be submitted using PDF format
  - Assignment submission (Submit to EEClass)
    - EEClass submission (code, test bench, and report) due on 11/9/2023 (Thu) 23:59:59 for the Advanced Questions.
    - EEClass submission (code and report) due on 11/9/2023 (Thu) 23:59:59 for the FPGA Question

Lab 4 Rules

- Please note that grading will be based on **NCVerilog**
- You can use **ANY** modeling techniques

- If not specifically mentioned, we assume the following SPEC
  - **clk** is **positive edge triggered**
  - Synchronously reset the Flip-Flops when **rst_n == 1'b0, if there exists one rst_n signal in the specification**

Lab 4 Submission Requirements

- Source codes and testbenches
  - Please follow the templates EXACTLY
  - We will test your codes by TAs' testbenches

- Lab 4 report
  - Please submit your report in a single PDF file
  - Please draw the **block diagrams** and **state transition diagrams** of your designs
  - Please explain your designs in detail
  - Please explain how you test your design
  - What you have learned from Lab 4

Agenda

- Lab 4 Outline
- Lab 4 Basic Questions
- Lab 4 Advanced Questions

Basic Questions

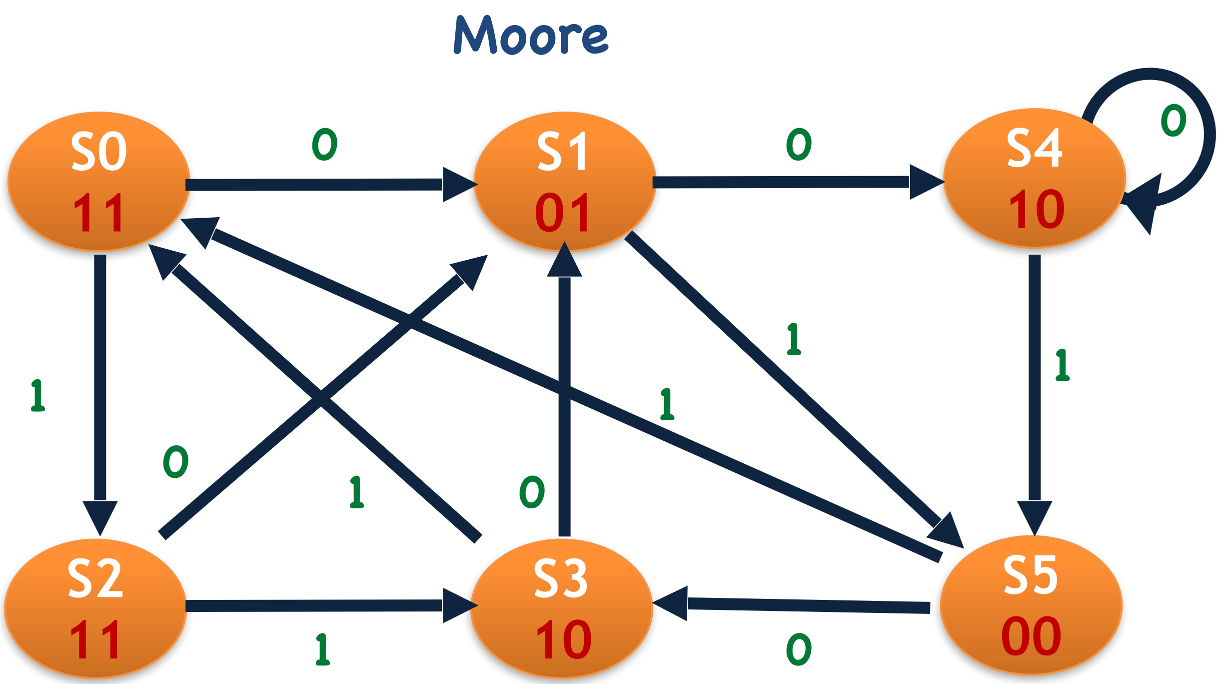
- Individual assignment

- Verilog questions (due on 10/26/2023. In class.)
  - Moore machine
  - Mealy machine
  - Many-to-one linear-feedback shift register
  - One-to-many linear-feedback shift register

- Demonstrate your work by waveforms

Verilog Basic Question 1

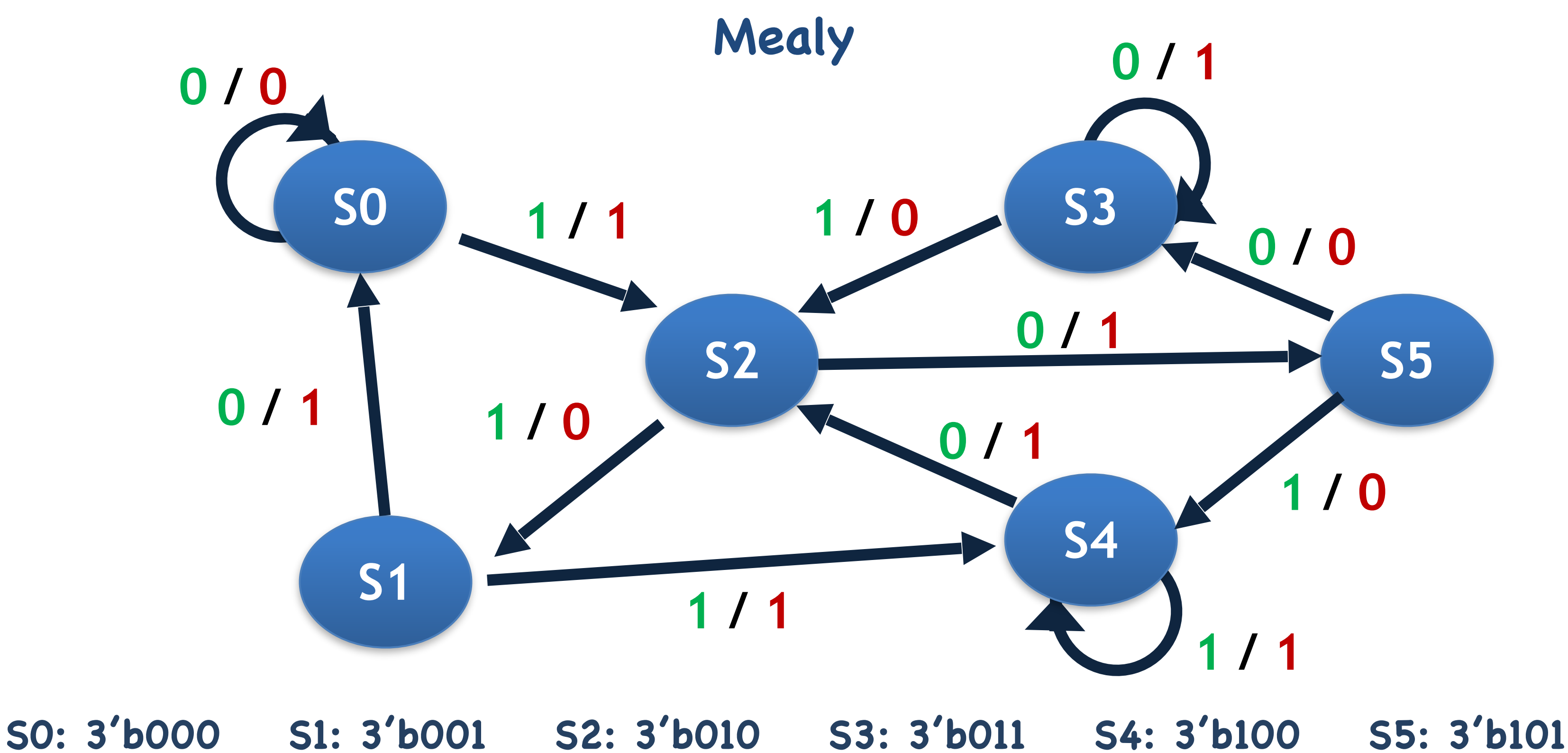
- Moore machine
  - Green represents input, while red represents output
  - Output your **current state** as well
  - When **rst_n** == 1’b0, **state** = S0



S0: 3’b000 S1: 3’b001 S2: 3’b010 S3: 3’b011 S4: 3’b100 S5: 3’b101

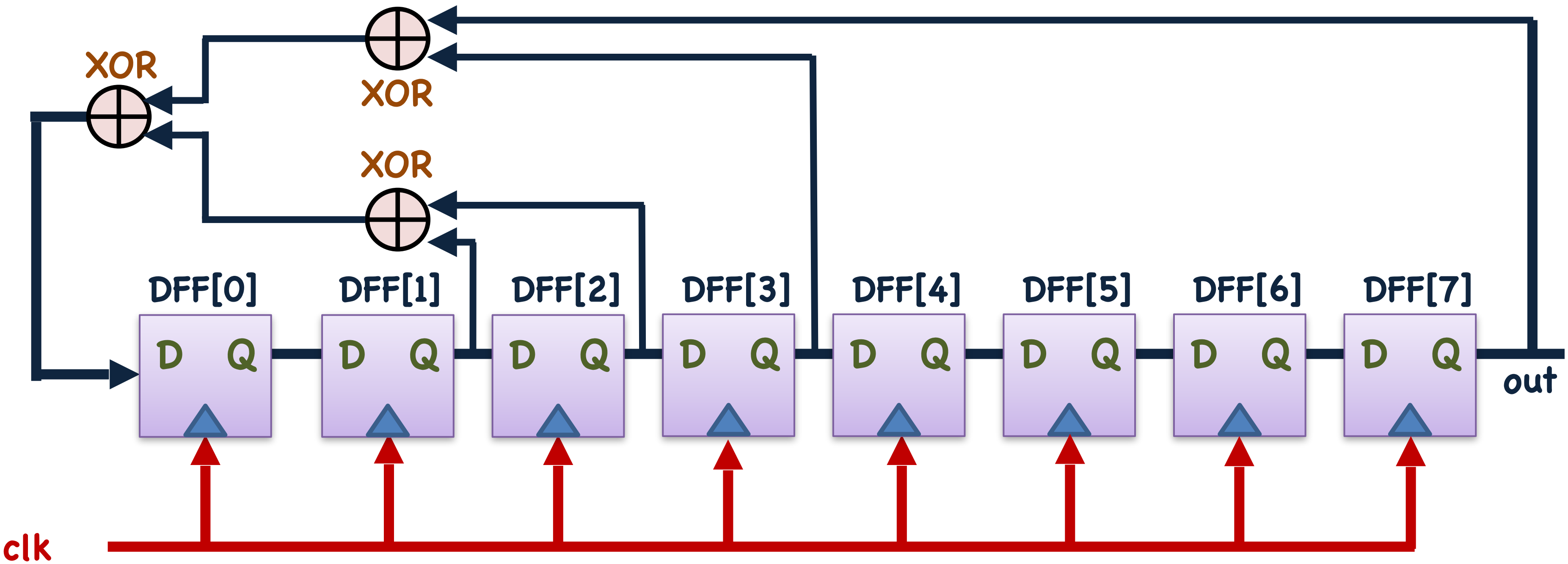
Verilog Basic Question 2

- Mealy machine
  - Green represents input, while red represents output
  - Output your **current state** as well
  - When **rst_n** == 1'b0, **state** = S0



Verilog Basic Question 3

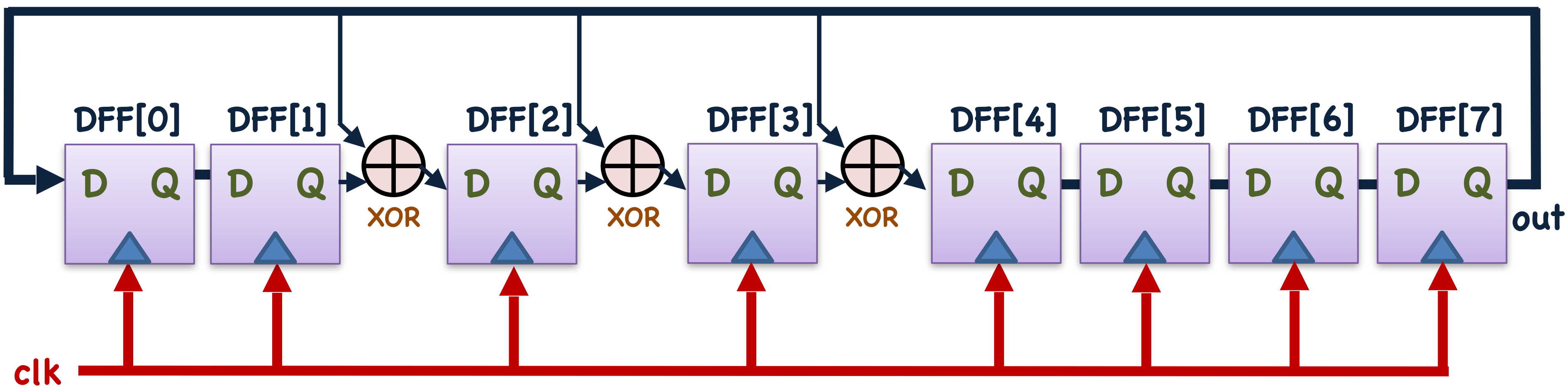
- Many-to-one linear-feedback shift register (LFSR)



- When **rst_n == 1'b0**, reset DFF[7:0] to **8'b10111101**
- Please draw the **state transition diagram** of the DFFs in LFSR for the first ten states after **rst_n** is raised to **1'b1** in your report
- Please describe what happens if we reset the DFFs to **8'd0** in your report

Verilog Basic Question 4

- One-to-many linear-feedback shift register (LFSR)



- When **RESET == 1’b0**, reset DFF[7:0] to **8'b10111101**
- Please draw the **state transition diagram** of the DFFs in LFSR for the first ten states after **rst_n** is raised to **1’b1** in your report
- Please describe what happens if we reset the DFFs to **8’d0** in your report

Agenda

- Lab 4 Outline
- Lab 4 Basic Questions
- Lab 4 Advanced Questions

Advanced Questions

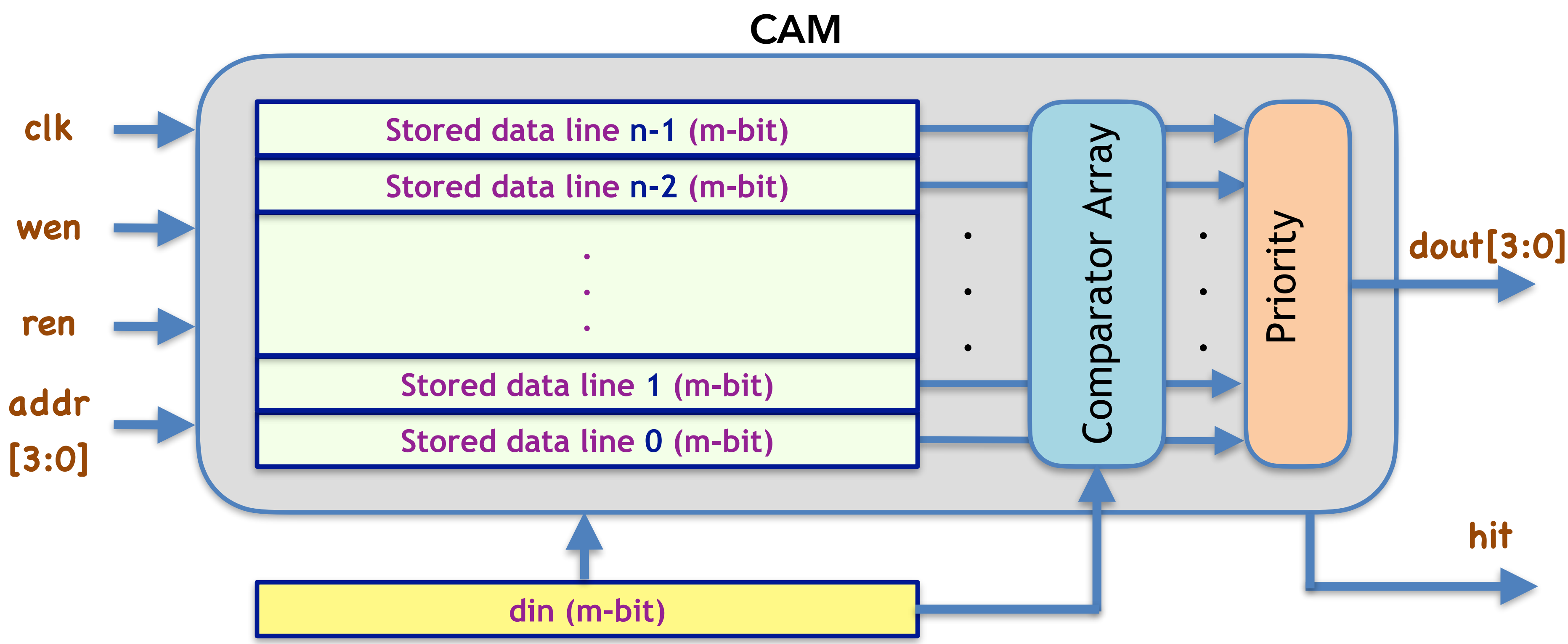
- Individual assignment

- Verilog questions
  - Source codes and the report due on 11/9/2023. 23:59:59.
    - Necessary: Content-addressable memory (CAM) design
    - Necessary: Scan chain design
    - Necessary: Built-in self test
    - Optional: Mealy machine sequence detector

- FPGA demonstration (due on 11/9/2023. In class.)
  - Necessary: Built-in self test

Verilog Advanced Question 1

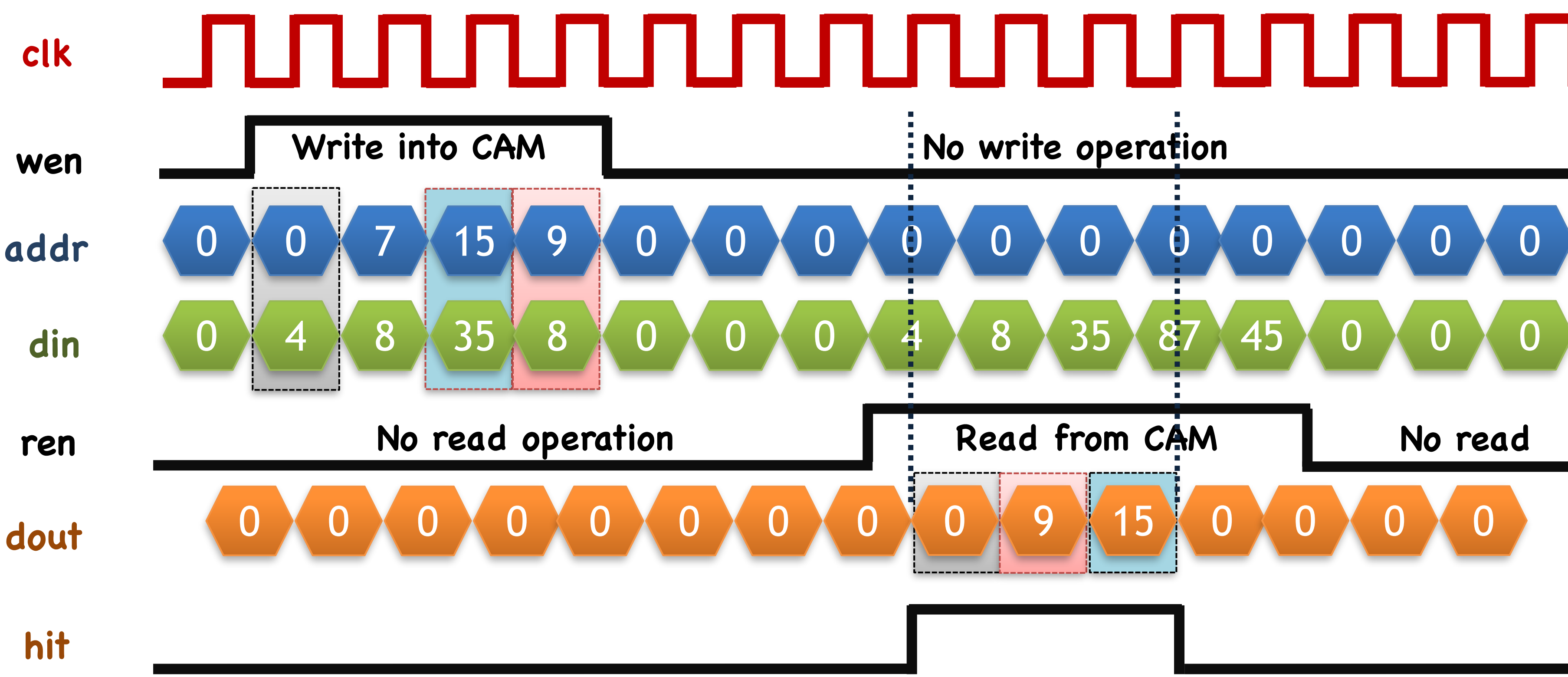
- **Content-addressable memory (CAM) design**
  - Design a CAM that stores **n** sets of **m**-bit data lines **(n = 16, m = 8)**
  - Input: **clk, wen, ren, addr[3:0], din[m-1:0]**
  - Output: **dout[3:0], hit**



Verilog Advanced Question 1 (Con't)

- When **wen == 1'b1**, write **din** to CAM[**addr**] and set both **dout** and **hit** to **1'b0**
- When **ren == 1'b1**:
  - If there is only one matching data in the CAM, set **dout** to the matching data's address and set **hit** to **1'b1**
  - If there are multiple matches in the CAM, set **dout** to the **largest address among them** and set **hit** to **1'b1**.
  - If there is no match in the CAM, set **dout** to **1'b0** and set **hit** to **1'b0**
- When both **wen** and **ren** are **1'b1**, **perform read operation only and ignore the write request**
- When both ren and wen are 1b'0, set **dout** to 1'b0 and set **hit** to 1'b0
- Please refer to the next page for example waveform

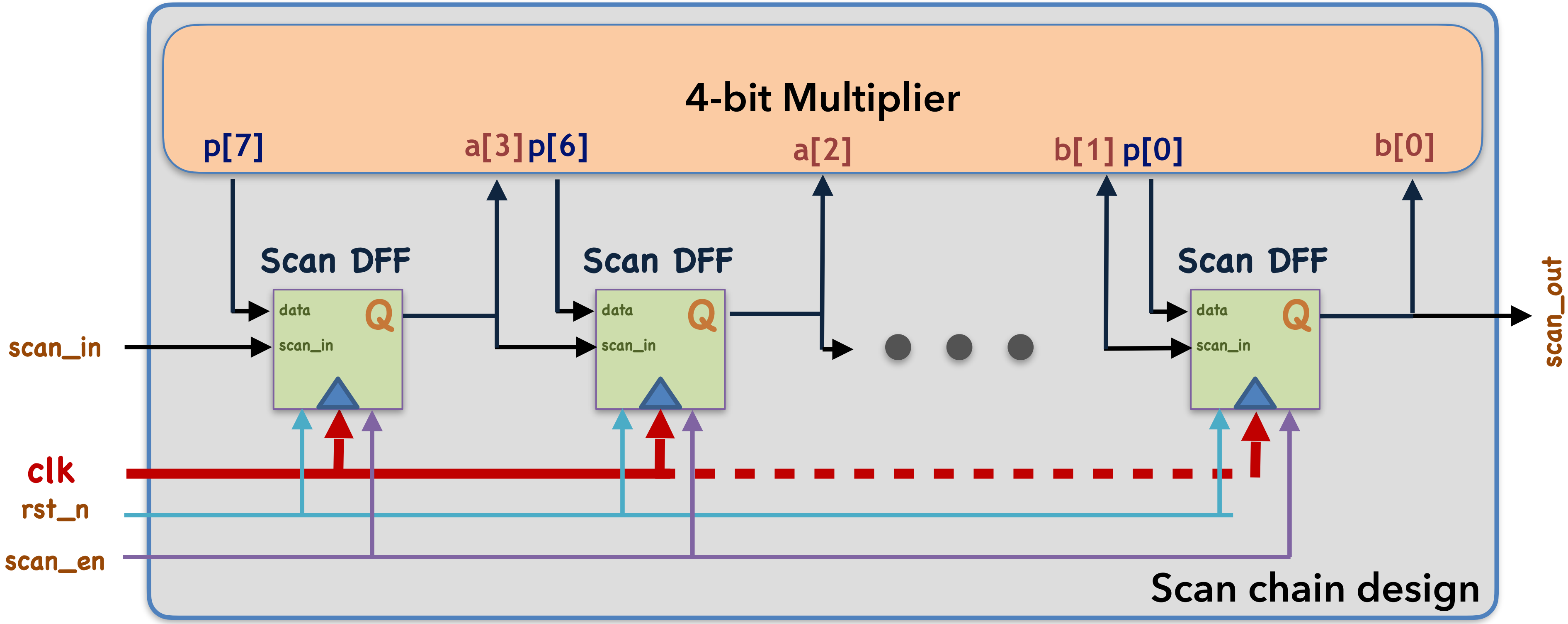
Verilog Advanced Question 1 (Con't)

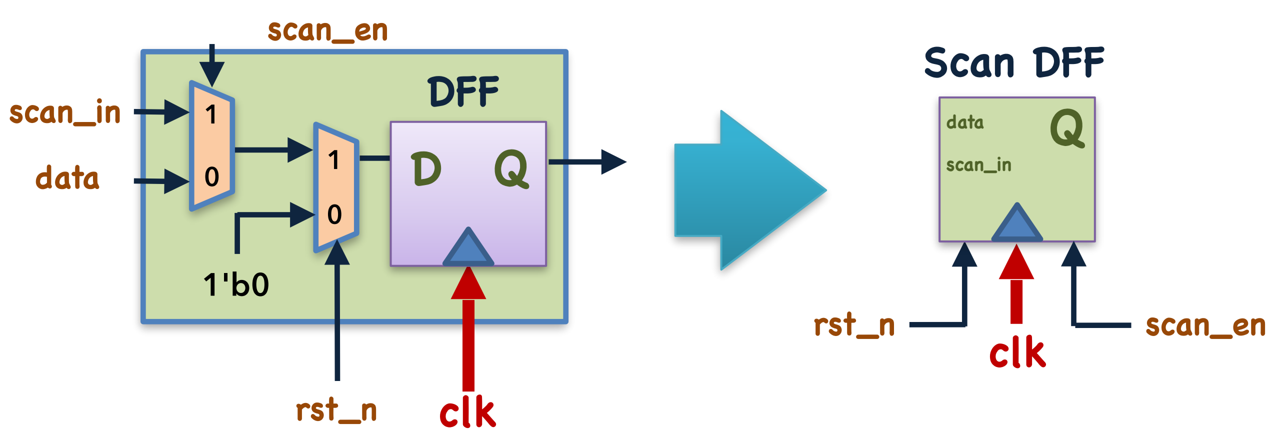


Verilog Advanced Question 2

- **Scan chain design**
- Scan chain is a technique used in design for testing. The objective is to make testing easier by providing a simple way to set and observe every flip-flop in a circuit. The structure of a scan chain is illustrated in the next page.
  - In order to achieve the above objective, the DFFs in a circuits are all replaced by a special type of DFF, called scan DFF (SDFF), which is also shown in the next page. An SDFF contains several extra ports: **scan_in** and **scan_en**, and is larger than the original DFF.
  - All the SDFFs are connected in a chain, which is called a scan chain.
- In this question, you are required to design a scan chain for a 4-bit multiplier, which is a combinational circuit and can be designed by any modeling technique.
  - Input: **clk**, **rst_n**, **scan_in**, **scan_en**
  - Output: **scan_out**
- Reset all SDFFs to **1’b0** when **rst_n** == 1’b0

Verilog Advanced Question 2 (Con't)

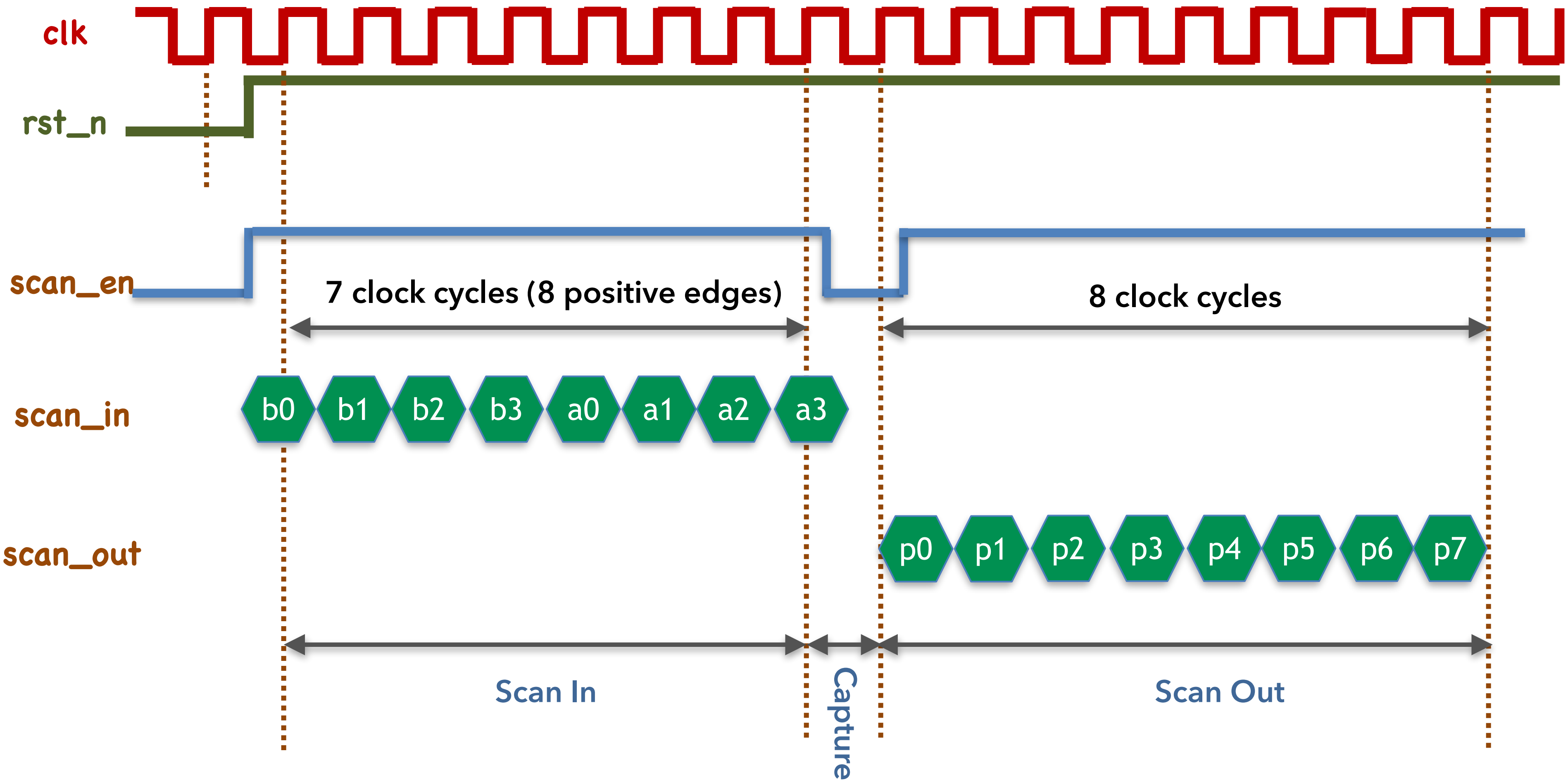




Verilog Advanced Question 2 (Con’t)

- **The behavior of a scan chain**
- The behavior of a scan chain contains three phases: **scan in**, **capture**, and **scan out**.
  - **Scan in**
    - In this phase, **scan_en** is set to **1’b1**, and a test pattern is scanned (shifted) from the **scan_in** port into the scan chain bit-by-bit.
  - **Capture**
    - In this phase, **scan_en** is set to **1’b0**, and the circuit performs its original functionality.
    - The inputs of the multiplier is provided by the values stored in SDFF. The output of the multiplier is stored back to the SDFFs at the positive clock edge.
  - **Scan out**
    - In this phase, **scan_en** is set to **1’b1** again, and the values stored in the SDFFs are shifted to the **scan_out** port of the scan chain bit-by-bit.
- In TA’s test bench, the **scan_en** signal is controlled **according to this three-phase behavior pattern** to test your scan chain design.
- Please refer to the next page for the example behavior waveform.

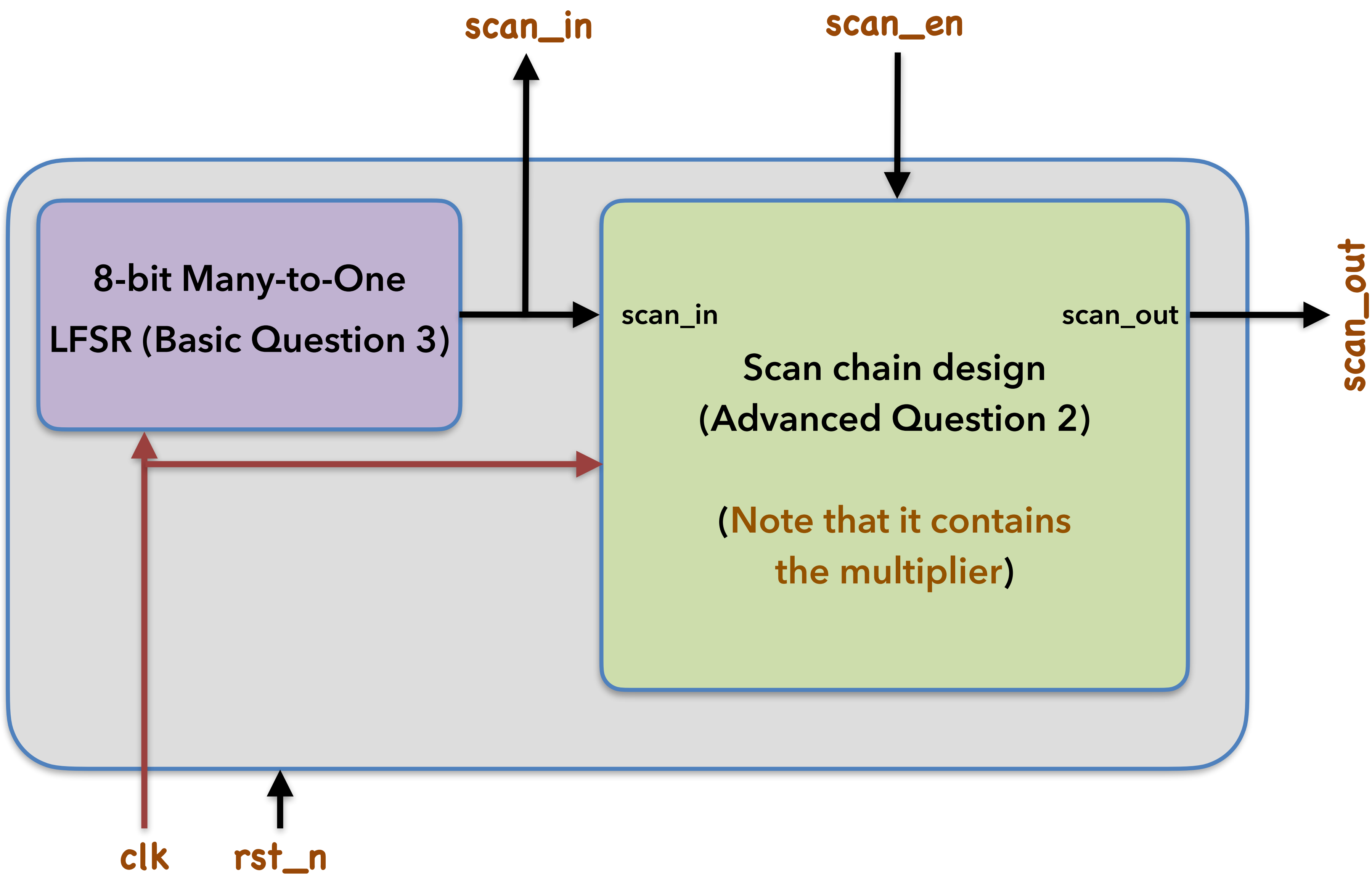
Verilog Advanced Question 2 (Con't)



Verilog Advanced Question 3

- **Built-in self test (BIST)**
- In the previous question, we designed a scan chain. Now we add a test pattern generator in front of it. The test pattern generator is implemented by a 8-bit many-to-one LFSR, which is the same as the design in the basic question 3. Since the test pattern generator is inside a chip, this architecture is called "built-in self test (BIST)".
- Please reuse the scan chain from the advanced question 2
- **Please modify your LFSR from the basic question 3 so that only the MSB of the LFSR is shifted into the scan chain.**
- Typically, a circuit with BIST does not have **scan_in** and **scan_out** ports. However, for the grading purposes, the two ports are set as output ports, so as to allow them to be observable.
- Input: **clk**, **rst_n**, **scan_en**
- Output: **scan_in**, **scan_out**

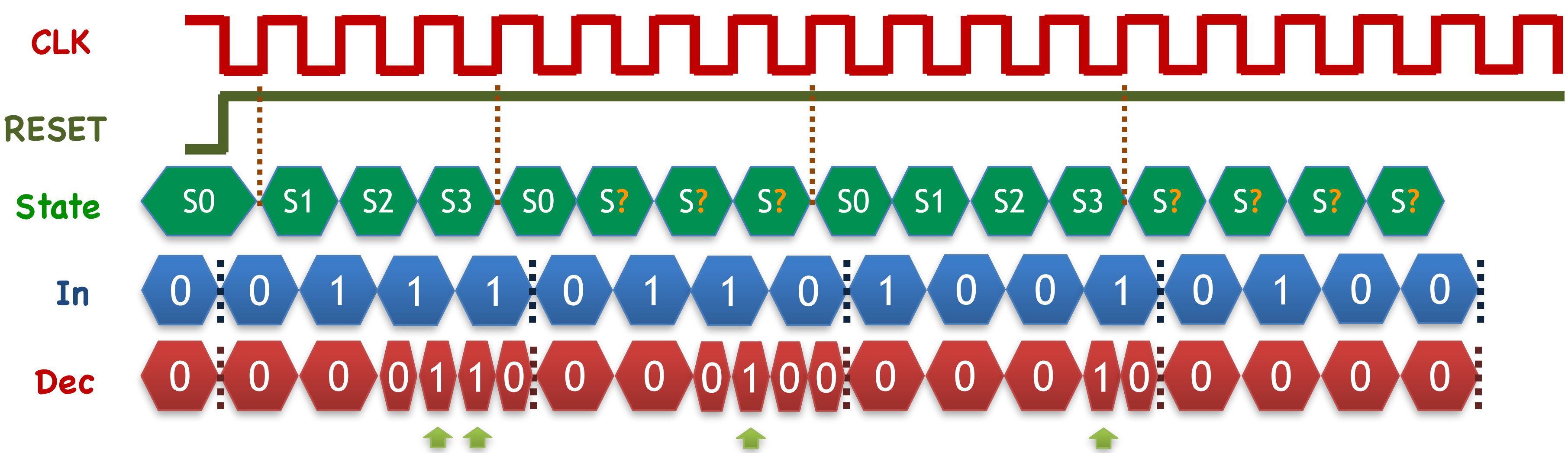
Verilog Advanced Question 3 (Con’t)



Verilog Advanced Question 4

- **Mealy machine sequence detector**
  - 1-bit input **In** and 1-bit output **Dec**
  - When the four bit sequence is **0111**, **1001**, or **1110**, **Dec** is set to **1**
  - Re-detect the sequence **every four bits**
  - Please draw your state diagram in your report

Complete the **complete** state diagram in your report



Advanced Questions

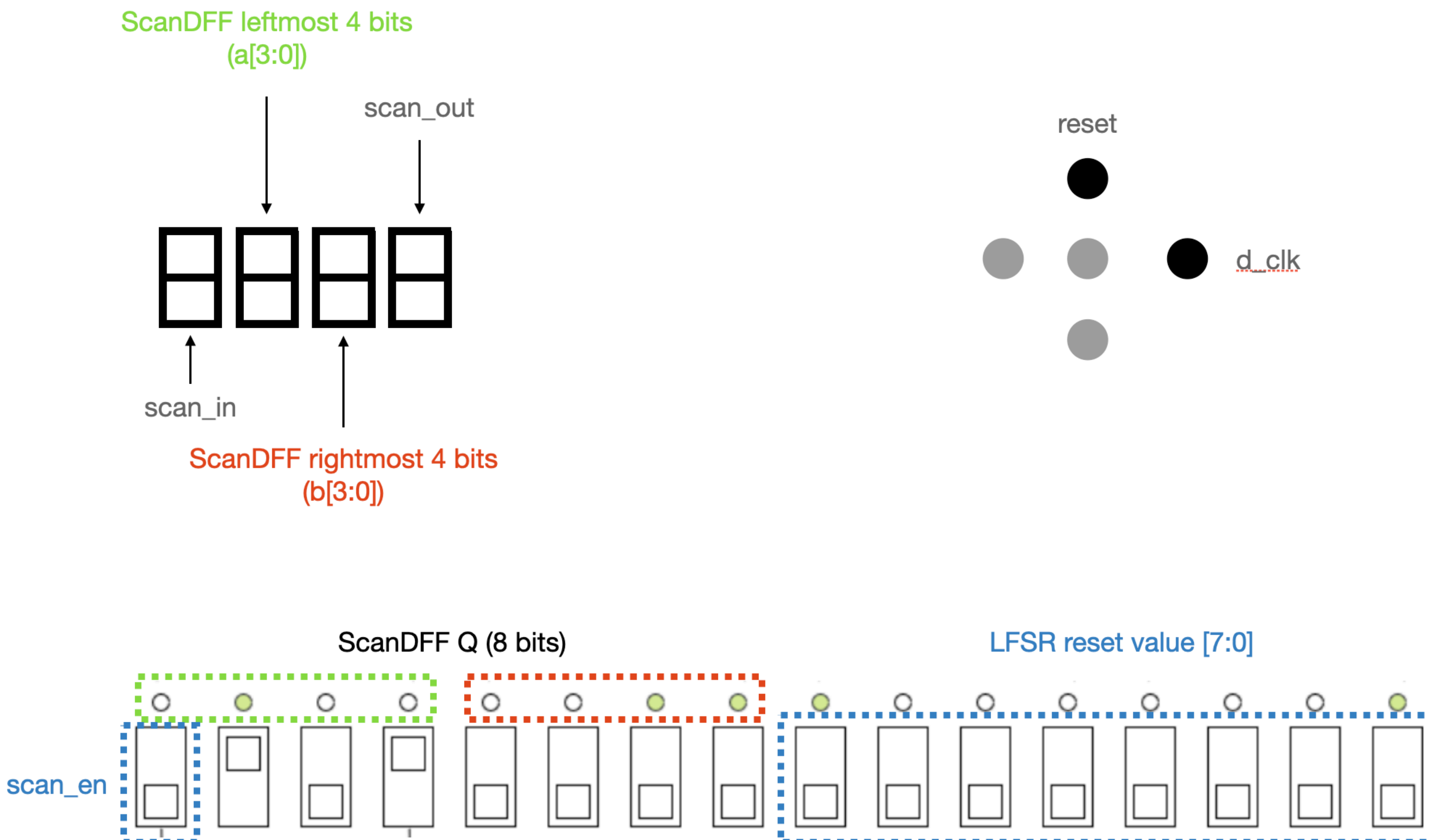
- Individual assignment

- Verilog questions
  - Source codes and the report due on 11/9/2023. 23:59:59.
    - Necessary: Content-addressable memory (CAM) design
    - Necessary: Scan chain design
    - Necessary: Built-in self test
    - Optional: Mealy machine sequence detector

- FPGA demonstration (due on 11/9/2023. In class.)
  - Necessary: Built-in self test

FPGA Demonstration

- Implementation of advanced question 3 on the FPGA
- The configuration of the switches, the buttons, seven segment display, and LEDs are as follows



FPGA Demonstration (con't)

- **For the sake of convenient observation, the FPGA question requires the Q signal from the Scan DFFs to be displayed on the LEDs:**
  - Please use the top button to reset the entire system.
  - Please use the leftmost switch as the scan_en signal.
  - Please use the rightmost eight switches to provide the reset values for the LFSR.
  - A total of 8 bits should be displayed on the leftmost 8 LEDs.
  - The 7-segment display also needs to show the signals of a[3:0] & b[3:0] (in hexadecimal).
  - To make it easy for the human eye to observe the output for each clock, the clock signals for the LFSR and Scan DFF will be triggered by the right button. (Pressing the right button once will generate one d_clk signal).

Thank you for your attention!

*The first Starbucks at Seattle, Washington, USA
This picture is taken by Chun-Yi Lee himself, who is also a fan of photography