Lab04
Synchronous FIFO

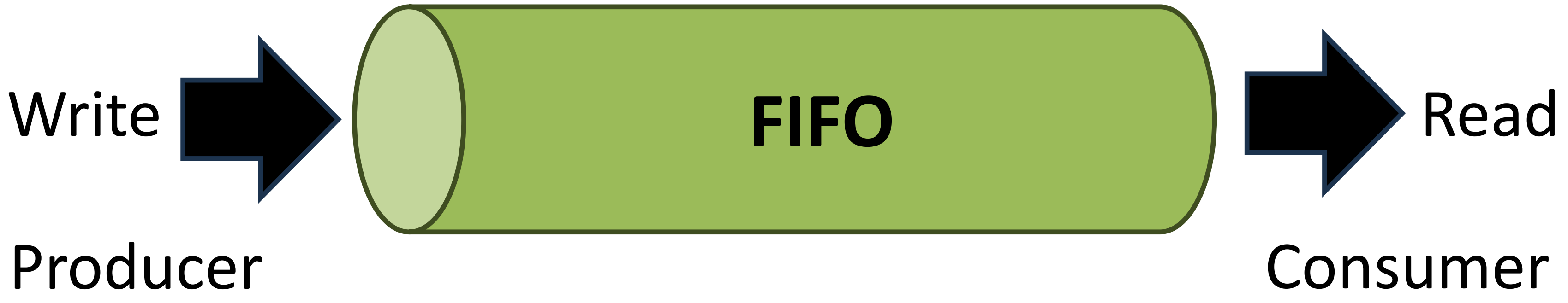
賴奕翔

Purpose

- Be familiar with Verilog syntax
  - Avoid multiple-driven
  - Separate sequential and combinational block
- Be familiar with circuit specification
  - Output data should be strictly **RESET** !!!
- Sequential circuit
- Synchronous FIFO

Synchronous FIFO

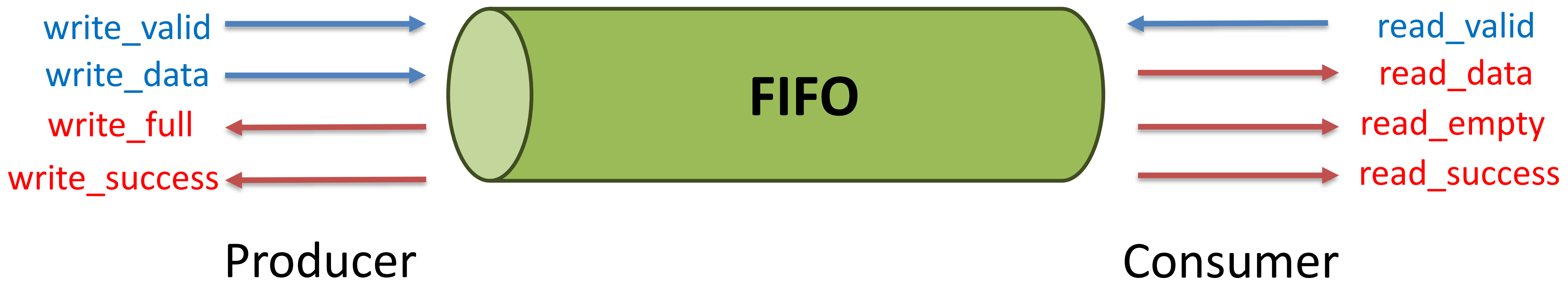
- A synchronous FIFO (First-In, First-Out) is a commonly used digital circuit for buffering and transferring data in sequential order
  - It is particularly useful when the producer and consumer operate at inconsistent processing rate
- "Synchronous" means that read and write operations are controlled by the same clock signal



Synchronous FIFO

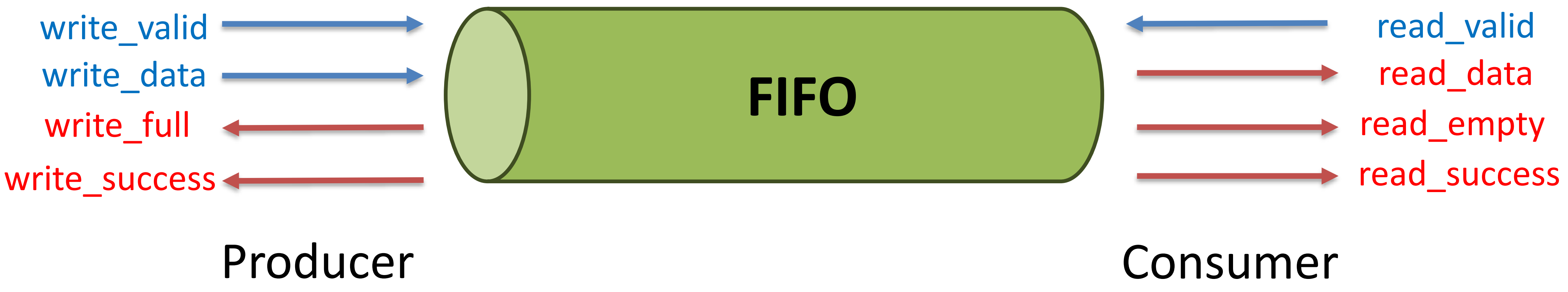
- In this lab, assume the FIFO has a depth of 10
  - The FIFO can buffer up to 10 data entries, each data entry is an 8-bit integer

Blue signal: circuit input
Red signal: circuit output



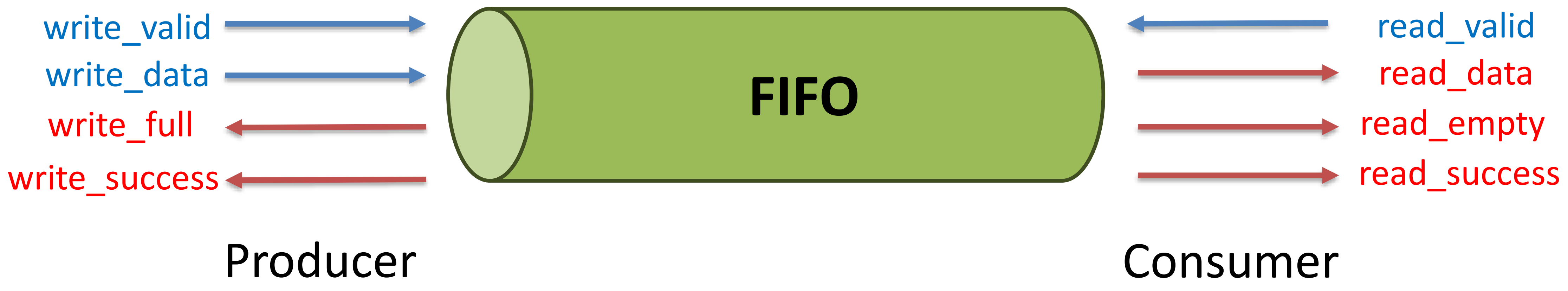
Synchronous FIFO

- Read operation: When "read_valid" signal is high
  - If the FIFO is empty, the output should be
    - read_success = 0, read_empty = 1, read_data = 0
  - Otherwise, the first data entry is removed from FIFO, and the output should be
    - read_success = 1, read_empty = 0, read_data = the first data in FIFO
- When "read_valid" signal is low, all read-related output should be reset to zero



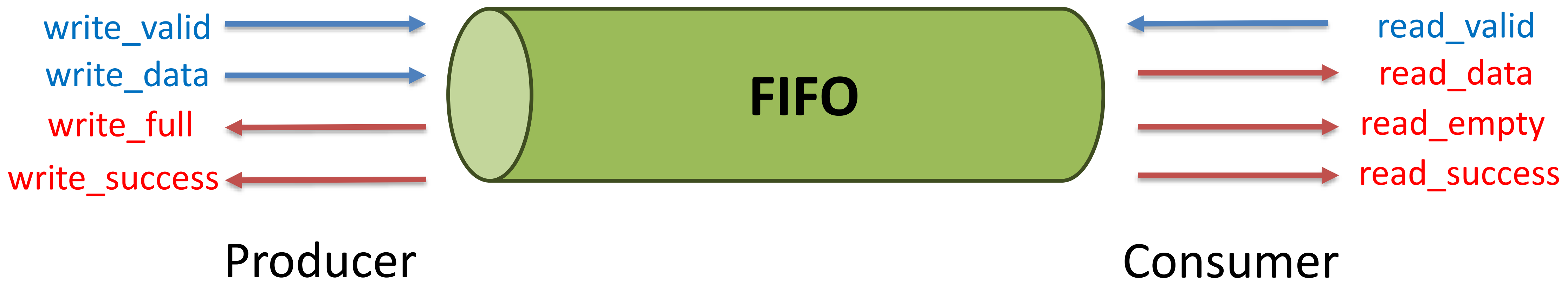
Synchronous FIFO

- Write operation: When “write_valid” signal is high
  - If the FIFO is full (it already contains 10 data entries), the output should be
    - write_success = 0, write_full = 1
  - Otherwise, the “write_data” is written into the FIFO, and the output should be
    - write_success = 1, write_full = 0
- When “write_valid” signal is low, all write-related output should be reset to zero

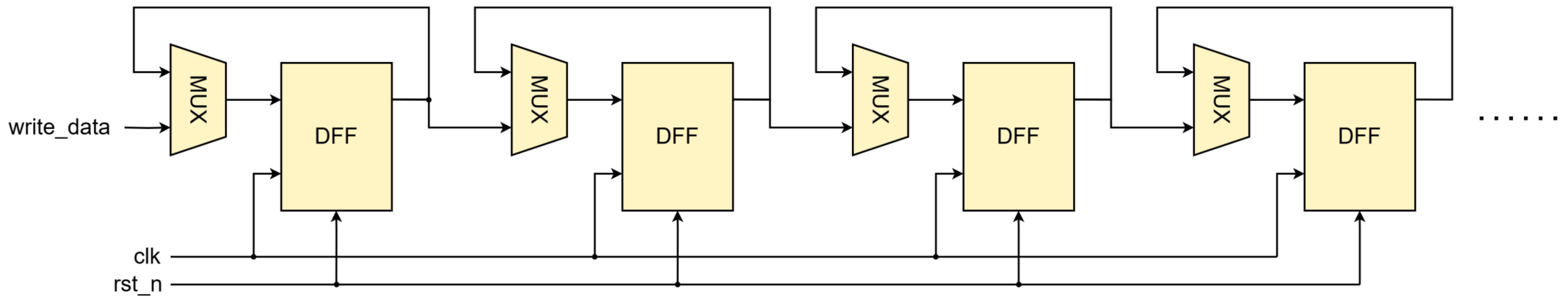


Synchronous FIFO

- When both “read_valid” and “write_valid” are high in the same cycle
  - The FIFO should perform the read operation first, then perform the write operation



Hint: Shift Register



Specification

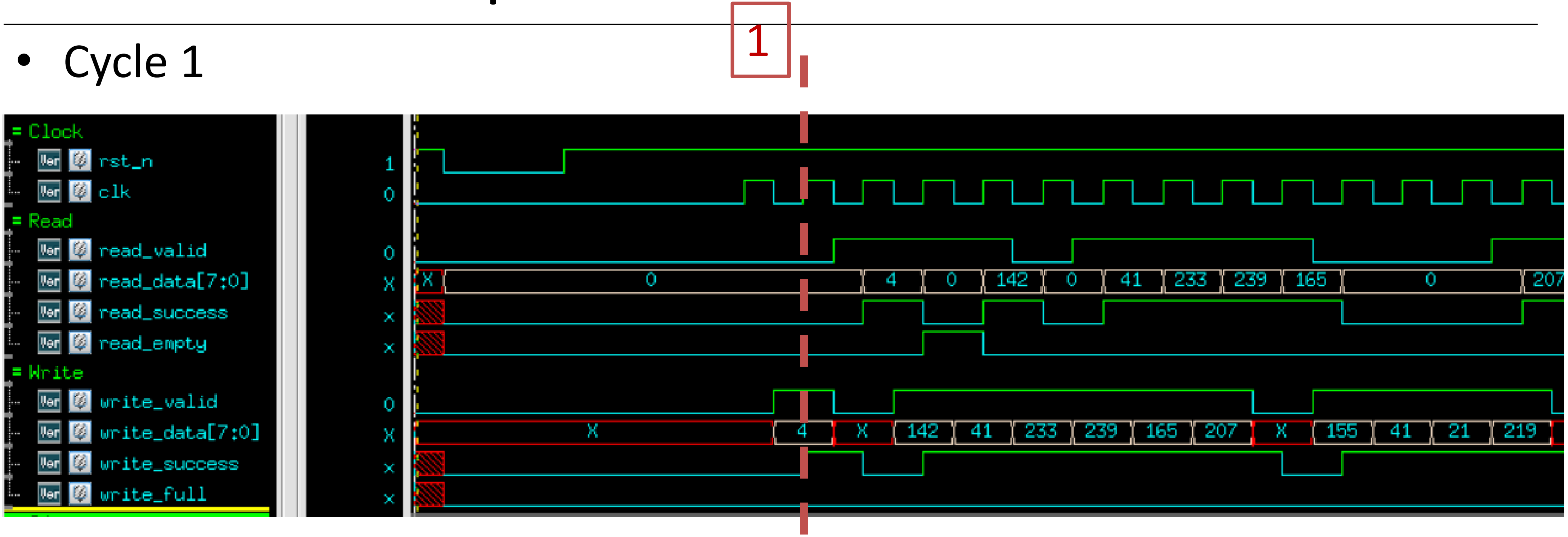
- Fifo.sv

Input Signal	Bit Width	Definition
clk	1	Clock
rst_n	1	Asynchronous active-low reset
read_valid	1	High when there is read request
write_valid	1	High when there is write request
write_data	8	The data to be written into the FIFO

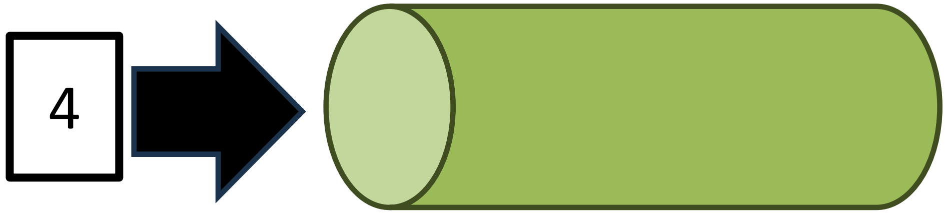
Output Signal	Bit Width	Definition
read_data	8	The first data read out from FIFO
read_success	1	High when read operation is successful
read_empty	1	High when empty FIFO is read
write_success	1	High when write operation is successful
write_full	1	High when full FIFO is written

Waveform Example

- Cycle 1

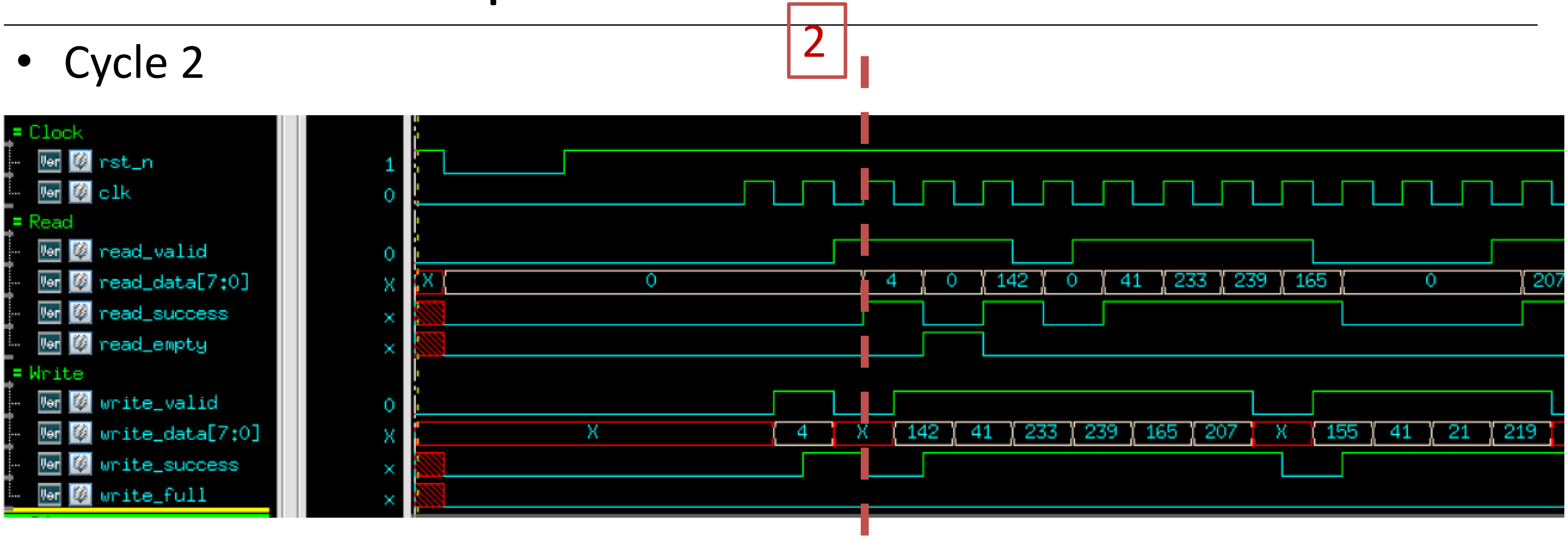


"write_valid" is high: write data 4 into the FIFO
"write_success" → 1
"write_full" → 0

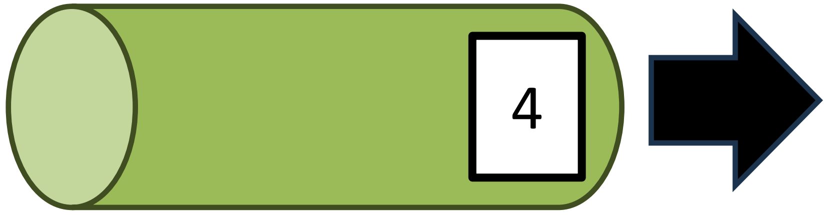


Waveform Example

- Cycle 2

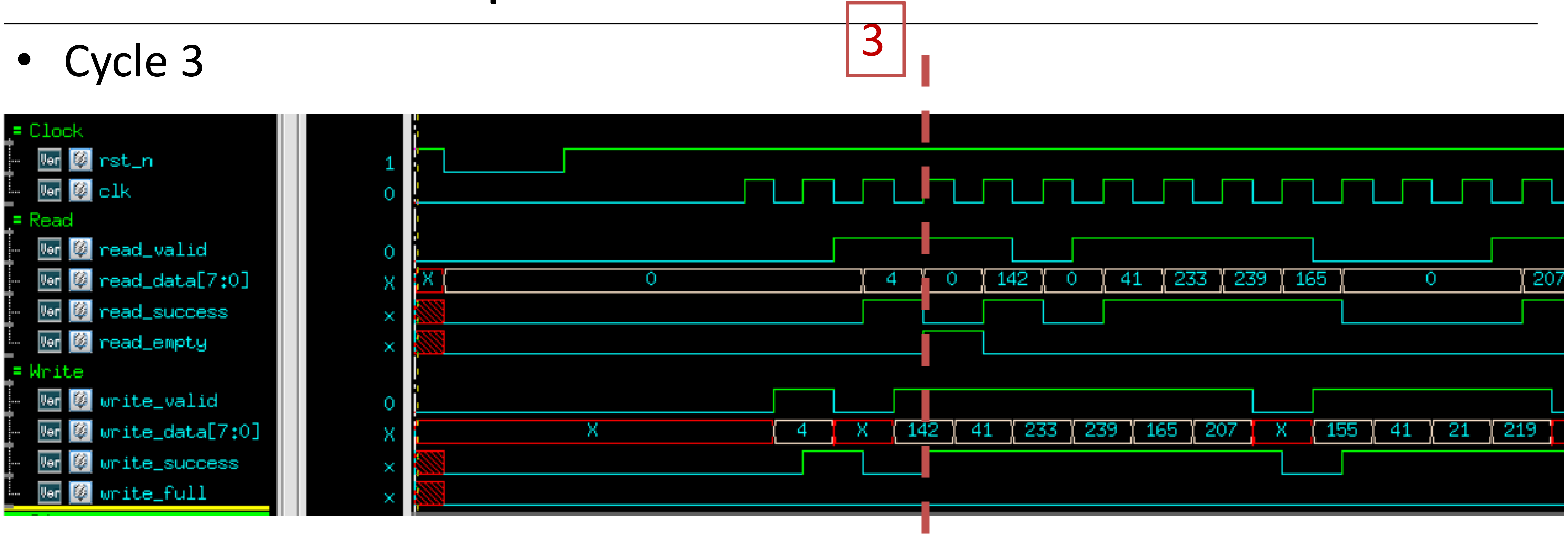


"read_valid" is high: read data out from FIFO
"read_data" → 4
"read_success" → 1
"read_empty" → 0



Waveform Example

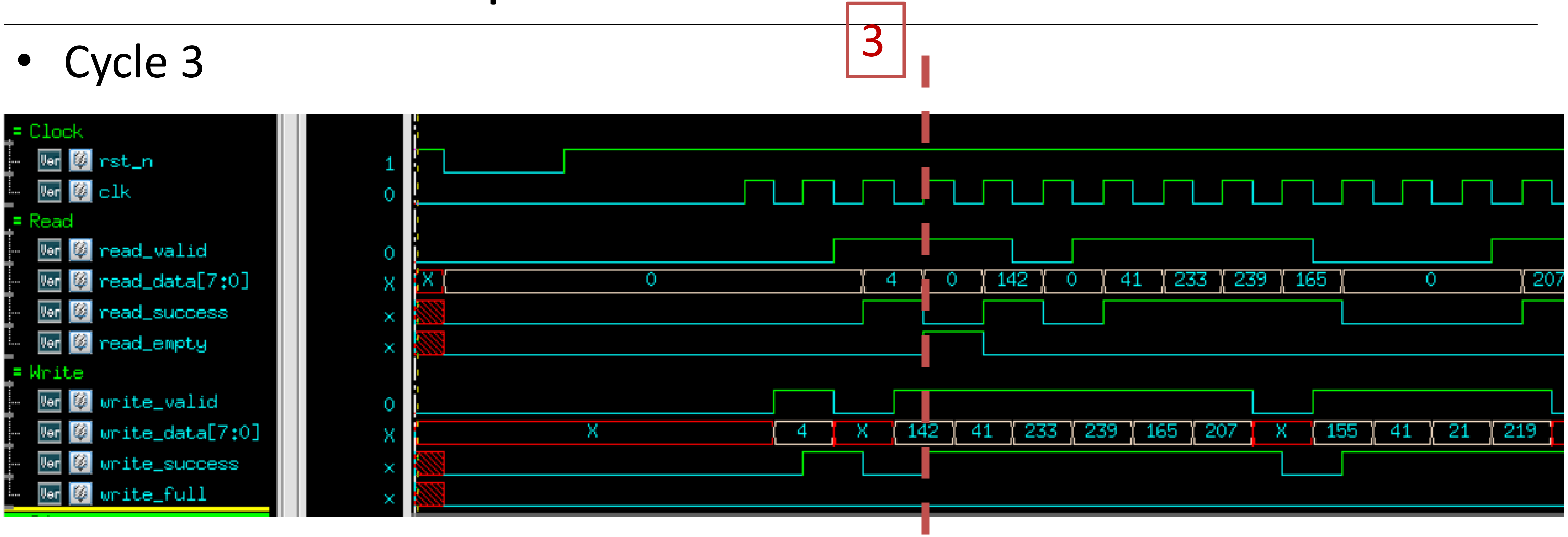
- Cycle 3



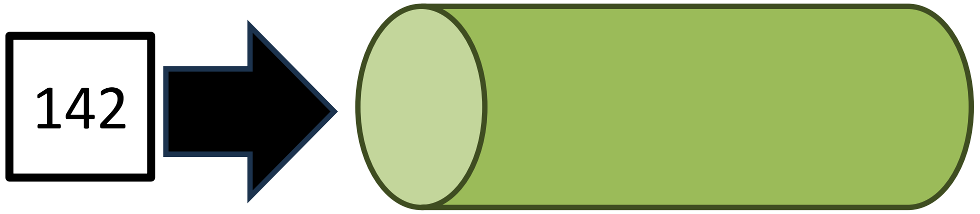
"read_valid" and "write_valid" are both high: read first
"read_data" → 0 (the FIFO is empty)
"read_success" → 0
"read_empty" → 1

Waveform Example

- Cycle 3

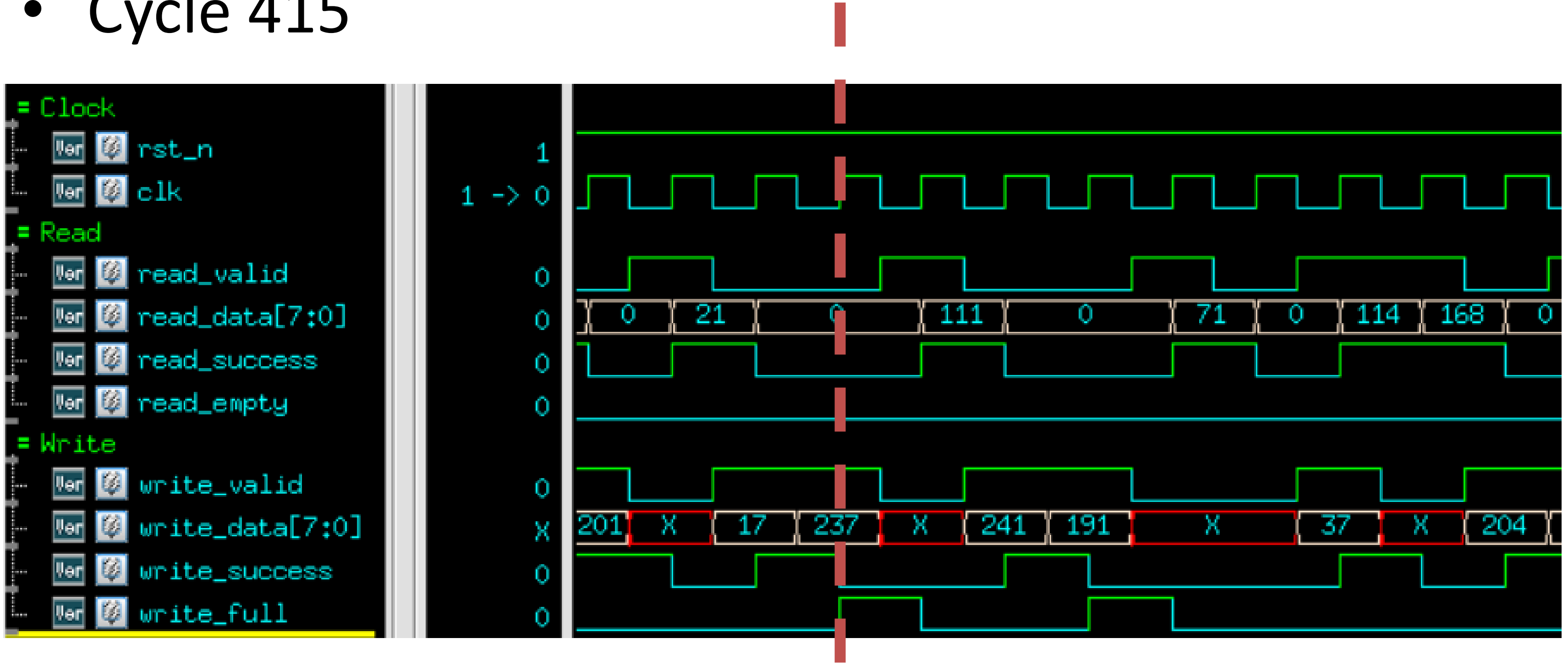


Then write data 142 into the FIFO
"write_success" → 1
"write_full" → 0

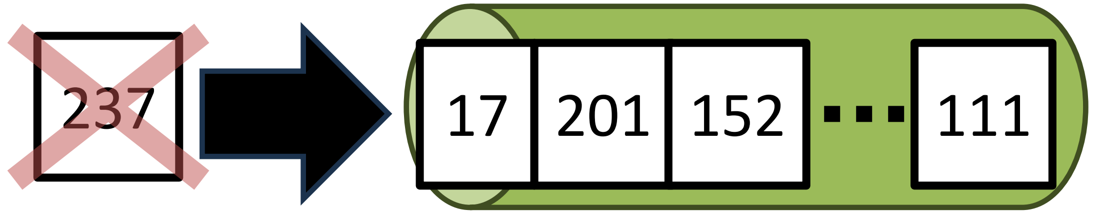


Waveform Example

- Cycle 415

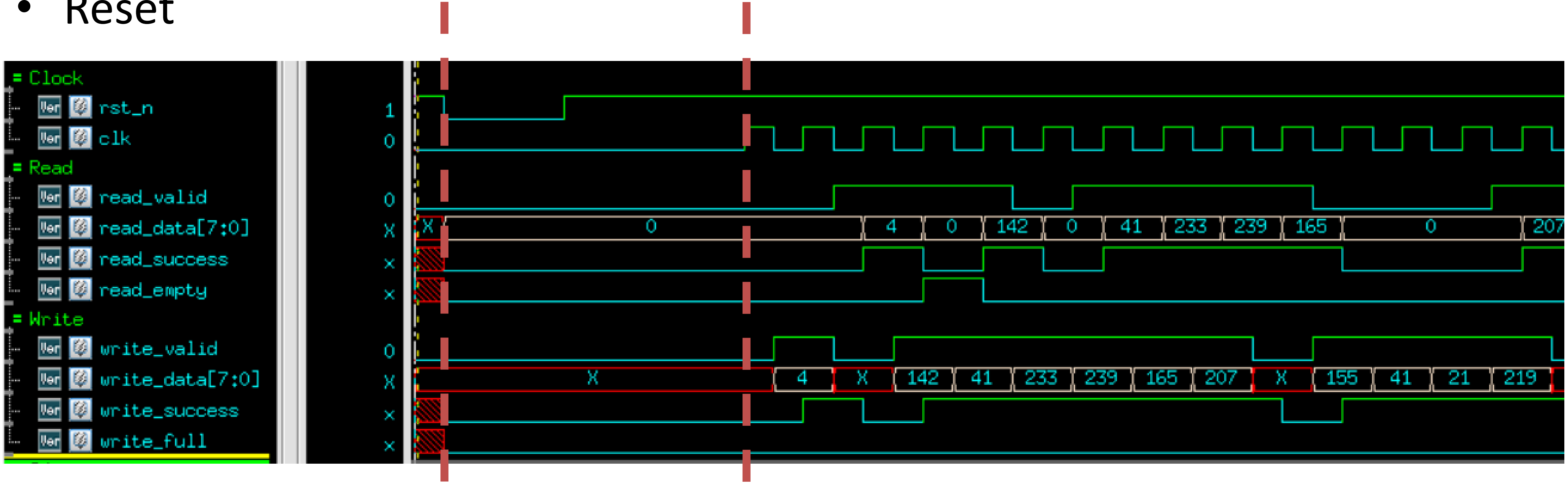


"write_valid" is high but the FIFO is already full:
"write_success" → 0
"write_full" → 1
and discard the written data 237



Waveform Example

- Reset



Note all output should be reset to 0 at the negative edge of rst_n
When “read_valid” is low, all read-related output should be reset to 0
When “write_valid” is low, all write_related output should be reset to 0

Directory

- 00_TESTBED
  - TESTBED.sv
  - PATTERN.sv
- 01_RTL
  - 01_run
  - 09_clean_up
  - Fifo.sv
- 02_SYN
  - 01_run_dc
  - 09_clean_up
- 03_GATE
  - 01_run
  - 09_clean_up

Command

- tar -xvf ~dcsTA01/Lab04.tar
- cd Lab04/01_RTL/

Grading Policy

- Pass the RTL & Synthesis & Gate-level simulation: 100%
  - 合成結果: (**不能有Error**、**Timing report slack met**、**不能有Latch**)
- Demo2 打7折

Upload

- 請將Lab04/01_RTL裡的Fifo.sv依以下命名規則重新命名後上傳至E3
- 命名規則：Fifo_dcsxxx.sv，xxx為工作站帳號號碼
- **命名錯誤扣5分!!!**
- **Deadline:**
  - **Demo 1: 3/20 17:25**
  - **Demo 2: 3/20 23:59**