CS4125
Design Pattern:
A FIFO Module

黃稚存

國立清華大學
資訊工程學系

Lecture 11

An FSM Coding Practice for a FIFO Design

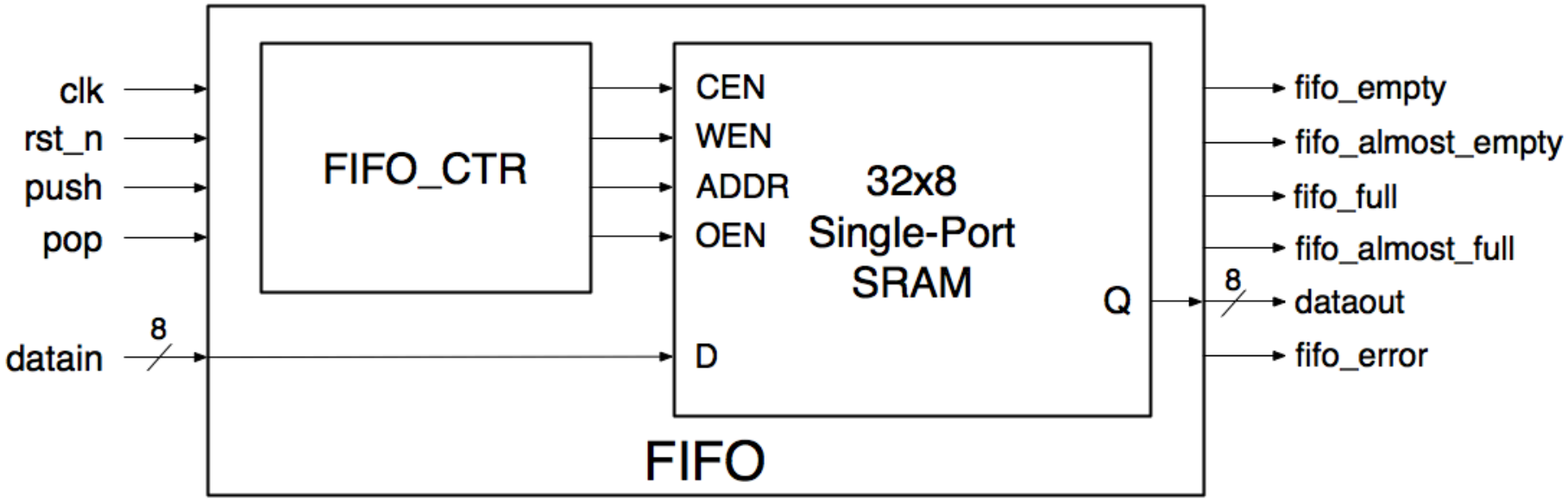
- Using the memory generator for simulation (and for synthesis later)
- A coding practice for an FSM

- Design target: FIFO
  - Start with spec and block diagram

Specification

- Implement a FIFO design with 32 entries
  - Each has 8 bits
- The block diagram with its primary inputs and outputs
  - 8-bit data in
  - 8-bit data out
  - What else?

Block Diagram



Primary IOs and Block-Level Interconnection

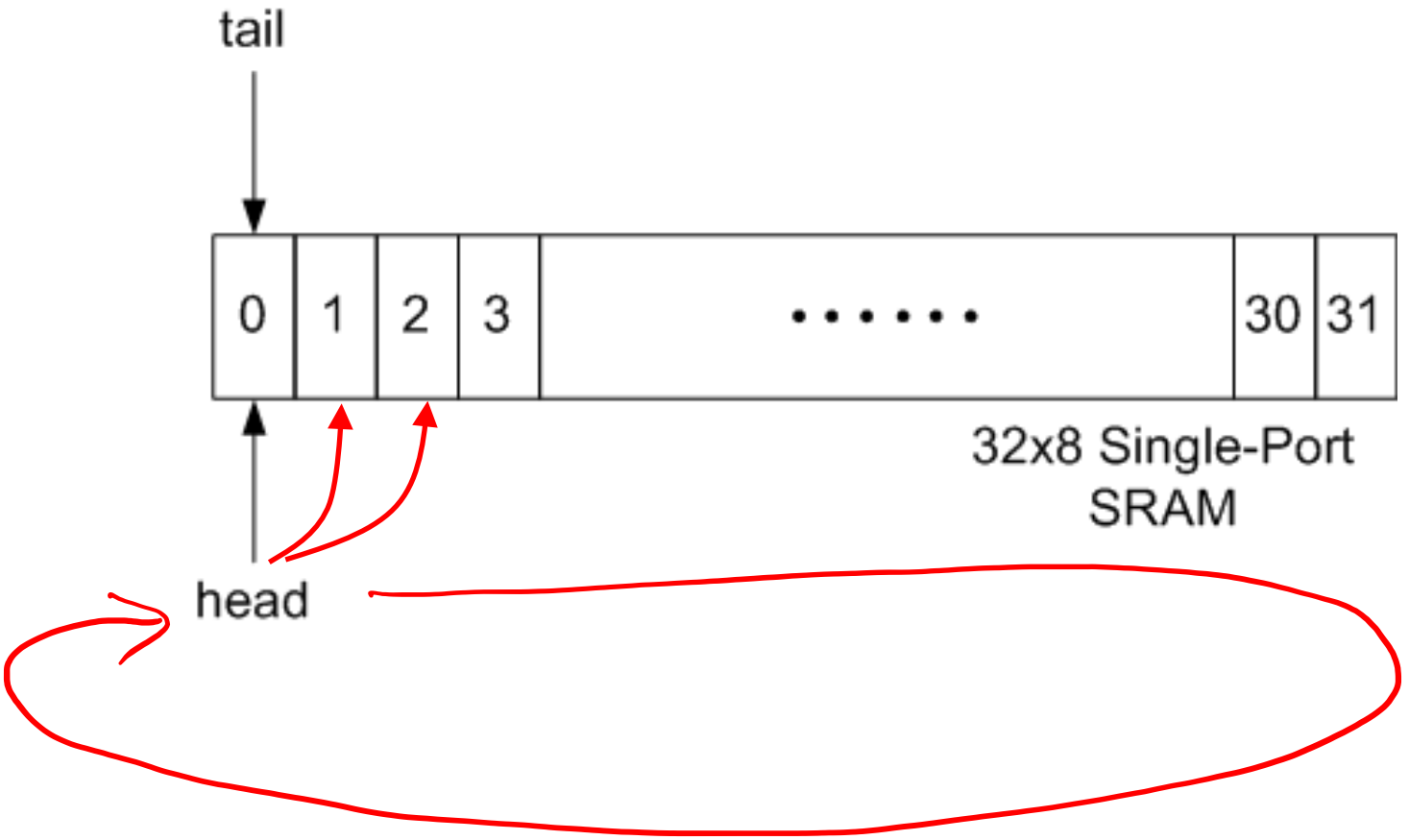
- Inputs:
  - `clk`: clock
  - `rst_n`: negative reset
  - `push`: a control signal to push a data
  - `pop`: a control signal to pop a data
  - `datain`: 8-bit input data
- Outputs:
  - `dataout`: 8-bit output data
  - `fifo_empty`: a flag to indicate that the FIFO is empty
  - `fifo_almost_empty`:
    a flag to indicate the FIFO will be empty with one more pop
  - `fifo_full`: a flag to indicate the FIFO is full
  - `fifo_almost_full`:
    a flag to indicate the FIFO will be full with one more push
  - `fifo_error`: a flag to indicate error

Primary IOs and Block-Level Interconnection

- Internal signals:
  - head, tail:
    There are two pointers, head and tail, to indicate the beginning and end of the FIFO, respectively
  - Inputs to SRAM:
    - CEN, WEN, OEN, ADDR, D
  - Outputs from SRAM:
    - Q

Two Pointers in FIFO

- head and tail are reset to zero



FIFO Controller
States and Input Control Signals

- FSM with 4 states
  - EMPTY, BETWEEN, FULL and READOUT
- Major operations of FIFO
  - Push
    - `RAM[head] = datain`
    - `head = (head + 1) % size_of_ram`
  - Pop
    - `dataout = RAM[tail]`
    - `tail = (tail + 1) % size_of_ram`
- Four possible input combinations (actions)
  - `do_idle:` `push == 0 && pop == 0`
  - `do_pop:` `push == 0 && pop == 1`
  - `do_push:` `push == 1 && pop == 0`
  - `do_push_pop:` `push == 1 && pop == 1`

FIFO Controller
Status Flags

- Two status flags to affect the state transition (why?)
  - fifo_almost_empty: `tail == head - 1`
  - fifo_almost_full: `head == tail - 1`
- Some flags determined by states
  - `fifo_empty = 1` at EMPTY state
  - `fifo_full = 1` at FULL state
  - `oen = 0` at READOUT state
    - Or you can keep it low constantly

FIFO Controller
Error Flags

- `fifo_error = 1`
  - Whenever both push and pop are activated at the same time
  - When pop is activated at EMPTY state
  - When push is activated at FULL state
  - When push or pop is activated at READOUT state

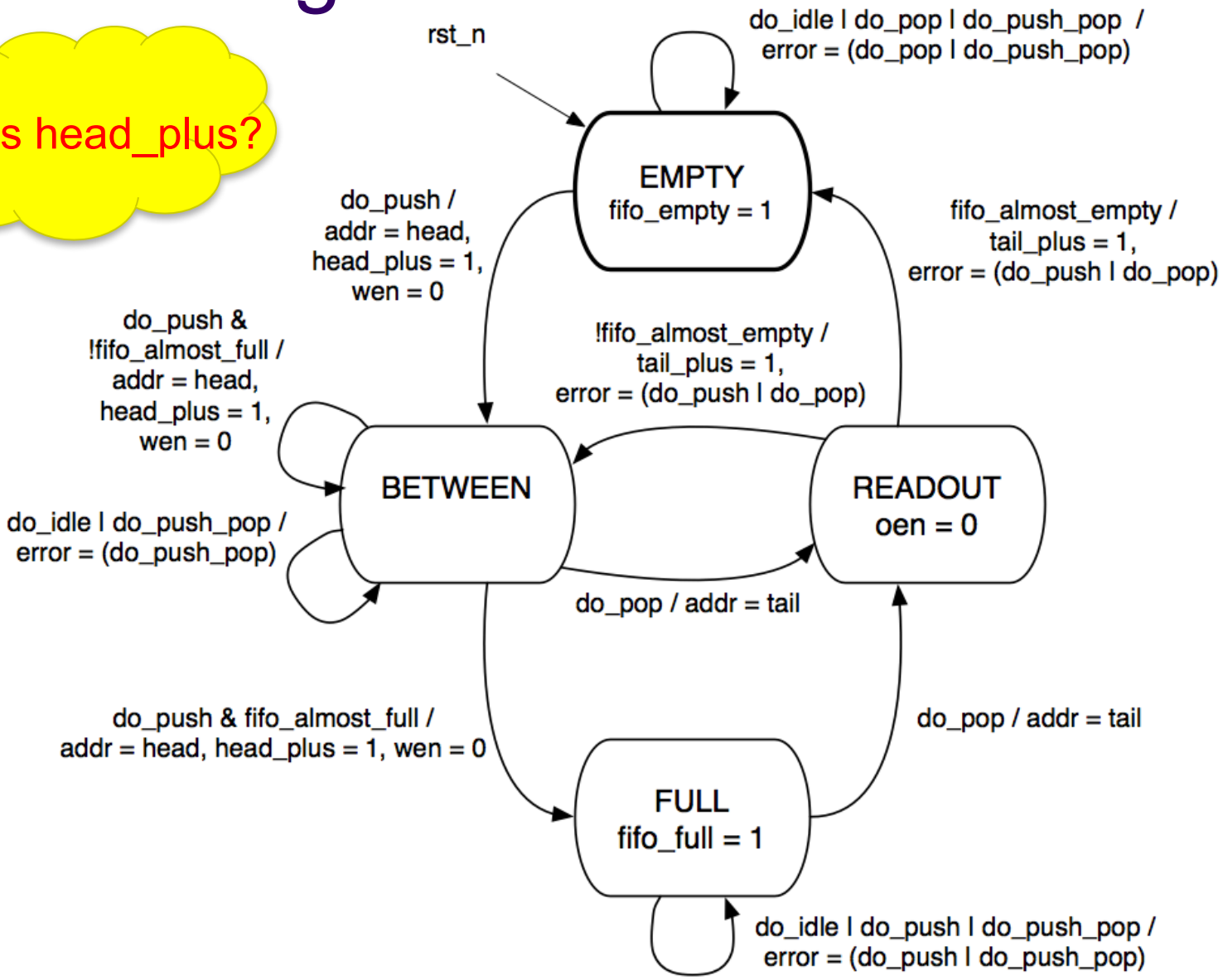
FIFO Controller Execution

- Pushing (write) takes one clock cycle
- Popping (read) takes two clock cycles (why?)

- The state diagram is the spec
  - Is the state diagram (in the next slide) complete?
  - When in doubt, verify the state diagram and modify the RTL code accordingly

State Diagram

What is head_plus?



Note and Hint:
Generating Flags with Head/Tail Pointers

- Potentially Buggy Verilog code
  (tail = 4'b1111, head = 4'b0000)

```
always @(tail or head) begin
  if (tail == head - 1) begin
    fifo_almost_empty = 1;
  end else begin
    fifo_almost_empty = 0;
  end
  ……
end
```

always @* begin

tail == head – 1’b1

Note and Hint:
Try to symbolize every condition

```
always @(head_plus or head) begin
  if (head_plus == 1'b1) begin
    head_next = head + 1'b1;
  end else begin
    head_next = head;
  end
end
```

Guideline of Verilog Design (1/2)

- Prepare the block diagram and FSM well
  - Before your Verilog coding
- FSM: the simpler the better
  - Try to symbolize every condition for state transition
  - One single state can take multiple cycles
  - States can be nested

Guideline of Verilog Design (2/2)

- There is no best style for every design
  - State arrangement
  - Naming convention
    - With `fifo_` or without `fifo_` (`do_`)
- This FIFO design can be improved
  - E.g., with pipelined architecture, etc.
- The order of always (or assign) blocks does not matter

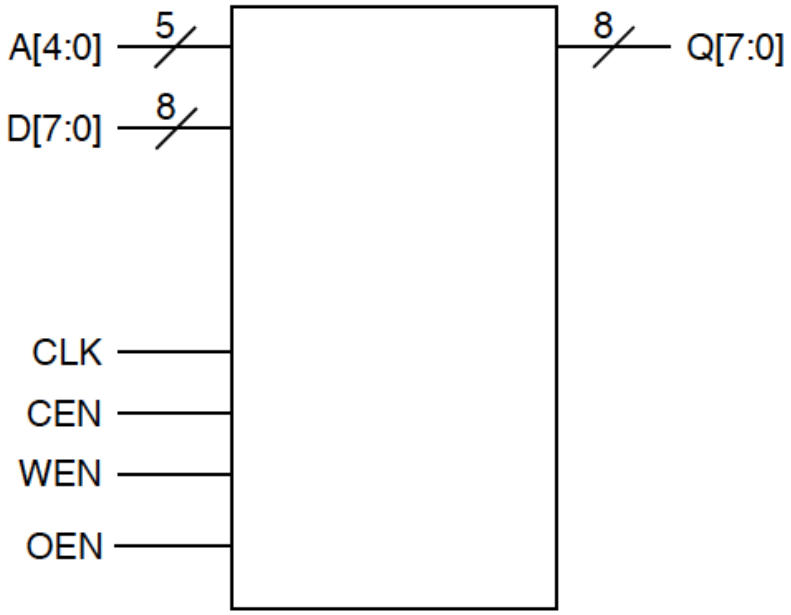
Memory Block: Description

- Single-port synchronous SRAM
- Behavior model: `RAM32x8.v`
  - Simulation only
  - Non-synthesizable
  - Black box

Memory Block: IO Interface

Pin Description

Pin	Description
A[4:0]	Addresses (A[0] = LSB)
D[7:0]	Data Inputs (D[0] = LSB)
CLK	Clock Input
CEN	Chip Enable
WEN	Write Enable
OEN	Output Enable
Q[7:0]	Data Outputs (Q[0] = LSB)



Assignment

- Complete your own FIFO module
- There is a design template
  - `hw03_fifo2017.tgz`
- How to extract the tar-gzipped file
  `$ tar zxvf hw03_fifo2017.tgz`
- Refer to `00_README.txt` carefully
  - File list with short description
  - How to avoid timing violation in RTL simulation?
  - How to synthesis with SRAM block?

FIFO Design Part 1

- Complete the 32x8 FIFO design with the given memory model
- You should improve the stimulus and verify the design as complete as you can
- Use nWave's debug features, e.g., alias table to help the signal debugging of state vector and control signals
- Try to use as many as previous simulation techniques we mentioned

FIFO Design Part 2

- Extend the FIFO to 64x16
  - Using the 32x8 memory as a basic block
  - You cannot modify RAM32x8.v
  - Instead, you may use multiple (hint: four) 32x8 memory blocks
- Prepare proper test patterns to validate your design
- Perform the synthesis

Write a Summary Report

- Show that you can or can not access
  - Every state
  - Every state transition
  - Every condition to trigger a transition
- Improve your testbench and try to maximize the accessibility
- Discussion: can you improve the latency of popping (e.g., with the pipeline of consecutive reads)?
- Discussion: how can you modify the FIFO to a Stack (i.e., first-in last-out)?

A Few More Note for You

- Many of you ask for a "good" RTL coding sample.
  - Well, this is a good design example instead, with an explicit finite state machine and its datapath.
- So please treat it not only a simple homework assignment but also a design pattern.
  - Take a careful study in the design structure and style.
- Let me know if you encounter problem.
- Have fun with your designing!