CSED311 Lab1: RTL Design

Hyunuk Cho

gusdnr9779@postech.ac.kr
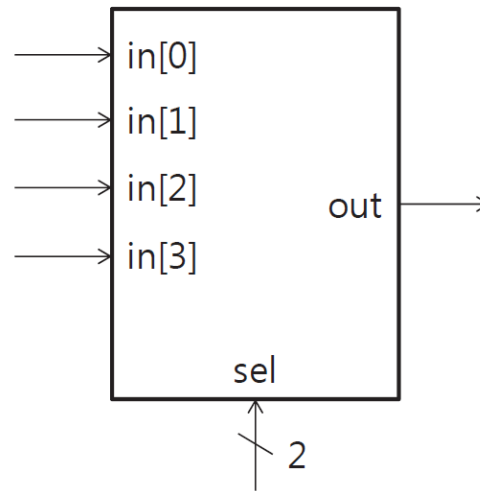
Contact the TAs at csed311-ta@postech.ac.kr

POSTECH

Contents

- Combinational logic
- Register-Transfer Level
- Finite State Machine
- Assignments
  - Combinational logic (ALU)
  - FSM (vending machine)

POSTECH

Combinational logic

- Represents Boolean function (time(clk)-independent)
- Output is a function of inputs only
- `output = f(input)`

- `Example: 4-to-1 mux`

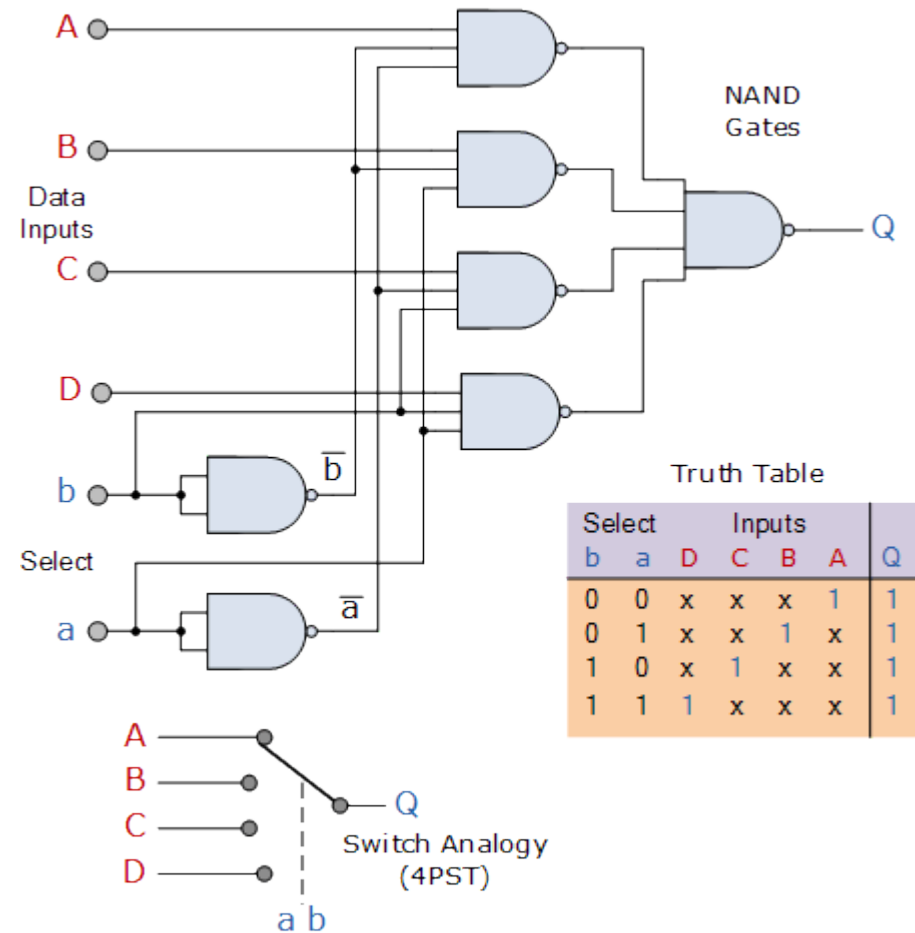| [3:0]in	sel	out
2'bxxx0	2'b00	0
2'bxxx1	2'b00	1
2'bxx0x	2'b01	0
2'bxx1x	2'b01	1
2'bx0xx	2'b10	0
2'bx1xx	2'b10	1
2'b0xxx	2'b11	0
2'b1xxx	2'b11	1

Combinational logic

■ Implementation with Verilog (mux)
— If you use "always", simply giving "(*)" as the sensitivity list is equivalent to the code below

Using Assign	Using Always
| ```verilog
module mux(in, sel, out);
    input [3:0] in;
    input [1:0] sel;
    output out;

    assign out = (sel == 2'b00) ? in[0] :
            (sel == 2'b01) ? in[1] :
            (sel == 2'b10) ? in[2] : in[3];
endmodule
``` | ```verilog
module mux(in, sel, out);
    input [3:0] in;
    input [1:0] sel;
    output out;
    reg out;

    always @ (sel or in) begin
        if (sel == 2'b00)
            out  = in[0];
        else if (sel == 2'b01)
            out  = in[1];
        else if (sel == 2'b10)
            out  = in[2];
        else
            out  = in[3];
    end
endmodule
``` |

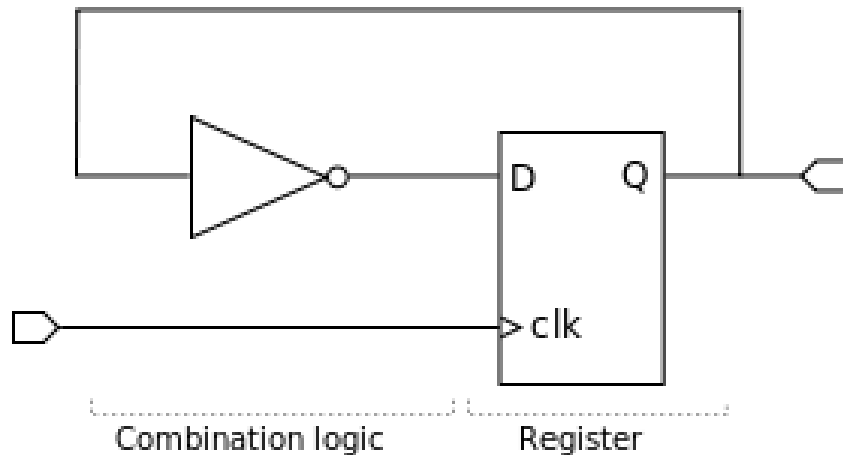# Combinational logic

■ RTL schematic

Register-Transfer Level

■ The design method to implement a **synchronous circuit** with an HDL (Hardware Description Language)

■ Synchronous circuit consists of:

— **Register**: a memory element synchronized by the clock signal

— **Combinational logic**: logical function

RTL - example

■ Toggler example

<Fig 1. "Register transfer level - example toggler" (Register-transfer level, Wikipedia, https://en.wikipedia.org/wiki/Register-transfer_level)>

```verilog
module toggler(input clk, output out);
    reg state;
    wire comb;

    assign comb = ~state;
    assign out = state;

    initial begin
        state <= 0;
    end

    always @(posedge clk) begin
        state <= comb;
    end
endmodule
```
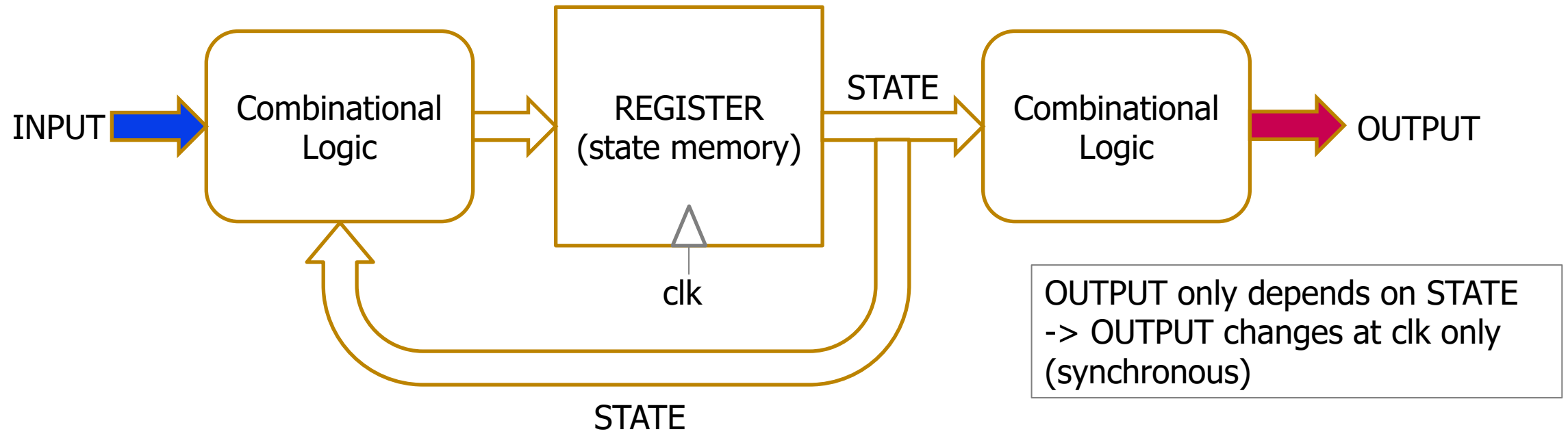
Finite State Machine (FSM)

- Moore Machine

- Mealy Machine

POSTECH

Finite State Machine (FSM)

- Moore Machine
  - Outputs only depend on the current state.

INPUT → Combinational Logic → REGISTER (state memory) → STATE → Combinational Logic → OUTPUT

clk

STATE

OUTPUT only depends on STATE -> OUTPUT changes at clk only (synchronous)

Finite State Machine (FSM)

- Mealy Machine
  - Outputs depend on the current state and the current inputs

OUTPUT depends on STATE and INPUT
-> OUTPUT can be changed when INPUT changes, even without clock tick (i.e., asynchronously)
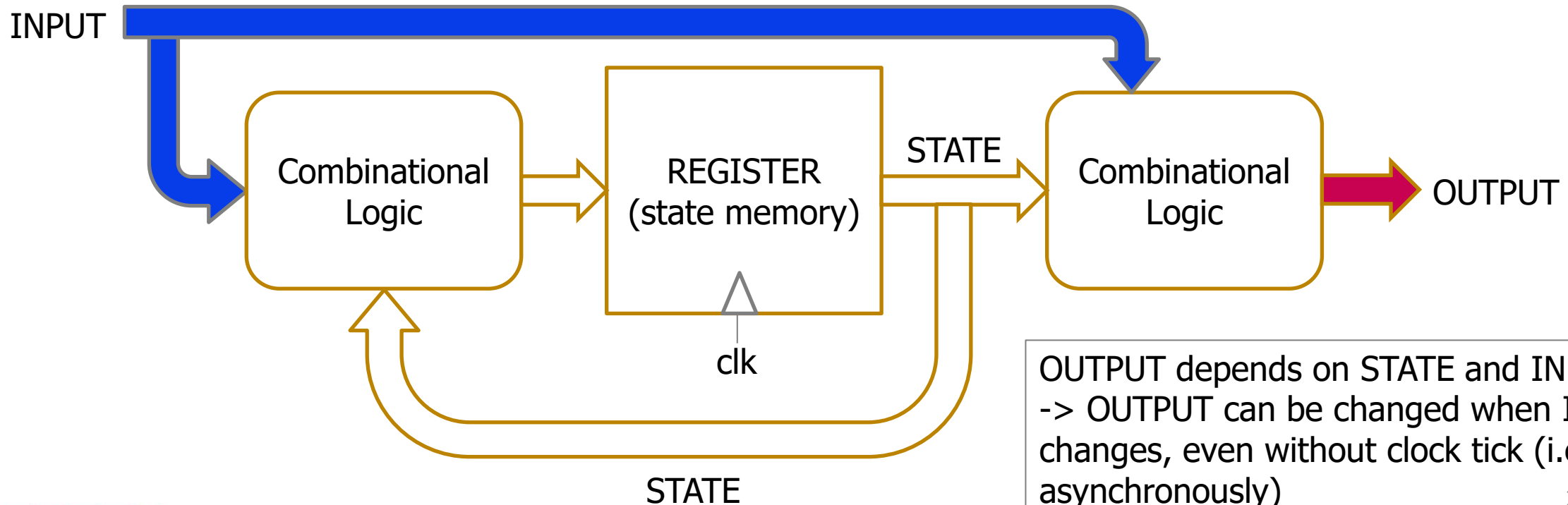
Finite State Machine (FSM)

- Mealy Machine
  - Mealy Machine can be made synchronous

FSM Code Structure

ports and variables declaration
Combinational logic
Sequential logic

Combinational logic derives the next state and outputs
- assign statement + wire data type
- always @(...) block + reg data type (assignment)

Sequential logic stores the current state
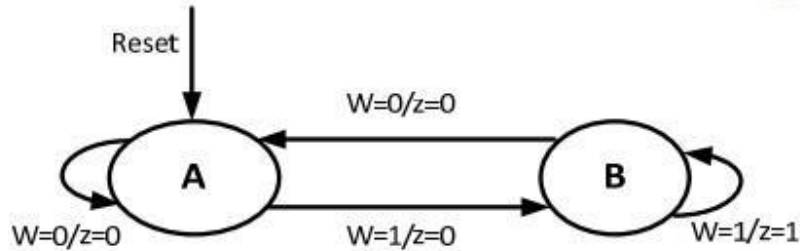- always @(posedge clock) block + reg data type (assignment)
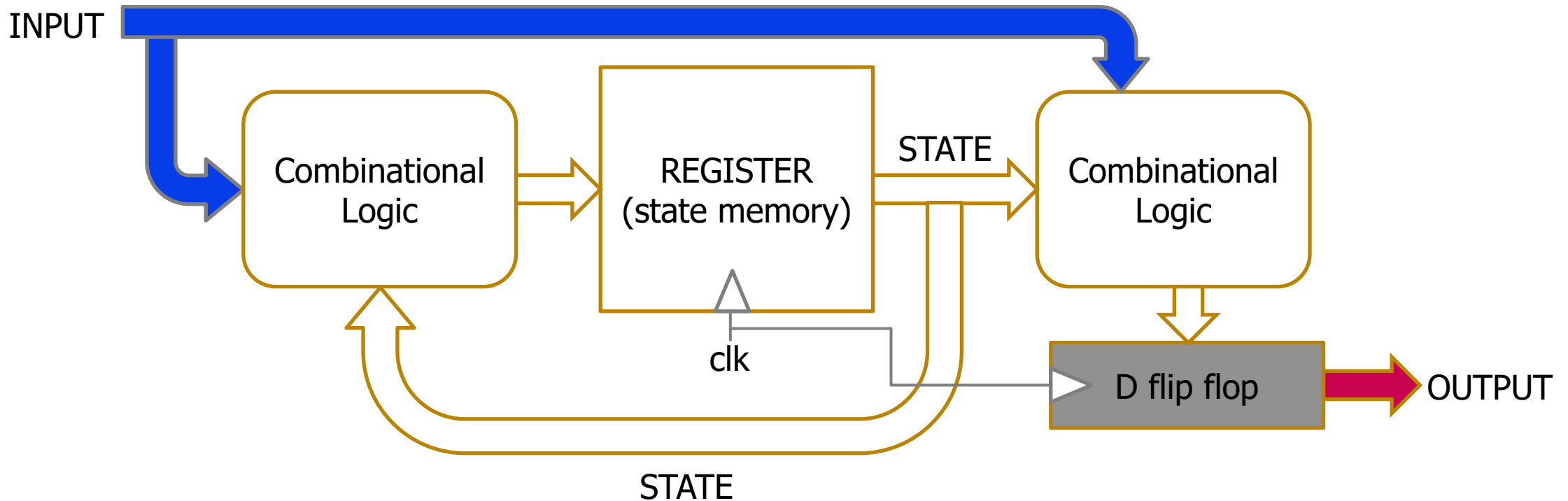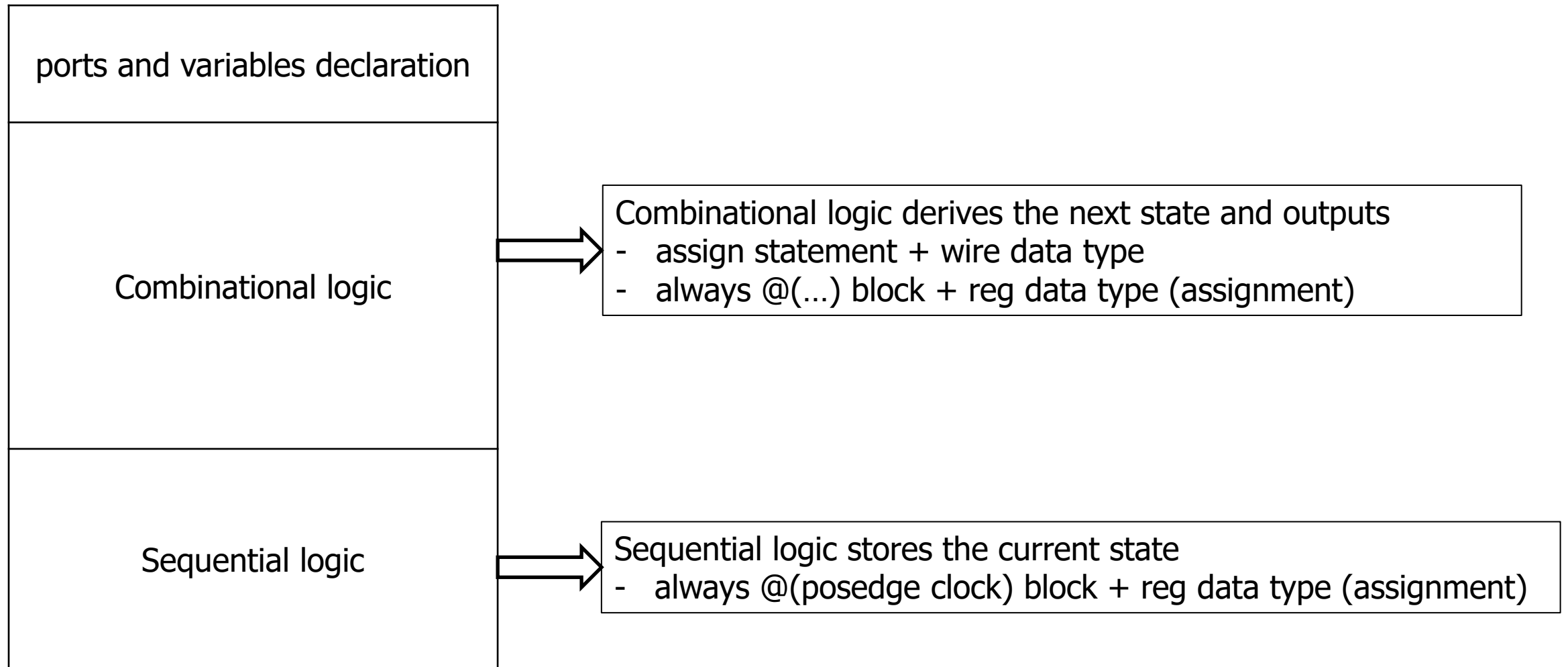
FSM Code Structure

❖ Example: Mealy Machine

Output: reg ⟹

Example: Reduce 1's
Given a sequence of 0s and 1s, *change the first 1 of consecutive 1s into 0*

Examples:
000011100 -> 000001100
010101010 -> 000000000
011001100 -> 001000100

z is output and z is in combinational logic part
-> asynchronous Mealy Machine

```
module mealy (Clock, w, Resetn, z);
input Clock, w, Resetn ;
output reg z ;
reg CS, NS;
parameter A = 1'b0, B = 1'b1;
always @(w, CS)
  case (CS)
    A:  if (w == 0)
          begin
            NS = A;   z = 0;
          end
        else
          begin
            NS = B;   z = 0;
          end
    B:  if (w == 0)
          begin
            NS = A;   z = 0;
          end
        else
          begin
            NS = B;   z = 1;
          end
  endcase
always @(posedge Clock, negedge Resetn)
  if (Resetn == 0)
    CS <= A;
  else
    CS <= NS;
endmodule
```

NS & Output Calculation

CS Calculation

Combinational (Blocking)

Sequential (Non-Blocking)

POSTECH

13

FSM Code Structure
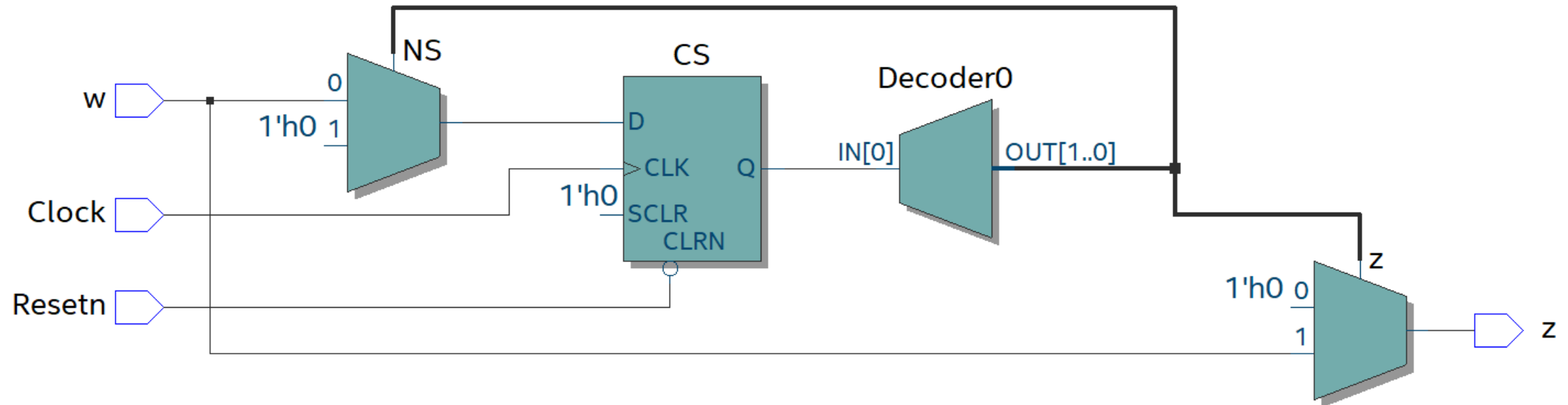
Assignment 1.a

- Implement ALU (Arithmetic Logic Unit) in Verilog
- A skeleton code and testbench will be provided
  - Lab1/ALU/*

- Input: (A, B, FuncCode)
  - A: left operand (16-bit signed binary)
  - B: right operand (16-bit signed binary)
  - FuncCode: operator (4-bit binary)

- Output: (C, OverflowFlag)
  - C: operation result (16-bit signed binary)
  - OverflowFlag: overflow flag (1-bit binary)

POSTECH

Assignment 1.a (cont'd)

- ALU operations

FuncCode	Operation	Comment	FuncCode	Operation	Comment
0000	A + B	Signed Addition	1000	A ⊕ B	Bitwise XOR
0001	A − B	Signed Subtraction	1001	~(A ⊕ B)	Bitwise XNOR
0010	A	Identity	1010	A << 1	Logical Left Shift
0011	~A	Bitwise NOT	1011	A >> 1	Logical Right Shift
0100	A & B	Bitwise AND	1100	A <<< 1	Arithmetic Left Shift
0101	A | B	Bitwise OR	1101	A >>> 1	Arithmetic Right Shift
0110	~(A & B)	Bitwise NAND	1110	~A + 1	Two's Complement
0111	~(A | B)	Bitwise NOR	1111	0	Zero

Assignment 1.a (cont'd)

- Overflow detection
- For addition and subtraction, you should detect overflow and set the flag.

Overflow Flag	Singed Addition	Signed Subtraction
0	Correct Result	Correct Result
1	**Wrong Result**	**Wrong Result**

- For other operations, OverflowFlag is always zero.

Assignment 1.b

- Implement a simple vending machine RTL in Verilog

  - A simple Finite State Machine (FSM)

- Your vending machine should cover all use-cases (in the next slides)

- A skeleton code and testbench will be provided

  - Lab1/vending_machine/*

Assignment 1.b (cont'd)

- Vending machine interface

INPUT		
Signal	**Description**	**Number of bit(s)**
i_input_coin	Insert Coin	1 for each type of coins
i_select_item	Select Item	1 for each type of items
i_return_trigger	Return change	1
clk	clock	1
reset_n	reset	1

OUTPUT		
Signal	**Description**	**Number of bit(s)**
o_output_item	Indicate dispensed items	1 for each type of items
o_available_item	Indicate item availability	1 for each type of item
o_return_coin	Indicate type of coin (change)	1 for each type of coins

POSTECH

Assignment 1.b (cont'd)

■ Vending machine use-case

Assumption: infinite item, change

Sequence
1. Insert money (available money unit: 100, 500, 1000 won) and initialize waiting time(=100)
2. Vending machine shows all available items where (item cost <= current money)
  2.a. decrease waiting time(-1)

3.a. Insert money within the waiting time
  3.a.1. Go to 2 and initialize waiting time(=100)
3.b. Select an item within the waiting time
  3.b.a. Case1: the item is available
    3.b.a.1. The item is dispensed
    3.b.a.2. Go to 2 and initialize waiting time(=100)
  3.b.b. Case 2: the item is unavailable
    3.b.b.1. Nothing happens. Waiting time is not reset.
3.c. No input within the waiting time
  3.c.1 Return changes

a*. Whenever press the return button
  a*.1. Return changes
  a*.2. Go to 1

POSTECH

Assignment 1.b (cont'd)

- If your modelsim is not displaying the output of $display or $monitor, then try using vending_machine_tb_f.v. (output will be written to output.txt)

- If your simulation keeps running, the testbench module is waiting for the output signals. (o_output_item or o_return_coin)

- If your vending machine code passes all tests, you will get the message "Passed = 23, Failed = 0"

POSTECH

Evaluation Criteria (Source code)

- ALU
  - Number of tests passed from the shared TB

- Vending machine
  - Number of tests passed from the shared TB as well as additional TB not shared
  - Modularization
    - Your implementation should include at least two modules
    - Using modules lowers implementation complexity
      - Similar to the functions in other programming languages
    - It would be helpful to practice before implementing CPUs
  - Understanding of combinational/sequential logic
    - E.g.) Appropriate use of blocking/non blocking assignment

POSTECH

Evaluation Criteria (Report)

- You can write report in Korean or English

- Report should include (1) introduction, (2) design, (3) implementation, (4) discussion, and (5) conclusion

- ALU
  — Report not evaluated, source code only

- Vending machine
  — Combinational logic design and implementation
  — Sequential logic design and implementation
  — FSM type of your design
    • Moore or mealy machine

POSTECH

Assignment tips

#1: When modeling sequential logic, use nonblocking assignments.

#2: When modeling latches, use nonblocking assignments.

#3: When modeling combinational logic with an always block, use blocking assignments.

#4: Do not mix blocking and nonblocking assignments in the same always block.

#5: Do not make assignments to the same variable from more than one always block.

#6: Use $strobe to display values that have been assigned using nonblocking assignments.

#7: Do not make assignments using #0 delays.

POSTECH

Assignment Submission

- Submit your assignment to LMS with filename:

  — Lab1_{TeamID}_{StudentID1}_{StudentID2}.pdf

    - PDF file of your report

  — Lab1_{TeamID}_{StudentID1}_{StudentID2}.zip

    - Zip file of your source code (without testbench)

    - One directory including two subdirectories (ALU, vending_machine) when unzipped

- Due date

  — ALU (1 week), vending machine (1 week) merged into 2 weeks to help students to distribute time freely

  — Code: 2022. 3. 15 / 09:00 a.m.

  — Report: 2022. 3. 15 / 23:59 p.m.

POSTECH

Announcement

- Please check your team in PLMS
  - If you don't send email to TAs until tomorrow (Feb. 29) midnight we will consider you a single-person team

- If you have any question, please post them on the Q&A board in PLMS

POSTECH

Reference

- "Register-transfer level", Wikipedia, last modified Jan 24, 2019, accessed Mar 03, 2019, https://en.wikipedia.org/wiki/Register-transfer_level

- http://www.sunburst-design.com/papers/CummingsSNUG2000SJ_NBA.pdf

POSTECH