CSED311 Lab2: RTL Design

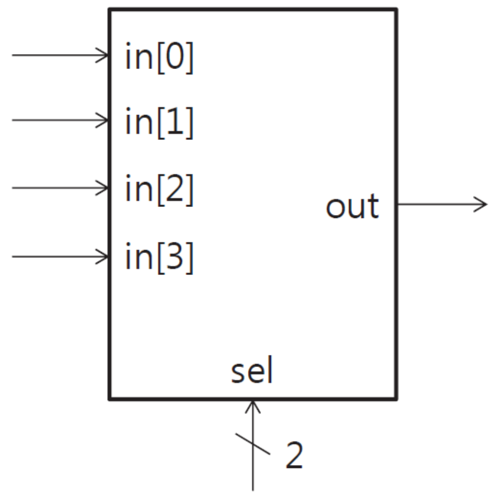
Sangyoun Kwak
ksy109@postech.ac.kr

Contents

Combinational logic

- Represents Boolean function (time-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)

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
```

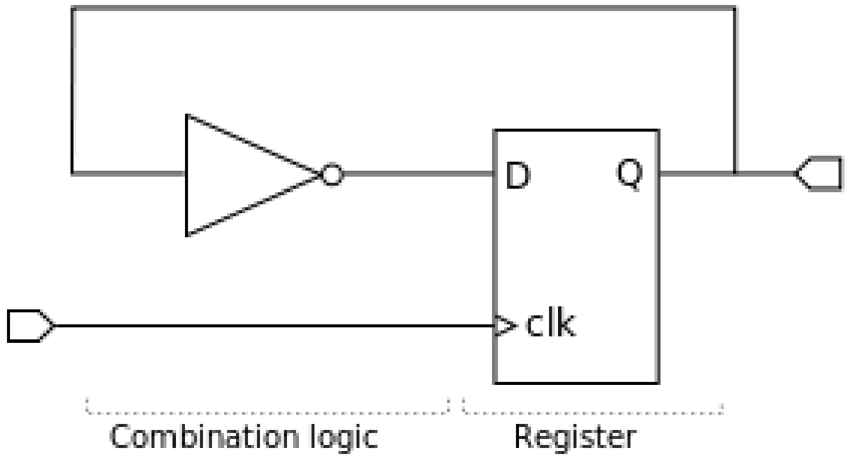
Register-Transfer Level

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

- Synchronous circuit consists of:
  - **Register**: a memory element synchronized by the clock signal
  - **Combinational logic**: logical function

RTL - Synchronous circuit example

- Toggler example



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

```
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
```

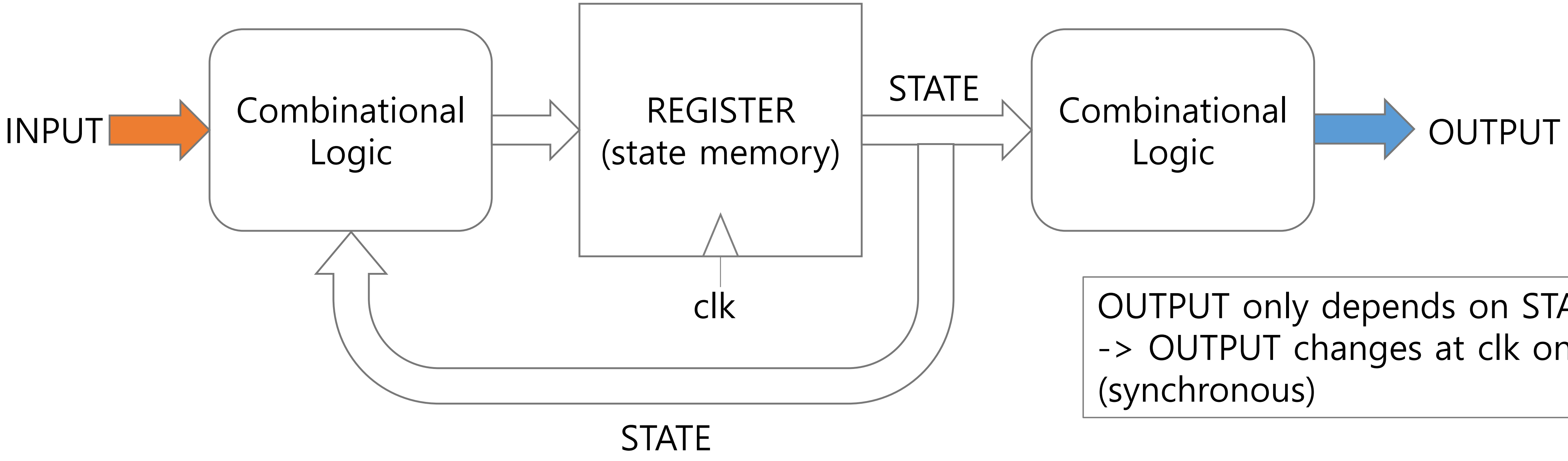
Finite State Machine (FSM)

- Moore machine

- Mealy Machine

Finite State Machine (FSM)

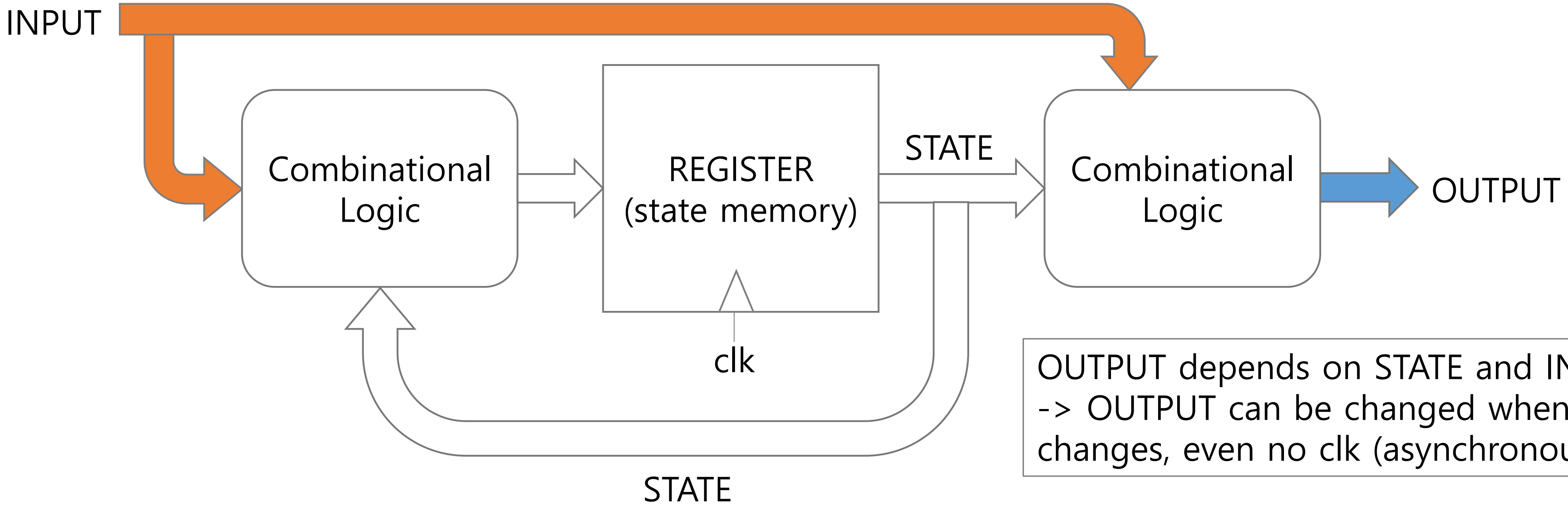
- Moore Machine
  - Outputs only depend on the current state.



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

Finite State Machine (FSM)

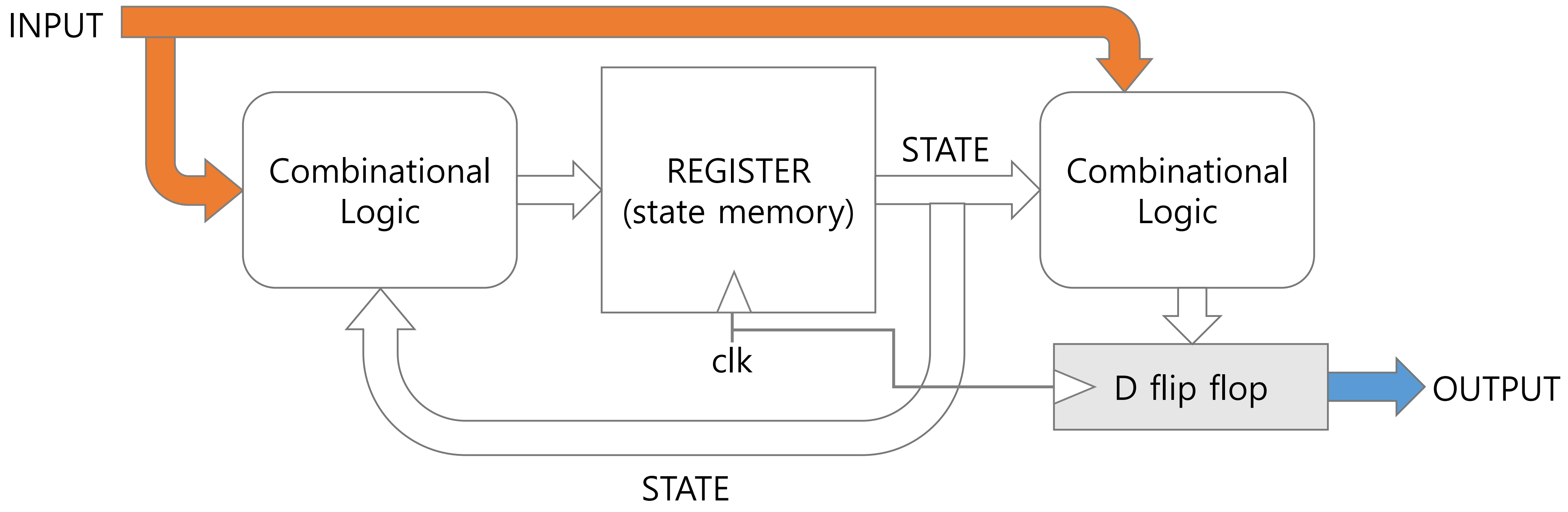
- Mealy Machine
  - Outputs depend on the current state and the current inputs



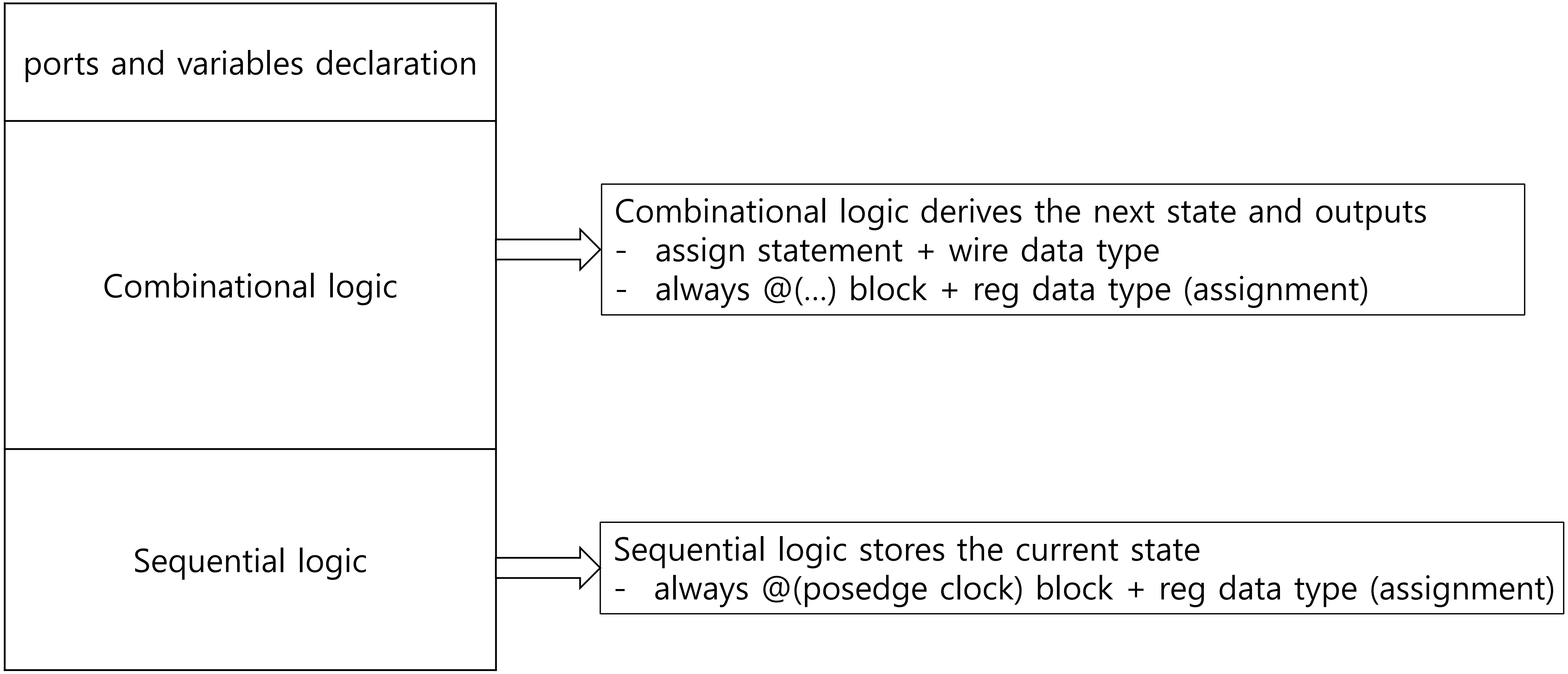
OUTPUT depends on STATE and INPUT
-> OUTPUT can be changed when INPUT changes, even no clk (asynchronous)

Finite State Machine (FSM)

- Mealy Machine
  - Mealy Machine can be made synchronous



FSM Code Structure

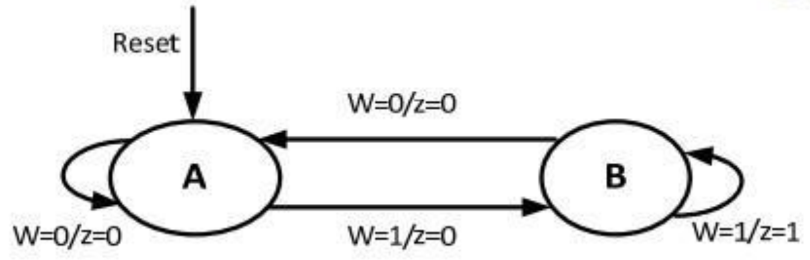


Combinational logic derives the next state and outputs
- assign statement + wire data type
- always @(…) block + reg data type (assignment)

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

FSM Code Structure

❖ Example: Mealy Machine



Example: Reduce 1's

Given a sequence of 0s and 1s, *change the first 1 of consecutive 1s into 0*

Examples:

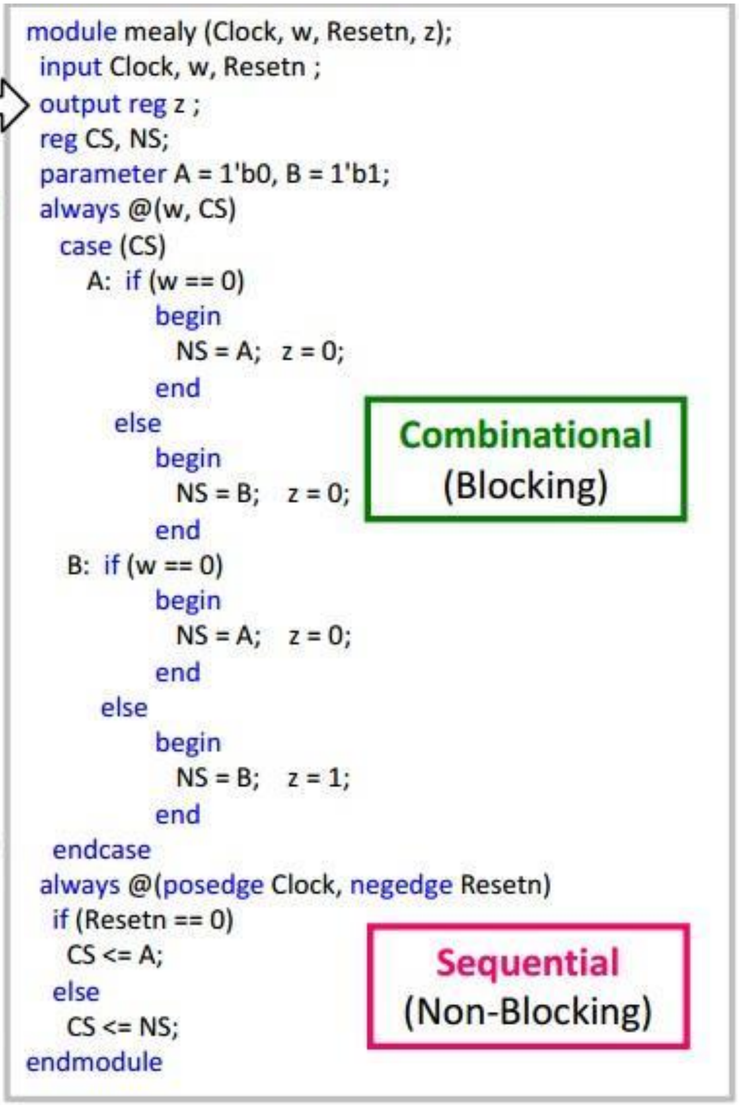
000011100 -> 000001100
010101010 -> 000000000
011001100 -> 001000100

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

Output: reg

NS & Output Calculation

CS Calculation



```
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
```

Combinational (Blocking)

Sequential (Non-Blocking)

Assignment

- 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 testbench will be provided.

- Submit a report and code to LMS
  - Due date: 2019. 3. 12 6:30 pm
  - We will check the demo in the next lab session.

Assignment

- 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

- Vending machine use-case

Assumption: infinite item, change

Sequence

```
1. Insert money (available money unit: 100, 500, 1000 won)
2. Vending machine shows all available items where (item cost <= current money && item count > 0)

3.a. Insert money within waiting time
  3.a.1. Go to 2
3.b. Select an item within 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
  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 waiting time
  3.c.1 Return changes

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

Assignment

- Submission
  - Submit your assignment to LMS in a single zip file with filename:
    - Lab2_StudentID.zip

  - This zip file should contain:
    - Lab2_StudentID_Report.pdf
      - PDF file of your report
    - Lab2_StudentID_Code.zip
      - Zip file of your code (*.v)

Assignment tips

- 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 = 10, Failed = 0”.

Demo

You will be asked to

1. Show the TA that your implementation works
    - We will provide a testbench code for testing your implementation.

2. Answer some questions about your design and implementation
    - If you did your assignment well, you don't have to worry!

POSTECH

Reference

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