# LAB 8.
# FSM 설계1: String Recognizer

2017 Fall Logic Design LAB

Computer Architecture & Embedded System Lab.

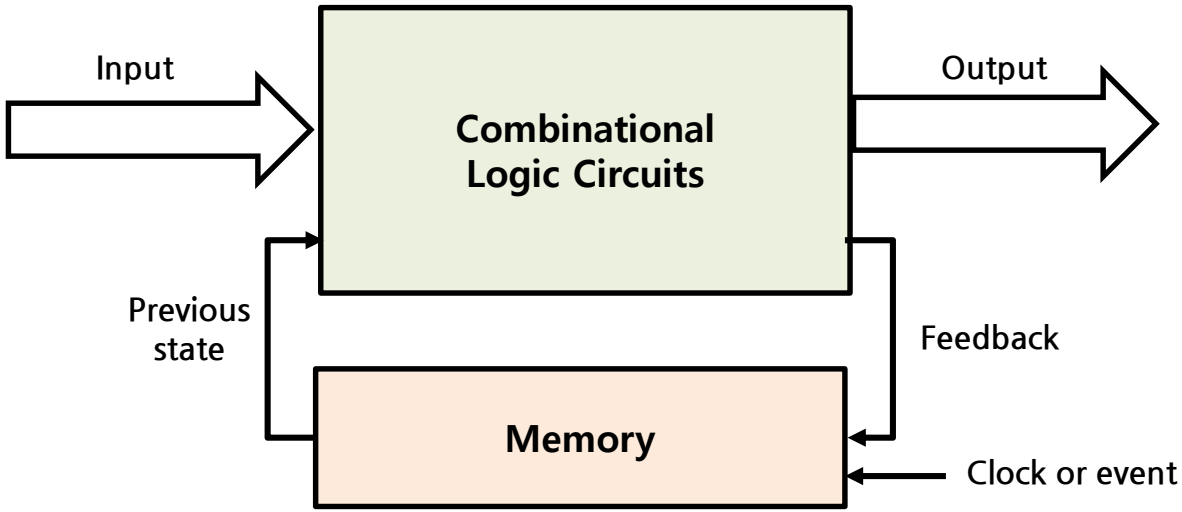
Seoul National University, Korea

# Outline

1. Finite State Machine
2. Finite State Machines in Verilog
3. Reduce-1-string-by-1 Moore Machine
4. String Recognizer

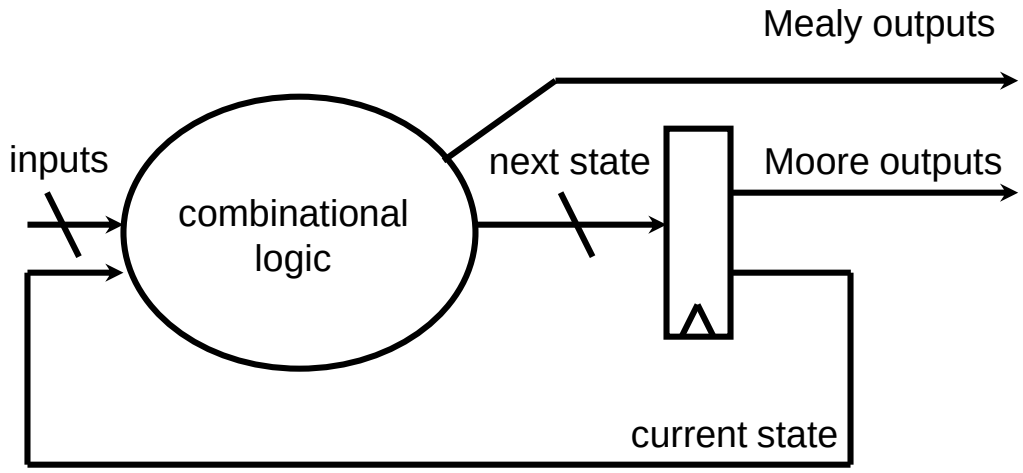
# 순차 논리 회로

- 순차논리회로(Sequential Logic Circuit)
  - 현재의 입력과 이전의 출력상태에 의해 출력이 결정
  - 동기(synchronous) 순차 논리회로 - clock pulse에 의해 동작하는 순차회로, 즉 clock pulse 시점에서 상태가 변화하는 회로
  - FSM으로 모델링 할 수 있음



# Finite State Machine

- Moore machine
  - 출력이 현재 상태만의 함수인 회로, 출력이 현재 상태에 따라 결정
- Mealy machine
  - 출력이 현재 상태 및 입력의 함수로 결정



# Finite State Machines in Verilog

- Verilog를 통한 FSM 모델링 (two always blocks)
  - Parameter 을 통하여 구현하고자 하는 FSM의 상태 정의 및 상태를 저장할 register 선언

```
reg [1:0] state, next;
parameter [1:0] STATE0 = 2'd0, STATE1 = 2'd1,
                STATE2 = 2'd2, STATE3 = 2'd3;
```

  - always 문을 이용하여 combinational logic 구현

```
always @ (in or state)
begin
    case (state)
        STATE0 : begin
            if (in) begin
                next = STATE1;
                out = 1'b1;
            end
...
```

  - always 문을 이용하여 sequential logic 구현 (sensitive list, non-blocking 문 사용)

```
always @ (posedge clk or posedge reset)
begin
    if (reset) state <= STATE0;
    else state <= next;
end
```

# Finite State Machines in Verilog

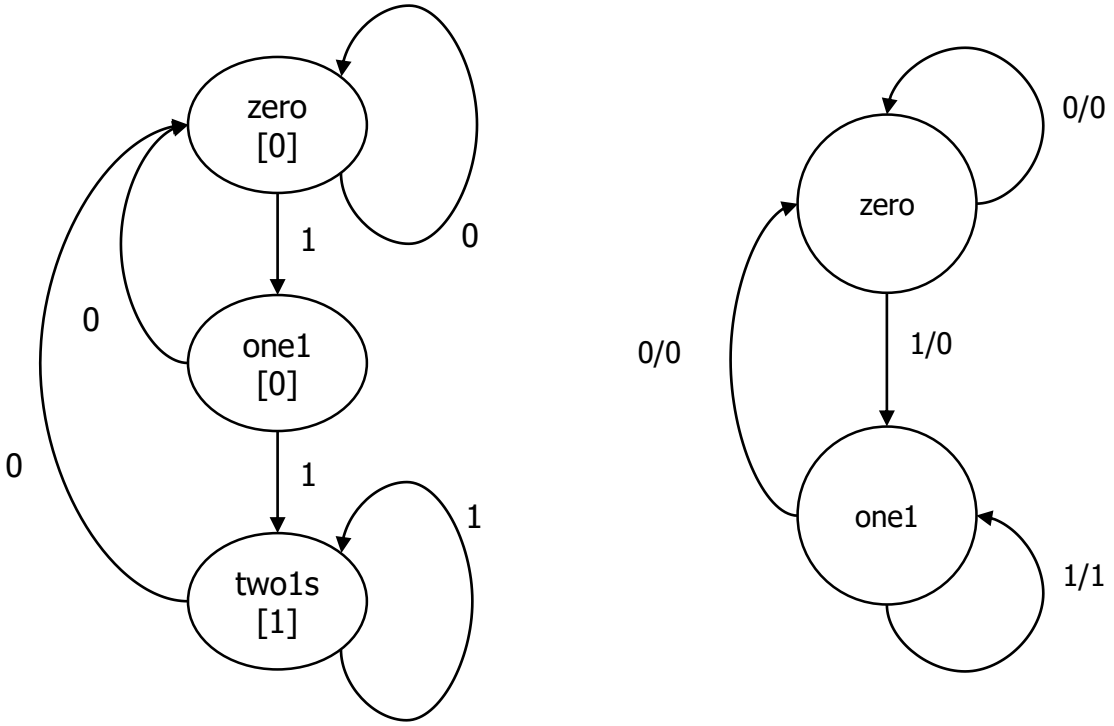
- Verilog를 통한 FSM 모델링 (three always blocks)
  - always 문을 하나 더 사용하여 state에 따른 출력 로직 분리 가능 (moore machine)

```
always @ (state)
begin
    case (state)
        STATE0: out = 1'b1;
        STATE1: out = 1'b0;
...
```

# Reduce-1-string-by-1 Moore Machine 예제

- 기능 : 1 bit입력을 받아 출력하되, 1이 연속될 때, 첫 번째 1제거
- 예 : 입력 => 0 1 1 1 0 1 0 1 1

  출력 => 0 0 1 1 0 0 0 0 1

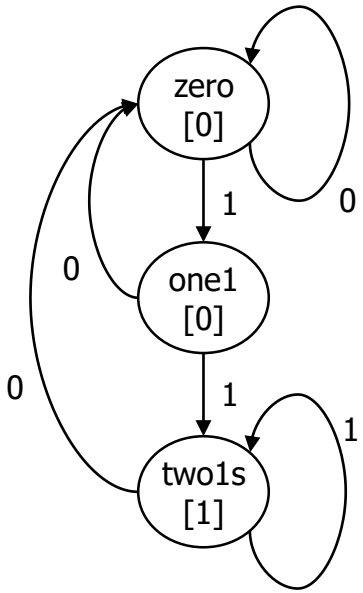



Reduce-1-string-by-1
moore, mealy machine

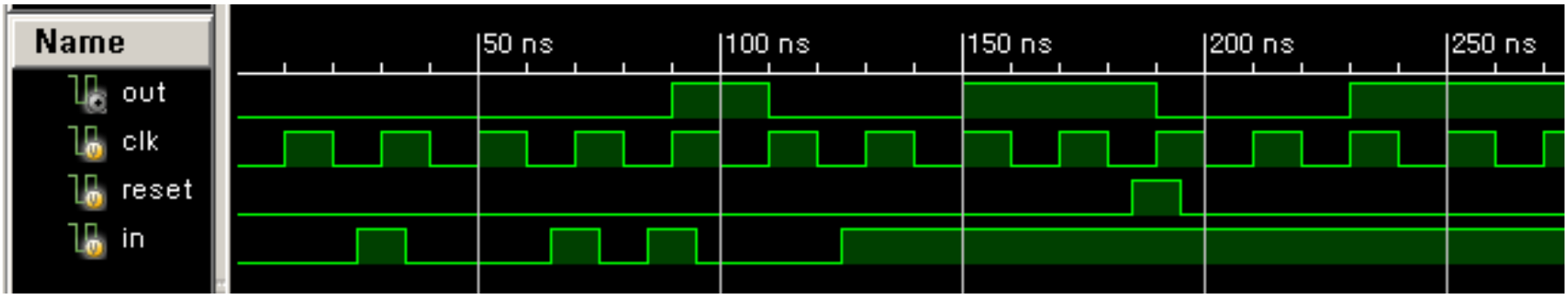
# Verilog Code for Reduce-1-string-by-1

- Moore machine

```
module reduce (
    input clk,
    input reset,
    input in,
    output reg out);

    parameter zero  = 2'b00, one1  = 2'b01,
              two1s = 2'b10;
    reg [1:0] state, next;

    always @(posedge clk or posedge reset)
    begin
        if (reset) state <= zero;
        else       state <= next;
    end

    always @(in or state)
    begin
        case (state)
        zero:  // last input was a zero
        begin
            if (in == 1'b1) next = one1;
            else next = zero;
        end
        one1:  // we've seen one 1
        begin
            if (in == 1'b1) next = two1s;
            else next = zero;
        end
        two1s: // we've seen at least 2 ones
        begin
            if (in == 1'b1) next = two1s;
            else next = zero;
        end
    endcase

    always @(state)
    begin
        case (state)
        zero: out = 0;
        one1: out = 0;
        two1s: out = 1;
        endcase
    end

endmodule
```



# 시뮬레이션 결과




| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| Out | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |

Reset

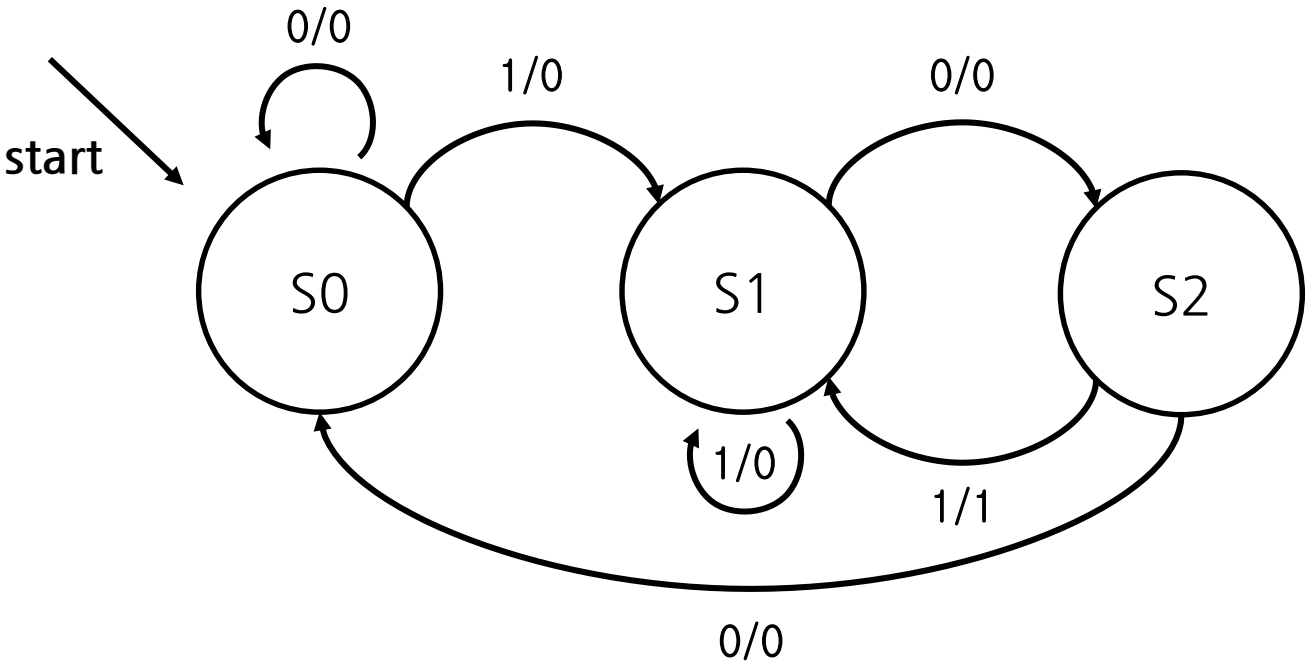
# String Recognizer Mealy Machine 예제

- 기능 : 입력 중 '101' 발견 시 1 출력, 그 외 0 출력
- 예 : Input => 1 0 1 0 1 0 0 1 1

  Output => 0 0 1 0 1 0 0 0 0

State Table

| | |
|---|---|
| S0 = '0' 발견 | 00 |
| S1 = '1' 발견 | 01 |
| S2 = '10' 발견 | 10 |

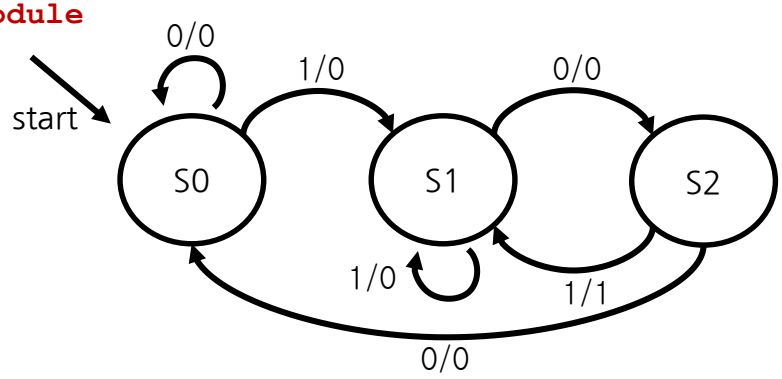



State diagrams (mealy machine)

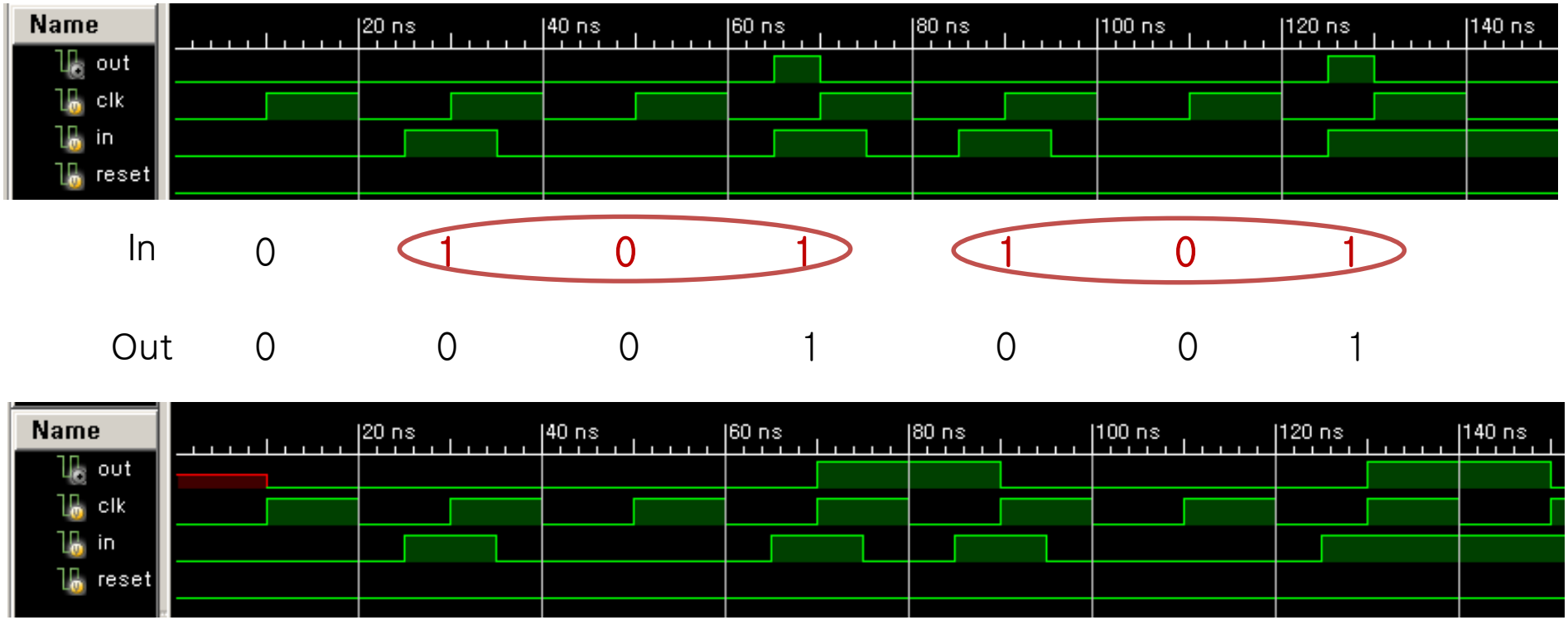
# Verilog Code for '101' String Recognizer

- Mealy machine

```
module recognizer101(
    input clk,
    input in,
    input reset,
    output reg out);

    parameter S0 = 2'b00,
              S1 = 2'b01,
              S2 = 2'b10;

    reg [1:0] state, next;

    always @ (in or state)
    begin
        out = 0;
        case (state)
        S0:
                if (in == 1'b1) next = S1;
                else next = S0;
        S1: // we've seen one 1
                if (in == 1'b1) next = S1;
                else next = S2;
        S2: // we've seen one 10
                if (in == 1'b1)
                begin
                        next = S1;
                        out = 1;
                end
                else next = S0;
        endcase
    end

    always @ (posedge clk or posedge reset)
    begin
        if (reset) state <= S0;
        else state <= next;
    end
endmodule
```



# 시뮬레이션 결과




Synchronous Mealy Machine 구현인 경우

# 실습 1. String Recognizer 구현

- 목표
  - String recognizer를 moore machine 과 mealy machine으로 각각 state diagram 그린 뒤 구현
- 실습 내용
  - 다음 세부 구현사항 참조
    - 1비트의 입력과 출력이 존재하며 reset input시 초기상태로 전이
    - 입력에서 …010… 이라는 string이 발견될 때 마다 1을 출력
    - 입력에서 …100… 이라는 string이 발견되면 영원히 0을 출력
  - 두 FSM을 구현한 뒤 시뮬레이션을 통해 작동을 확인한다.
- 제출 사항
  - 작성한 verilog 코드
  - 시뮬레이션 결과

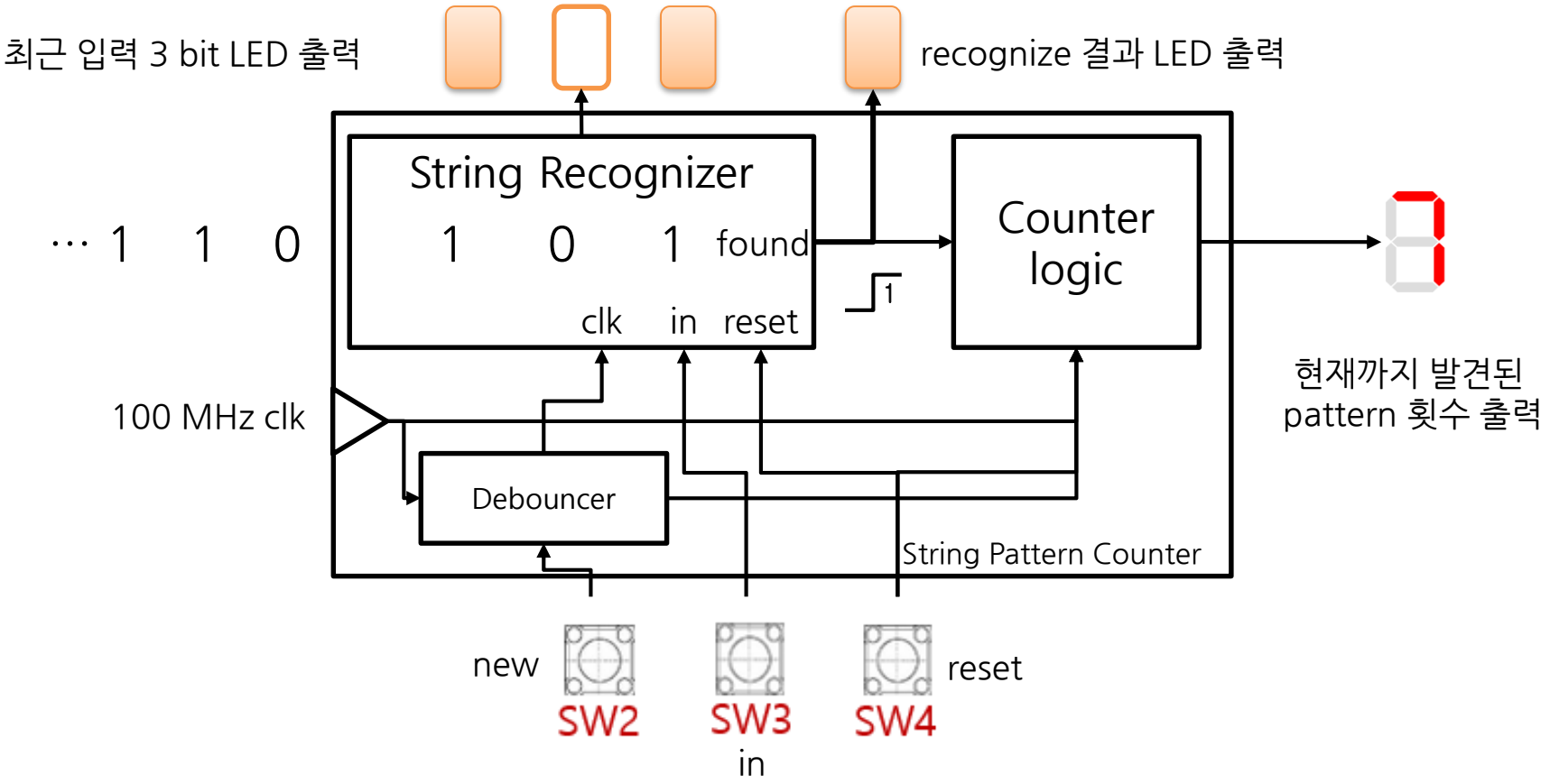
# 실습 2. String Recognizer FPGA 구현

- 목표
  - 실습 1에서 구현한 string recognizer moore machine을 실습 보드에서 작동 확인
- 세부 내용
  - 다음 세부 구현사항 참조
    - 입력은 push button 3개를 통하여 각각 clock, input, reset  입력
    - 지금까지 최근 들어온 3개의 input을 LED 3개에 출력
    - String recognize 결과 출력은 LED 1개를 통하여 출력
  - 클럭 입력은 보드의 오실레이터 대신 실습지에 제공되는 btn_debouncer 모듈을 통하여 push 버튼 입력을 클럭으로 사용하여 작동을 확인
- 제출 사항
  - 작성한 verilog 코드

# 실습 3. String Pattern Counter FPGA 구현

- 목표
  - 실습 2에서 구현한 string recognizer를 확장하여 현재까지 발견된 '010' 패턴 횟수를  7-segment display로 출력하도록 구현
- 세부 내용
  - String recognizer가 현재까지 발견한 '010' 패턴 횟수를 7-segment display로 출력하도록 구현
  - 세부 내용은 다음 슬라이드의 그림을 참조
- 제출 사항
  - 작성한 verilog 코드

# 실습 3. String Pattern Counter FPGA 그림



# 실습 제출 안내

- **제출 항목**
  - 실습지 참조하여 보고서 작성
  - 각 실습에서 작성한 Verilog 소스 코드

- **제출 방법 및 기한**
  - 작성한 보고서(PDF) 및 소스코드를 압축하여 하나의 파일로 제출
  - ETL 과제 게시판에 **팀 별로 제출**
  - **일요일 오후 6시**까지