An ambulance company wants a flashing yellow light that, when a button is held, will instead hold a solid red. Design this machine.
FSMs in Verilog - Declarations

module simple (clk, reset, w, out);
    input logic clk, reset, w;
    output logic out;

    enum { A, B, C} ps, ns; // Present state, next state

    // Diagram showing state transitions
FSMs in Verilog – Combinational Logic

// Next State Logic
always_comb begin
  case (ps)
    A: if (w) ns = B;
       else ns = A;
    B: if (w) ns = C;
       else ns = A;
    C: if (w) ns = C;
       else ns = A;
  endcase
end

// Output Logic - could also be "always",
// or part of next-state logic.
assign out = (ps == C);
FSMs in Verilog – DFFs

```verilog
// Sequential Logic (DFFs)
always_ff @(posedge clk) begin
  if (reset) ps <= A;
  else begin
    ps <= ps;  // Corrected the assignment
  end
endmodule
```
Circuit Diagram of FSM
module simple_testbench();
  logic  clk, reset, w, out;

  simple dut (.clk, .reset, .w, .out);

  // Set up the clock.
  parameter CLOCK_PERIOD=100;

  initial begin
    clk <= 0;
    forever #(CLOCK_PERIOD/2) clk <= ~clk;
  end
// Design inputs. Each line is a clock cycle.

// ONLY USE THIS FORM for testbenches!!!

initial begin
  @(posedge clk); //1
  reset <= 1;      //2
  @(posedge clk);
  reset <= 0;      //5
  w <= 0;          //6
  @(posedge clk);
  @(posedge clk);
  @(posedge clk);
  @(posedge clk);
  w <= 1;          //7
  @(posedge clk);
  w <= 0;          //8
  @(posedge clk);
  w <= 1;          //9
  @(posedge clk);
  @(posedge clk);
  @(posedge clk);
  w <= 0;          //10
  @(posedge clk);
  @(posedge clk);

$stop; // End the simulation.
end
endmodule
Testbench Waveforms

clk
reset
w
ps[1:0]
out

0/0 1/0 0/0 1/0 1/1

Reset
String Recognizer Example

- Recognize the string: 101
- Input: 1 0 0 1 0 1 0 1 1 0 0 1 0
- Output:
- State Machine: