12th Day

Sequence Detect(Moore) Design(1001)

```
module sequence_detector_moore( input clk, rst, in, output reg detected);

  parameter S0 = 3'b000,
            S1 = 3'b001,
            S2 = 3'b010,
            S3 = 3'b011,
            S4 = 3'b100;

  reg [2:0] state, next_state;

  always @(posedge clk )
   begin
     if (rst)
        state <= S0;
     else
        state <= next_state;
   end

  // Next State Logic
  always @(*)
  begin
     case (state)
        S0: next_state = (in) ? S1 : S0;
        S1: next_state = (in) ? S1 : S2;
```

```
        S2: next_state = (in) ? S0 : S3;
        S3: next_state = (in) ? S4 : S0;
        S4: next_state = (in) ? S1 : S0;
        default: next_state = S0;
      endcase
    end

    // Output Logic (Moore: Output depends only on State)
    always @(posedge clk)
     begin
     if(state==S4)
     detected <= 1'b1;
     else
     detected <=1'b0;

    end

endmodule
```

Testbench

```
module sequence_detector_tb;

    reg clk, rst, in;
    wire detected;

    sequence_detector_moore dut (.clk(clk), .rst(rst), .in(in), .detected(detected));


    always #5 clk = ~clk;
```

```
    initial begin
        clk = 0;
        rst = 1;
        in = 0;

        #10 rst = 0;

        // Apply test sequence (1001 pattern)
        @(posedge clk) in = 1;  // 1
        @(posedge clk) in = 0;  // 10
        @(posedge clk) in = 0;  // 100
        @(posedge clk) in = 1;  // 1001 -> Should be detected

        // Another sequence (01001)
        @(posedge clk) in = 0;  // 0
        @(posedge clk) in = 1;  // 01
        @(posedge clk) in = 0;  // 010
        @(posedge clk) in = 0;  // 0100
        @(posedge clk) in = 1;  // 01001 -> Should be detected again

        #50;
        $finish;
    end
endmodule
```

Simulation