

1011

```

module moore_1011 (
input wire din, clk,rst,
output reg dout);
localparam [2:0]
s0=3'd0,
s1=3'd1,
s2=3'd2,
s3=3'd3,
s4=3'd4;
reg[2:0] state,next_state;
always @(posedge clk or posedge rst) begin
if (rst)
state<=s0;
else
state <=next_state;
end

```

```

always @(*) begin
    case(state)
        s0: begin if (din) next_state =s1; else next_state=s0; end
        s1: begin if (!din) next_state =s2; else next_state=s1; end
        s2: begin if (din) next_state =s3; else next_state=s0; end
        s3: begin if (din) next_state =s4; else next_state=s2; end
        s4: begin if (din) next_state =s1; else next_state=s2; end
        default: next_state=s0;
    endcase
end
always @(*)begin
    case(state)
        s0: dout=1'd0;
        s1: dout=1'd0;
        s2: dout=1'd0;
        s3: dout=1'd0;
        s4: dout=1'd1;
        default: dout=1'd0;
    endcase
end
endmodule

```

---

```
`timescale 1ns / 1ps
```

```
module tb_moore_1011;
```

```
    reg clk, rst, din;
```

```
wire dout;
```

```
moore_1011 uut (  
    .clk(clk),  
    .rst(rst),  
    .din(din),  
    .dout(dout)  
);
```

```
initial begin
```

```
    clk = 0;
```

```
    forever #5 clk = ~clk;
```

```
end
```

```
initial begin
```

```
    $monitor("Time=%0t | din=%b | dout=%b", $time, din, dout);
```

```
    rst = 1; din = 0;
```

```
    #10 rst = 0;
```

```
    din = 1; #10;
```

```
    din = 0; #10;
```

```
    din = 1; #10;
```

```
    din = 1; #10;
```

```
    din = 0; #10;
```

```
    din = 1; #10;
```

```
din = 0; #10;
```

```
din = 1; #10;
```

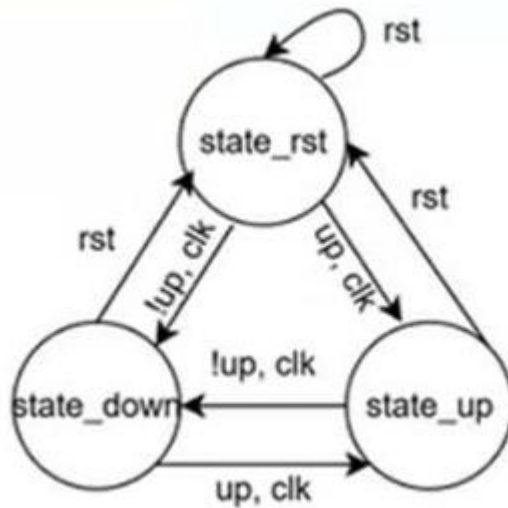
```
din = 1; #10;
```

```
#20 $finish;
```

```
end
```

```
endmodule
```

---



```
module fsm_updown_cnt(  
    input wire clk, rst, up,  
    output reg [3:0] count,  
    output reg [6:0] led_7  
);
```

```
localparam [1:0]
```

```
    state_rst    = 2'b00,
```

```

state_up_cnt = 2'b01,
state_down_cnt = 2'b10;

reg [1:0] current_state, next_state;

// Next-state logic
always @(*) begin
    next_state = current_state;
    case (current_state)
        state_rst: begin
            if (up) next_state = state_up_cnt;
            else next_state = state_down_cnt;
        end
        state_up_cnt: begin
            if (!up) next_state = state_down_cnt;
            else next_state = state_up_cnt;
        end
        state_down_cnt: begin
            if (up) next_state = state_up_cnt;
            else next_state = state_down_cnt;
        end
        default: next_state = state_rst;
    endcase
end

// State register + counter
always @(posedge clk or posedge rst) begin
    if (rst) begin
        current_state <= state_rst; // reset FSM về trạng thái ban đầu
    end
end

```

```
        count <= 4'd0;
    end else begin

        current_state <= next_state;
        case (next_state)
            state_rst: count <= count;
            state_up_cnt: begin
                if (count == 4'd9)
                    count <= 4'd0;
                else
                    count <= count + 1'b1;
                end
            state_down_cnt: begin
                if (count == 4'd0)
                    count <= 4'd9;
                else
                    count <= count - 1'b1;
                end
            endcase
        end
    end
end

always @(*) begin
    case (count)
        4'd0: led_7 = 7'b1000000;
        4'd1: led_7 = 7'b1111001;
        4'd2: led_7 = 7'b0100100;
        4'd3: led_7 = 7'b0110000;
        4'd4: led_7 = 7'b0011001;
        4'd5: led_7 = 7'b0010010;
```

```
    4'd6: led_7 = 7'b0000010;
    4'd7: led_7 = 7'b1111000;
    4'd8: led_7 = 7'b0000000;
    4'd9: led_7 = 7'b0010000;
    default: led_7 = 7'b1111111;
endcase
end
```

```
endmodule
```

---

```
`timescale 1ns/1ps
module tb_fsm_updown_cnt;

    reg clk, rst, up;
    wire [3:0] count;
    wire [6:0] led_7;

    // Kết nối với module chính
    fsm_updown_cnt uut (
        .clk(clk),
        .rst(rst),
        .up(up),
        .count(count),
        .led_7(led_7)
    );

    // Tạo xung clock 10ns
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
endmodule
```

end

// In trạng thái mô phỏng ra màn hình

initial begin

\$display("=== BẮT ĐẦU MÔ PHỎNG ===");

\$display("Thời gian\tclk\trst\tup\tcount\tled\_7");

\$monitor("%0t\t%b\t%b\t%b\t%d\t%b", \$time, clk, rst, up, count, led\_7);

end

// Kịch bản mô phỏng

initial begin

// Reset ban đầu

rst = 1; up = 1;

repeat(2) @(posedge clk);

rst = 0;

// Đếm lên

\$display("== ĐẾM TIẾN ==");

up = 1;

repeat(12) @(posedge clk);

// Đếm lùi

\$display("== ĐẾM LÙI ==");

up = 0;

repeat(12) @(posedge clk);

// Thay đổi liên tục

\$display("== THAY ĐỔI LIÊN TỤC ==");

up = 1; repeat(5) @(posedge clk);

```
up = 0; repeat(5) @(posedge clk);  
up = 1; repeat(5) @(posedge clk);  
up = 0; repeat(5) @(posedge clk);
```

```
// Reset giữa chừng  
$display("== RESET GIỮA CHỪNG ==");  
rst = 1; @(posedge clk);  
rst = 0; up = 1;  
repeat(6) @(posedge clk);
```

```
// Kết thúc mô phỏng  
$display("== KẾT THÚC MÔ PHỎNG ==");  
$stop;  
end
```

```
endmodule
```

.....