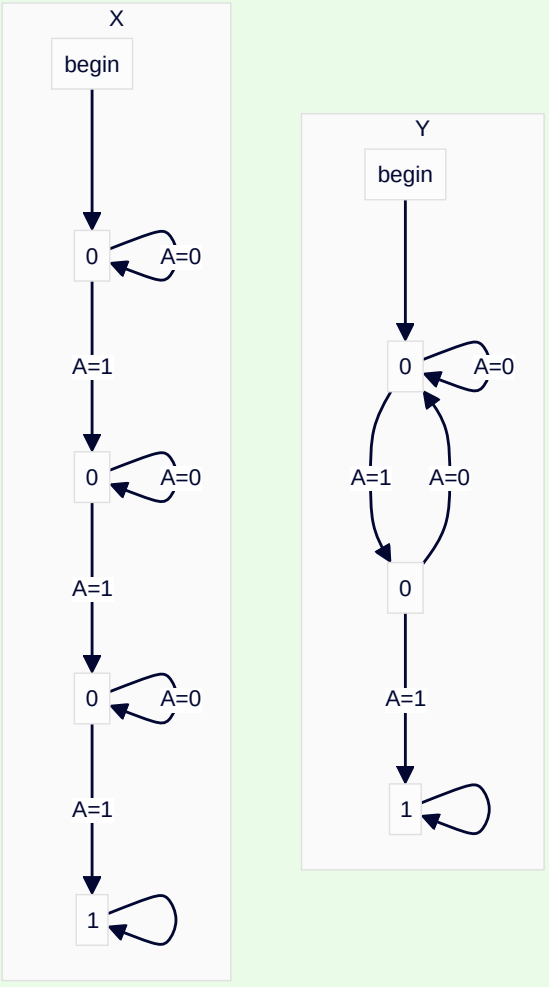
1

Draw the state diagram for the following problem: There is one input A and two outputs X and Y. X becomes 1 if A has been 1 for at least three cycles altogether (not necessarily consecutively). Y becomes 1 if A has been 1 for at least two consecutive cycles.

✓ Answer

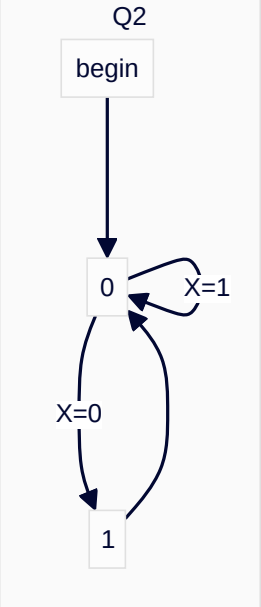


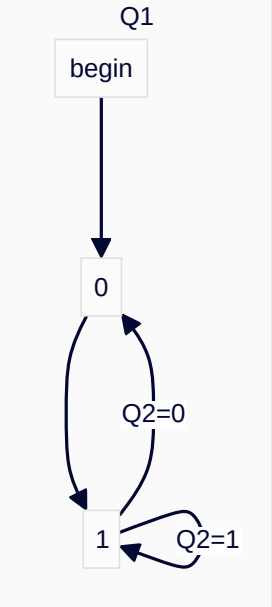
2

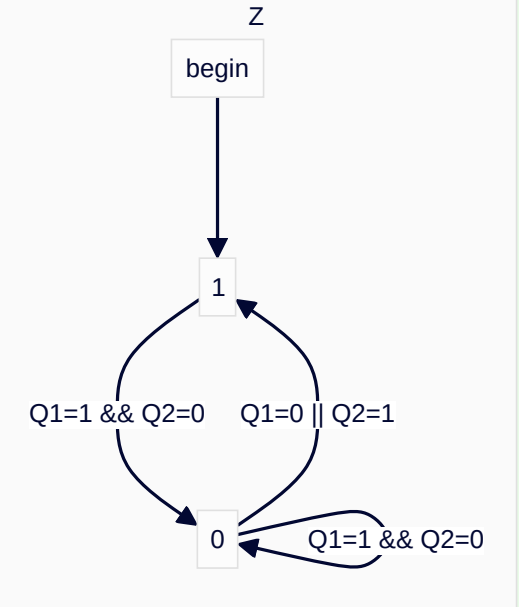
Given the following excitation and output equations, write the state transition and output tables. X is the input, Z is the output. Draw the sequential logic circuit. Please note that state memory is using positive-edge triggered D flip-flops.

$$D1 = Q1' + Q1 \cdot Q2$$
$$D2 = Q2' \cdot X'$$
$$Z = Q1' + Q2$$

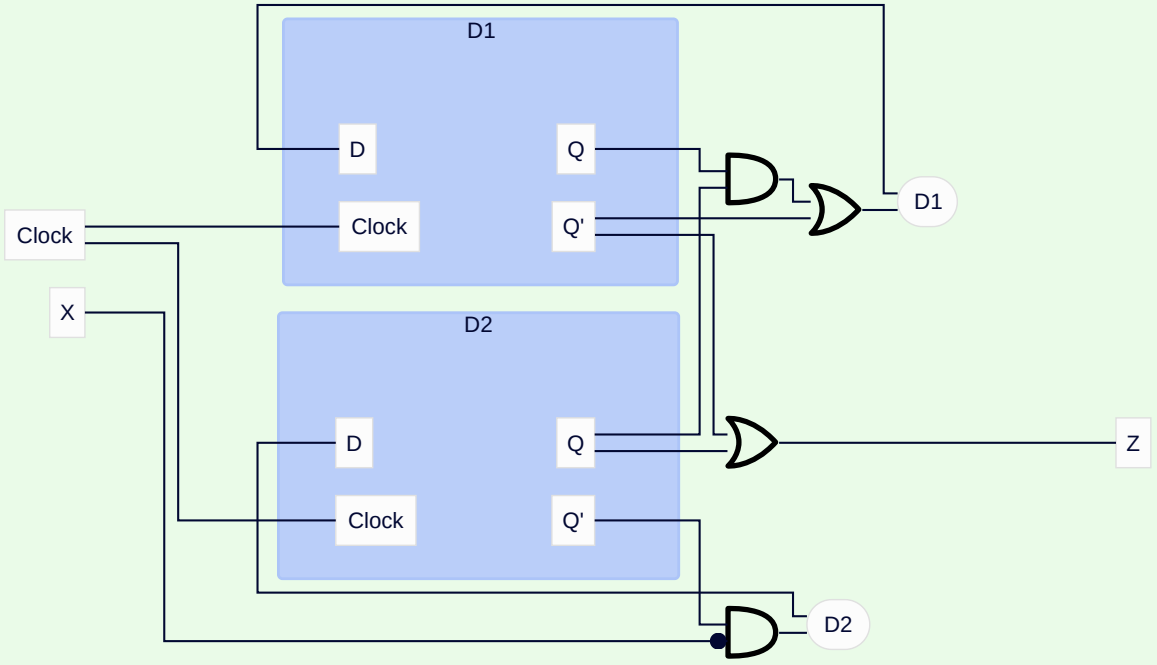
✓ Answer







Q1	Q2	X	D1	D2	Z
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1



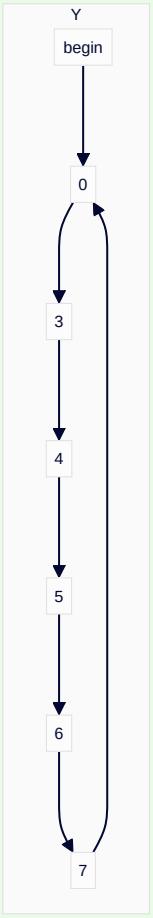
3

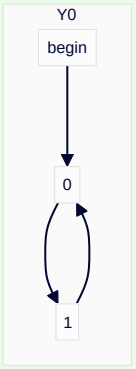
Design a 3-bit binary counter with counting sequence 0, 3, 4, 5, 6, 7, 0, 3, ...
The counter will increment at each clock tick. Use binary state assignments. Use D flip-flops, and combinational logic gates.

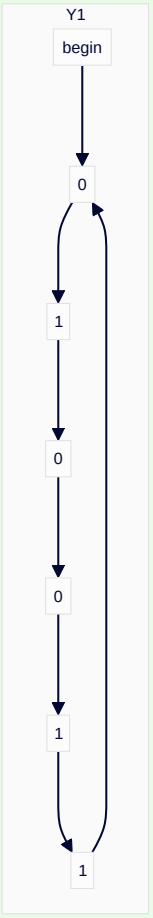
Follow the steps below:

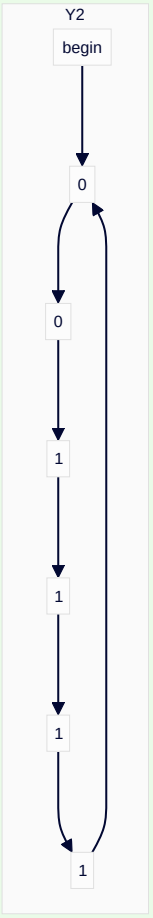
1. Draw a state diagram.
2. Write a state transition table.
3. Write state transition, and excitation equations.
4. Draw the circuit.

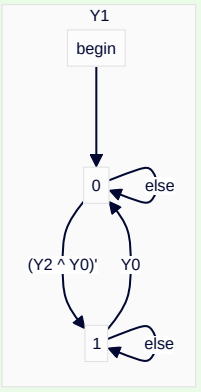
✓ **Answer**

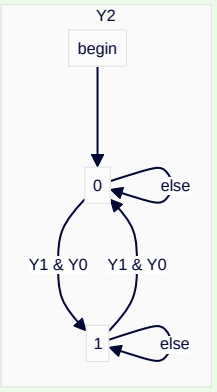






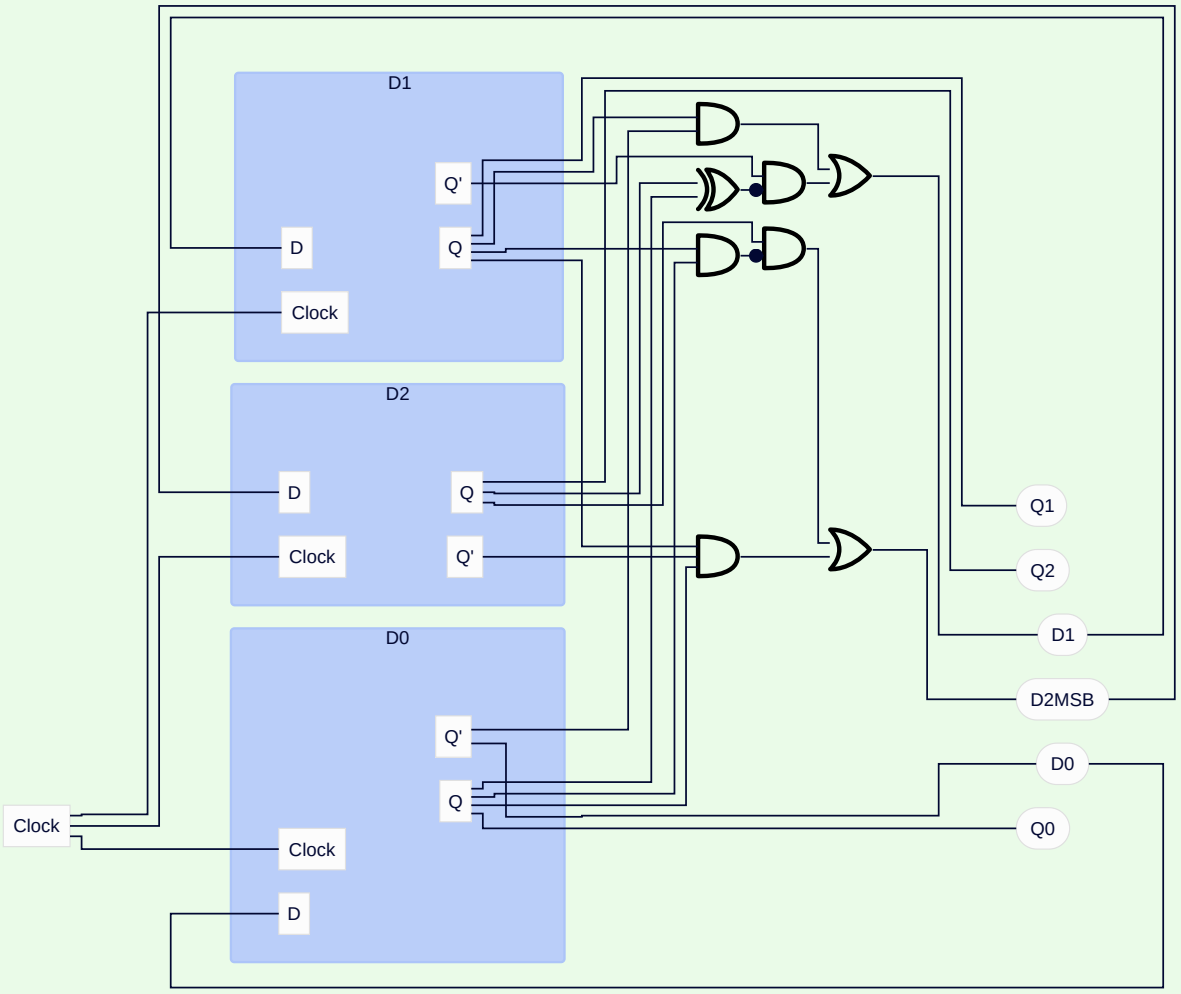






Y2	Y1	Y0	Y2*	Y1*	Y0*
0	0	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

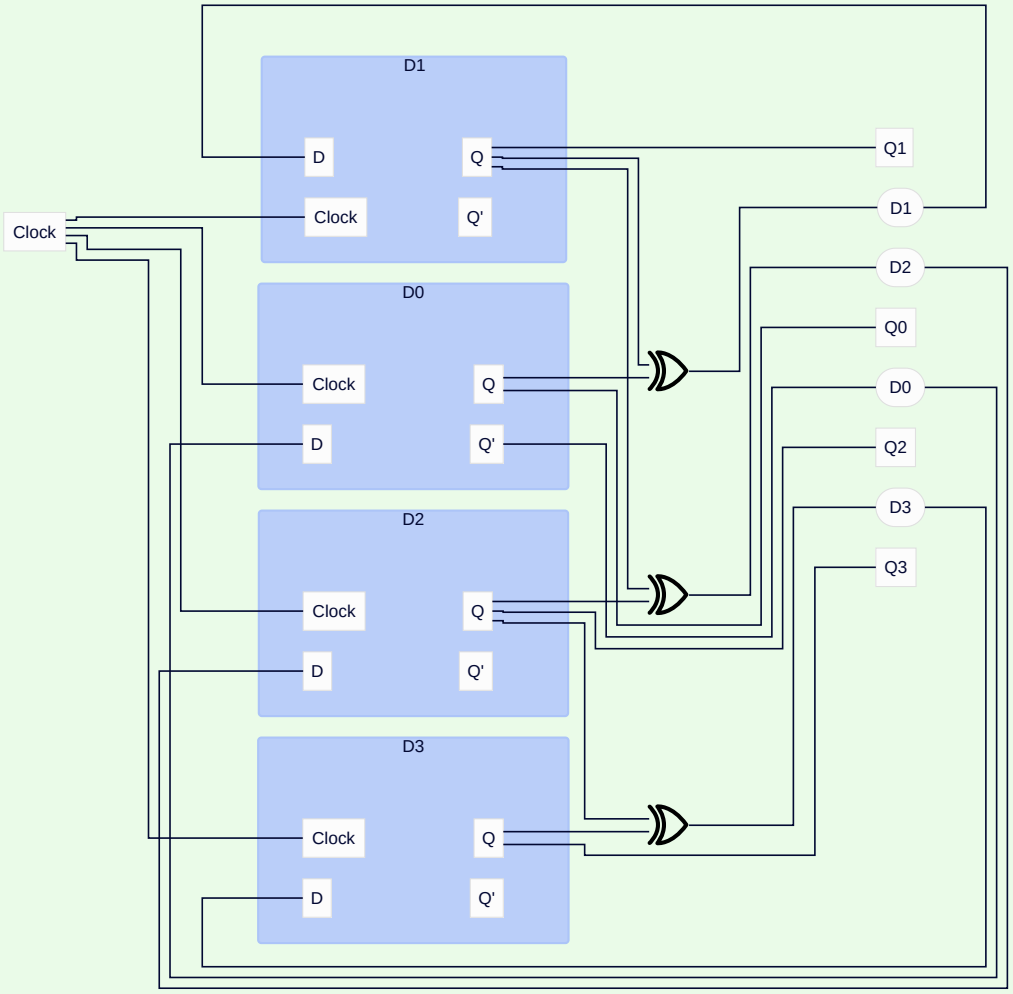
$$D0 = Q0'$$
$$D1 = Q1' \cdot (Q2 \wedge Q0)' + Q1 \cdot Q0'$$
$$D2 = Q2' \cdot Q1 \cdot Q0 + Q2 \cdot (Q1 \cdot Q0)'$$



4

Design a 4-bit ripple up counter using only four D flip-flops.

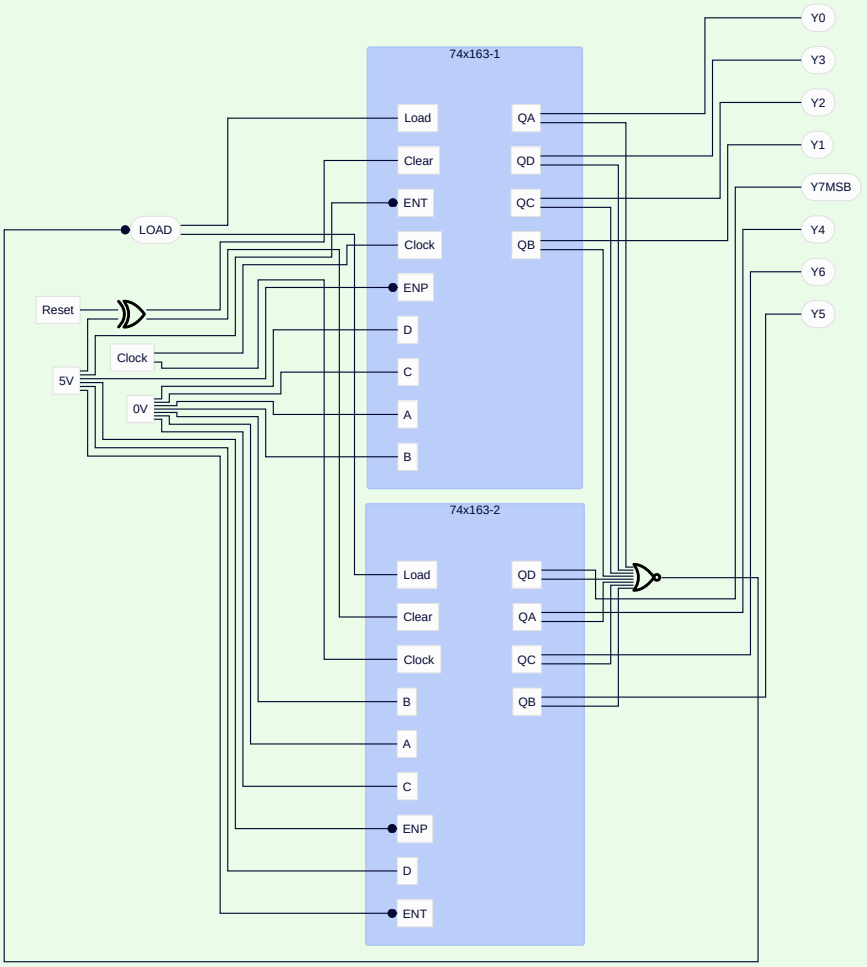
✓ **Answer**



5

Using two 74x163 counters, design a counter with counting sequence 0, 128, 129,..., 254, 255, 0, 128, 129, ... , 254, 255. Logic 0 and 1 are available.

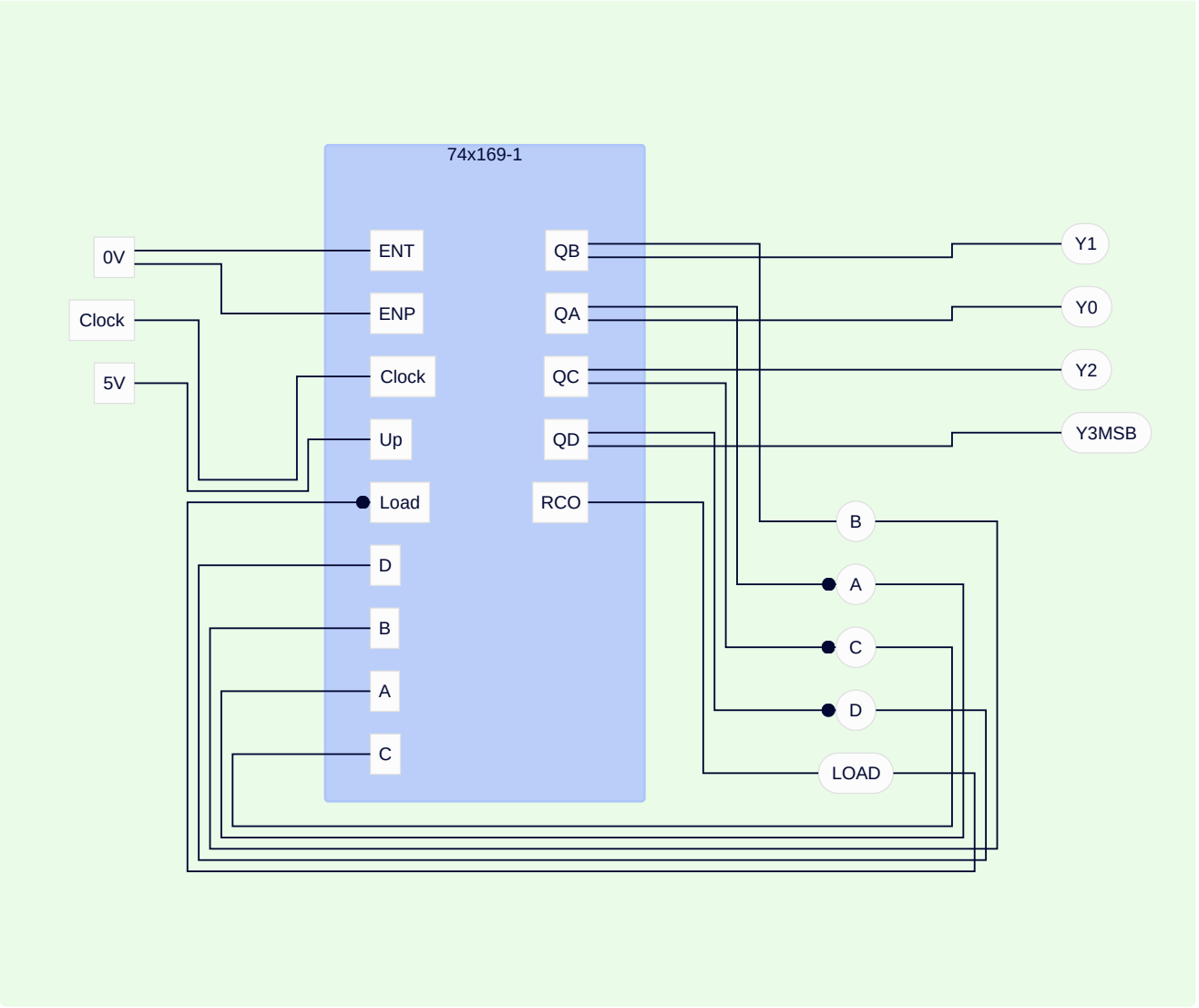
✓ **Answer**



6

Using one 74x169 and three inverters, design a counter with the counting sequence 4, 3, 2, 1, 0, 11, 12, 13, 14, 15, 4, 3 ...

✓ **Answer**



Code

tbird_fsm.sv

```
typedef enum logic [2:0] {
    IDLE = 3'b000,
    L1   = 3'b001,
    L2   = 3'b011,
    L3   = 3'b010,
    R1   = 3'b101,
    R2   = 3'b111,
    R3   = 3'b110,
    LR3  = 3'b100
} t_tbird_lights_state;

module tbird_fsm (
    input  logic       clk,
    input  logic       rst_b,
    input  logic       left,
    input  logic       right,
    input  logic       haz,
```

```
    output logic [2:0] l_lights,
    output logic [2:0] r_lights
);
    t_tbird_lights_state state;

    always_ff @(posedge clk or negedge rst_b) begin
        if (~rst_b)
            state <= IDLE;
        else
        case (state)
            IDLE: begin
                if ( (left & right) | haz)
                    state <= LR3;
                else if (left)
                    state <= L1;
                else if (right)
                    state <= R1;
                else
                    state <= IDLE;
            end
            L1:  state <= haz ? LR3 : L2;
            L2:  state <= haz ? LR3 : L3;
            L3:  state <= IDLE;
            R1:  state <= haz ? LR3 : R2;
            R2:  state <= haz ? LR3 : R3;
            R3:  state <= IDLE;
            LR3: state <= IDLE;
        endcase
    end

    always_comb begin
        case (state)
            IDLE: begin
                l_lights = 3'b000;
                r_lights = 3'b000;
            end
            L1: begin
                l_lights = 3'b001;
                r_lights = 3'b000;
            end
            L2: begin
                l_lights = 3'b011;
                r_lights = 3'b000;
            end
            L3: begin
                l_lights = 3'b111;
                r_lights = 3'b000;
            end
            R1: begin
                l_lights = 3'b000;
```

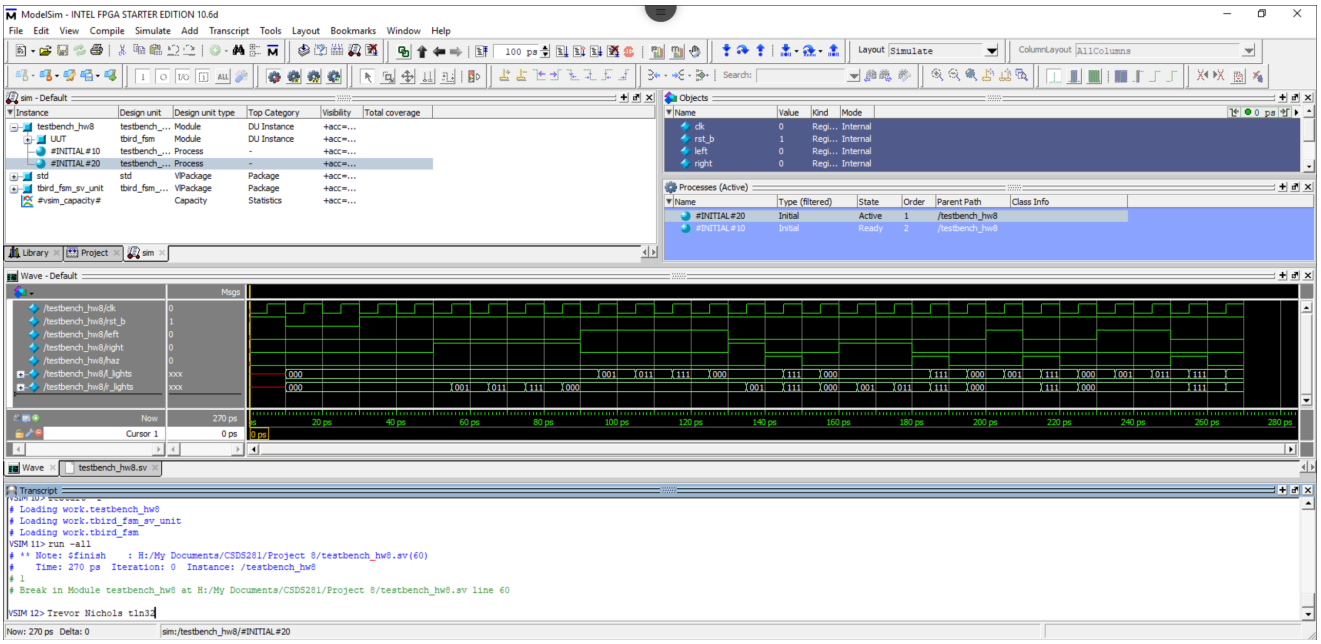
```
                r_lights = 3'b001;
            end
            R2: begin
                l_lights = 3'b000;
                r_lights = 3'b011;
            end
            R3: begin
                l_lights = 3'b000;
                r_lights = 3'b111;
            end
            LR3: begin
                l_lights = 3'b111;
                r_lights = 3'b111;
            end
        endcase
    end

endmodule
```

testbench_hw8.sv

```
module testbench_hw8 ();

    logic clk, rst_b, left, right, haz;
    logic [2:0] l_lights, r_lights;

    tbird_fsm UUT (
        clk, rst_b, left, right, haz, l_lights, r_lights
    );

    initial begin
        clk   = 1'b0;
        rst_b = 1'b1;
        left  = 1'b0;
        right = 1'b0;
        haz   = 1'b0;

        forever #5 clk = ~clk;
    end

    initial begin
        #10;
        rst_b = 1'b0;
        #20;
        rst_b = 1'b1;
        #20;
        right = 1'b1;
        #40;
```

```
        right = 1'b0;
        left  = 1'b1;
        #40;
        left  = 1'b0;
        right = 1'b1;
        #10;
        right = 1'b0;
        haz   = 1'b1;
        #10;
        haz   = 1'b0;
        #10;
        right = 1'b1;
        #20;
        right = 1'b0;
        haz   = 1'b1;
        #10;
        haz   = 1'b0;
        #10;
        left = 1'b1;
        #10;
        left = 1'b0;
        haz  = 1'b1;
        #10;
        haz   = 1'b0;
        #10;
        left = 1'b1;
        #20;
        left = 1'b0;
        haz  = 1'b1;
        #10;
        haz   = 1'b0;
        #10;
        $finish();
    end
endmodule
```

Deliverables


ModelSim - INTEL FPGA STARTER EDITION 10.6d
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
sim - Default
Instance Design unit Design unit type Top Category Visibility Total coverage
testbench_hw8 testbench_... Module DU Instance +acc=...
UUT tbird_fsm Module DU Instance +acc=...
#INITIAL#10 testbench_... Process - +acc=...
#INITIAL#20 testbench_... Process - +acc=...
std std VHPackage Package +acc=...
tbird_fsm_sv_unit tbird_fsm_... VHPackage Package +acc=...
#vsim_capacity# Capacity Statistics +acc=...
Objects
Name Value Kind Mode
clk 0 Regi... Internal
rst_b 1 Regi... Internal
left 0 Regi... Internal
right 0 Regi... Internal
Processes (Active)
Name Type (filtered) State Order Parent Path Class Info
#INITIAL#20 Initial Active 1 /testbench_hw8
#INITIAL#10 Initial Ready 2 /testbench_hw8
Library Project sim
Wave - Default
Msgs
/testbench_hw8/clk
/testbench_hw8/rst_b
/testbench_hw8/left
/testbench_hw8/right
/testbench_hw8/haz
/testbench_hw8/l_lights
/testbench_hw8/r_lights
Now 270 ps
Cursor 1 0 ps
20 ps 40 ps 60 ps 80 ps 100 ps 120 ps 140 ps 160 ps 180 ps 200 ps 220 ps 240 ps 260 ps 280 ps
Wave testbench_hw8.sv
Transcript
Loading work.testbench_hw8
Loading work.tbird_fsm_sv_unit
Loading work.tbird_fsm
VSIM 11> run -all
** Note: $finish : H:/My Documents/CSDS281/Project 8/testbench_hw8.sv(60)
Time: 270 ps Iteration: 0 Instance: /testbench_hw8
1
Break in Module testbench_hw8 at H:/My Documents/CSDS281/Project 8/testbench_hw8.sv line 60
VSIM 12> Trevor Nichols tln32
Now: 270 ps Delta: 0
sim:/testbench_hw8/#INITIAL#20