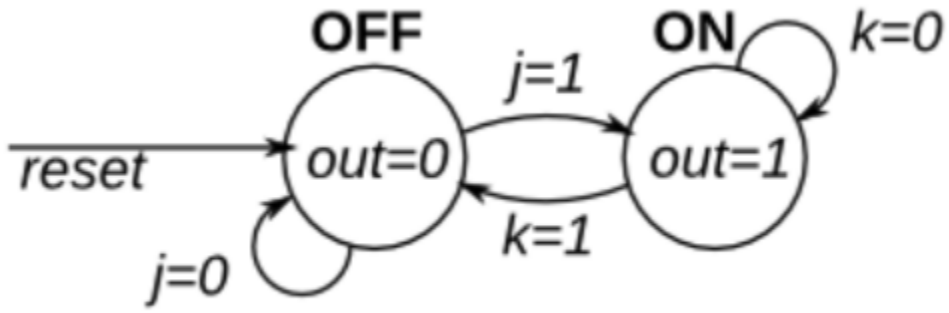
KARTHICK RAJA B
126004121

Given:

This is a Moore state machine with two states, two inputs, and one output. Implement this state machine.



Verilog Code:

```
module moore_machine (
    input clk,
    input reset,
    input j,
    input k,
    output reg out
);
    parameter OFF = 1'b0;
    parameter ON  = 1'b1;

    reg current_state, next_state;

    always @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= OFF;
        else
            current_state <= next_state;
    end

    always @(*) begin
```

```
        case (current_state)
            OFF: begin
                if (j)
                    next_state = ON;
                else
                    next_state = OFF;
            end

            ON: begin
                if (k)
                    next_state = OFF;
                else
                    next_state = ON;
            end

            default: next_state = OFF;
        endcase
    end


    always @(*) begin
        case (current_state)
            OFF: out = 0;
            ON:  out = 1;
            default: out = 0;
        endcase
    end
```

endmodule

Testbench code:

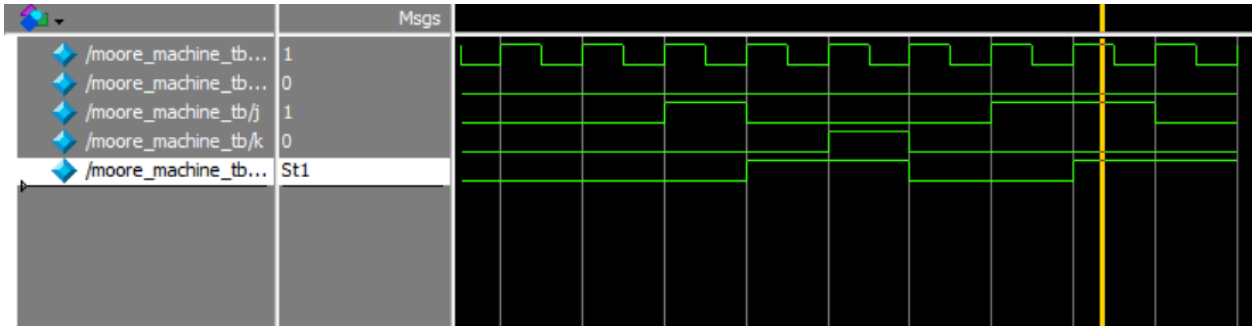
```
`timescale 1ns / 1ps
module moore_machine_tb();
    reg clk;
    reg reset;
```

```
    reg j;
    reg k;
    wire out;
moore_machine uut (
        .clk(clk),
        .reset(reset),
        .j(j),
        .k(k),
        .out(out)
    );
     always #5 clk = ~clk;
     initial begin
        clk = 0;
        reset = 1;
        j = 0;
        k = 0;
        #10;
        reset = 0;
        #10 j = 1;
        #10 j = 0;
        #10 k = 1;
        #10 k = 0;
        #10 j = 1;
        #10 j = 0;
        #10 reset = 1;
        #10 reset = 0;
        #10;

        $finish;
    end

endmodule
```

Functional Simulation:


Msgs
/moore_machine_tb...
/moore_machine_tb...
/moore_machine_tb/j
/moore_machine_tb/k
/moore_machine_tb...
1
0
1
0
St1