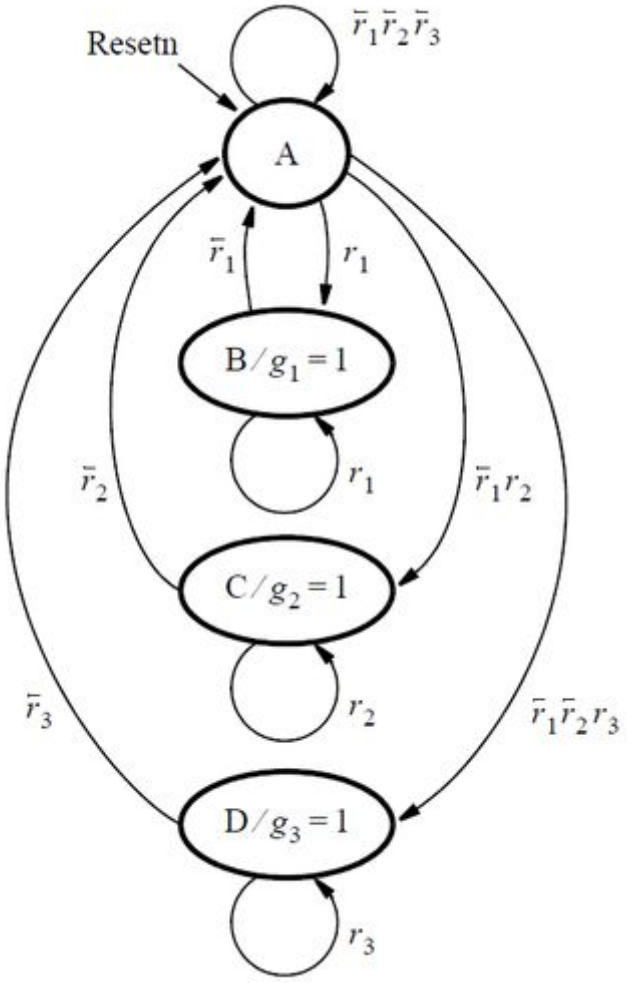
Exams/2013 q2afsm

Consider the FSM described by the state diagram shown below:



This FSM acts as an arbiter circuit, which controls access to some type of resource by three requesting devices. Each device makes its request for the resource by setting a signal *r[i]* = 1, where *r[i]* is either *r[1]*, *r[2]*, or *r[3]*. Each r[i] is an input signal to the FSM, and represents one of the three devices. The FSM stays in state *A* as long as there are no requests. When one or more request occurs, then the FSM decides which device receives a grant to use the resource and changes to a state that sets that device's *g[i]* signal to 1. Each *g[i]* is an output from the FSM. There is a priority system, in that device 1 has a higher priority than device 2, and device 3 has the lowest priority. Hence, for example, device 3 will only receive a grant if it is the only device making a request when the FSM is in state *A*. Once a device, *i*, is given a grant by the FSM, that device continues to receive the grant as long as its request, *r[i]* = 1.

Write complete Verilog code that represents this FSM. Use separate always blocks for the state table and the state flip-flops, as done in lectures. Describe the FSM outputs, *g[i]*, using either continuous assignment statement(s) or an always block (at your discretion). Assign any state codes that you wish to use.

這是一個有優先權的FSM, r1有最高權限, r3最小。 如果搶到權限後可以一直佔住grant持續使用。
既然是有優先順序，必須在next state裡寫出硬體上有順序性的if(第一順位), else if(第二順位), else if(第三順位)....

```
module top_module (
    input clk,
    input resetn,    // active-low synchronous reset
    input [3:1] r,   // request
    output [3:1] g   // grant
);
    parameter A = 2'd0;
    parameter B = 2'd1;
    parameter C = 2'd2;
    parameter D = 2'd3;

    reg [1:0] curr_state;
    reg [1:0] next_state;

    always @ (posedge clk )begin
        if(~resetn)begin
            curr_state <= A;
        end
        else begin
            curr_state <= next_state;
        end
    end

    always @ (*) begin
        case(curr_state)
            A: if (r[1]) begin
                next_state = B;
            end
            else if(r[2]) begin
                next_state = C;
            end
            else if(r[3]) begin
                next_state = D;
            end
            else begin
                next_state=A;
            end

            B: next_state = r[1]? B:A;
            C: next_state = r[2]? C:A;
            D: next_state = r[3]? D:A;
        endcase
    end

    assign g[1] = curr_state==B;
    assign g[2] = curr_state==C;
    assign g[3] = curr_state==D;

endmodule
```