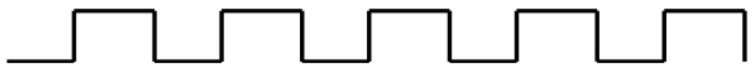
HW5

Saturday, October 12, 2024 12:53 PM

Thiago Gesteira

[illegible]

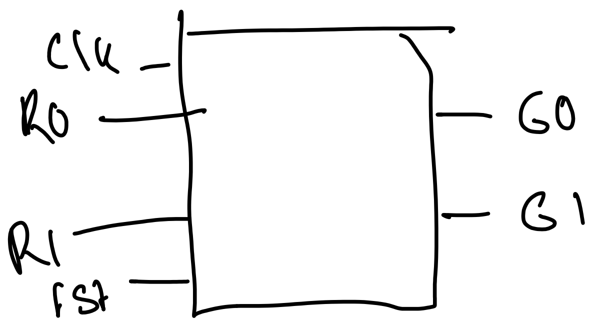


```
R0     0     1     0     1    0    0
R1     0     1     0     0    1    0
G0     0  0    1     0     1     0
G1     0  0    0     1     0     1
```

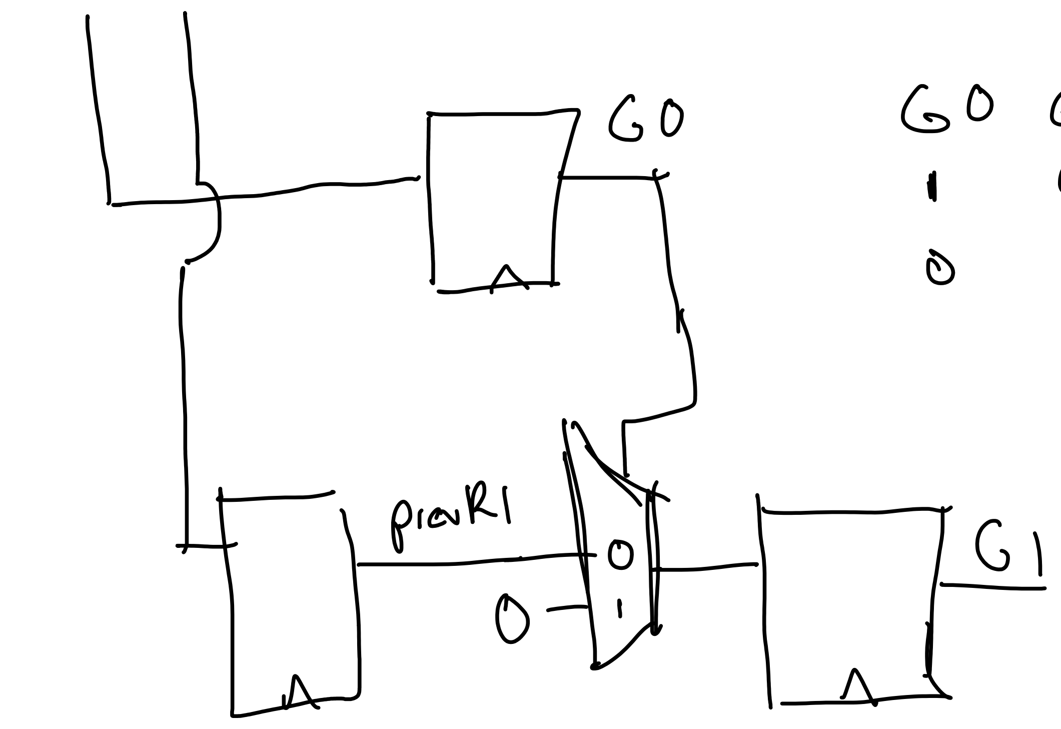
The module will have the following IO:

```
input reset, clock;
input R0, R1;  // request lines
output G0, G1; // grant lines
```

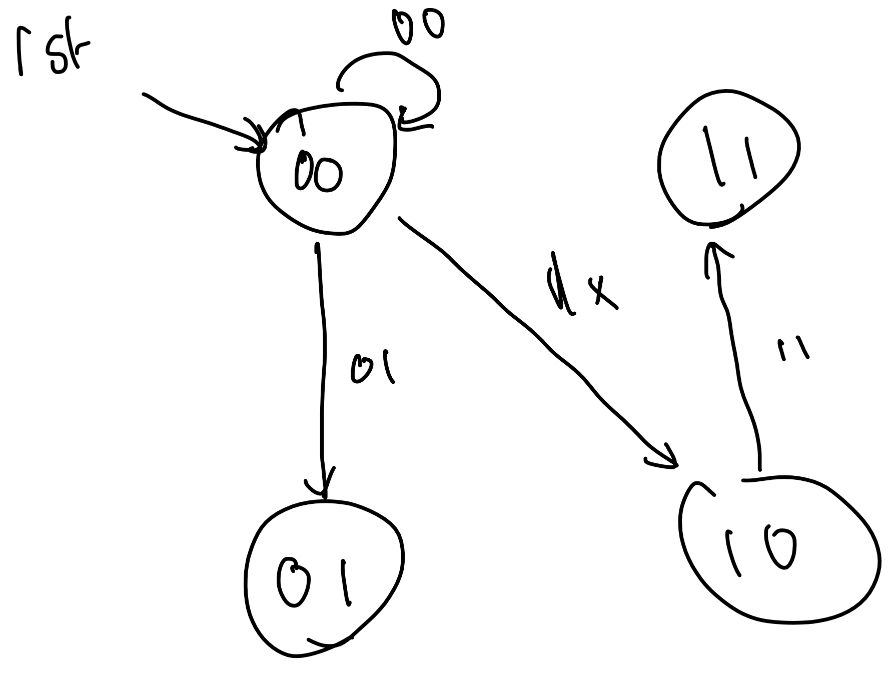
R0	R1	G0	G1
0	0	0	0
1	1	1	0
0	0	0	1
1	0	1	0
0	1	0	1



No FSM design



G0	G1
1	0
0	1



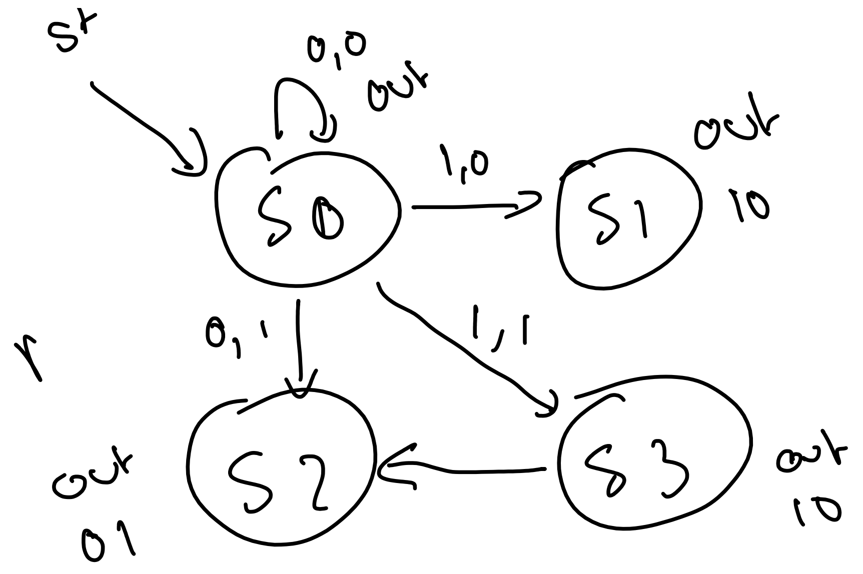
states:

G0 G1

input:

R0 R1

^



inputs R0, R1

out G0, G1

text fixture

```
// Code your design here
module simpleArbiter (
        input clk,
        input rst,
        input r0,
        input r1,
        output reg g0,
        output reg g1
);
        parameter [1:0] S0 = 2'b00;
   parameter [1:0] S1 = 2'b01;
   parameter [1:0] S2 = 2'b10;
   parameter [1:0] S3 = 2'b11;

        reg [1:0] state, next_state;


  always @ (rst) begin
```

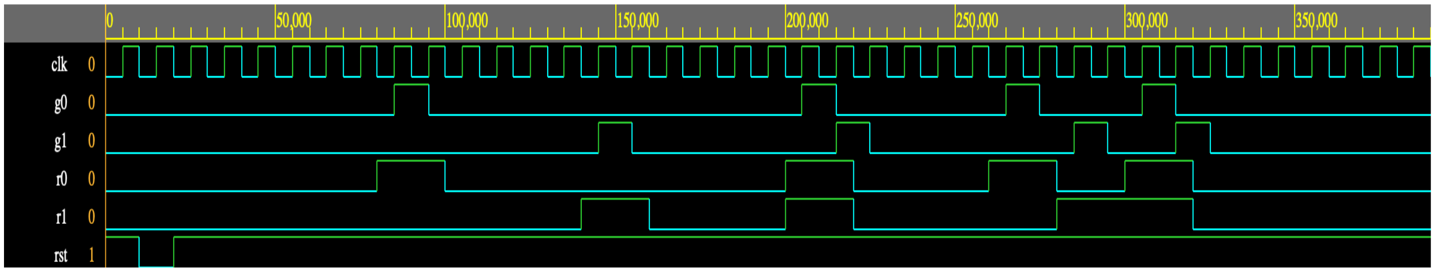
design

```
        state <= S0;
    end


    always @(*) begin
reg [1:0] in;
in = {r0,r1};

        g0 = 0;
        g1 = 0;

        case (state)
            S0: begin
                case (in)
                    2'b00: next_state <= S0;
                    2'b01: next_state <= S2;
                    2'b10: next_state <= S1;
                    2'b11: next_state <= S3;
                endcase
            end
            S1: begin
                g0 = 1;
                next_state <= S0;
            end
            S2: begin
                g1 = 1;
                next_state <= S0;
            end
            S3: begin
                g0 = 1;
                next_state <= S2;
            end
        endcase
    end

    always @(posedge clk) begin
        state <= next_state;
    end
```

```
endmodule

`timescale 1ns/1ps

module simpleArbiter_tb;

   // Testbench signals
   reg clk;
   reg rst;
   reg r0;
   reg r1;
   wire g0;
   wire g1;

   // Instantiate the simpleArbiter module
   simpleArbiter uut (
      .clk(clk),
      .rst(rst),
      .r0(r0),
      .r1(r1),
      .g0(g0),
      .g1(g1)
   );

   // Clock generation: 10ns period (100MHz)
   initial begin
      clk = 0;
      forever #5 clk = ~clk;  // Toggle every 5ns
   end

   // Test sequence
   initial begin
      // Dumping waveform for EDA Playground
      $dumpfile("simpleArbiter_tb.vcd");
      $dumpvars(0, simpleArbiter_tb);

      // Initialize inputs
      rst = 1;
```

```
    r0 = 0;
    r1 = 0;

    // Apply reset for a few cycles
    #10 rst = 0;
    #10 rst = 1;

    // Start test cases

    // Test case 1: No requests
    #20 r0 = 0; r1 = 0;  // No request (idle state)
    #20;

    // Test case 2: Only R0 requests
    #20 r0 = 1; r1 = 0;  // R0 requests
    #20 r0 = 0;          // Clear R0 request
    #20;

    // Test case 3: Only R1 requests
    #20 r0 = 0; r1 = 1;  // R1 requests
    #20 r1 = 0;          // Clear R1 request
    #20;

    // Test case 4: Both R0 and R1 request at the same time
    #20 r0 = 1; r1 = 1;  // Both requests
    #20 r0 = 0; r1 = 0;  // Clear both requests
    #20;

    // Test case 5: R0 and R1 alternate requests
    #20 r0 = 1; r1 = 0;  // R0 requests
    #20 r0 = 0; r1 = 1;  // R1 requests
    #20 r0 = 1; r1 = 1;  // Both request
    #20 r0 = 0; r1 = 0;  // Clear both
    #20;

    // End of simulation
    #50 $finish;
end
```

```
endmodule
```



Slack (setup path): 0.15 ns
Slack (hold path):   0.05 ns

Total cell area: 3456.78
Combinational area: 1200.45
Sequential area: 2256.33