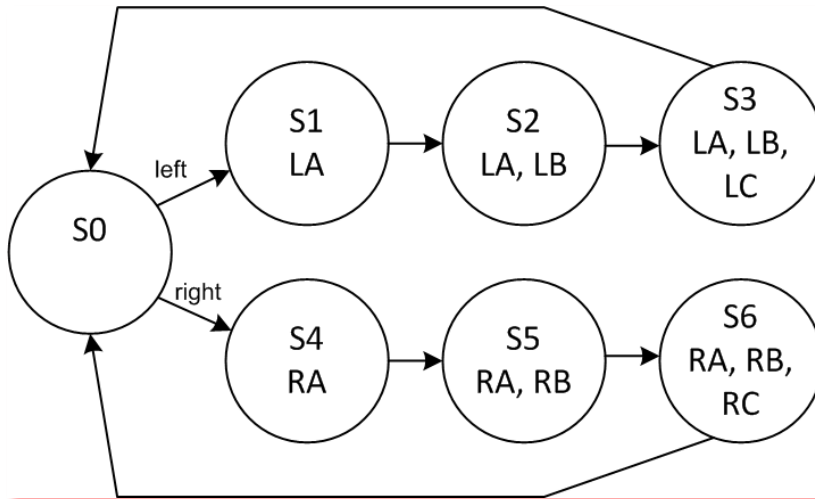


## Lab 3: Finite State Machine Design SOLUTIONS

*Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)*

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.
2. [2] State transition diagram



3. [2] State encoding and Boolean equations for next state and output

Use one-hot encoding

$$NS[0] = S[0] \& \sim\text{left} \& \sim\text{right} \mid S[3] \mid S[6];$$

$$NS[1] = S[0] \& \text{left};$$

$$NS[2] = S[1];$$

$$NS[3] = S[2];$$

$$NS[4] = S[0] \& \text{right};$$

$$NS[5] = S[4];$$

$$NS[6] = S[5];$$

$$LA = S[1] \mid S[2] \mid S[3];$$

$$LB = S[2] \mid S[3];$$

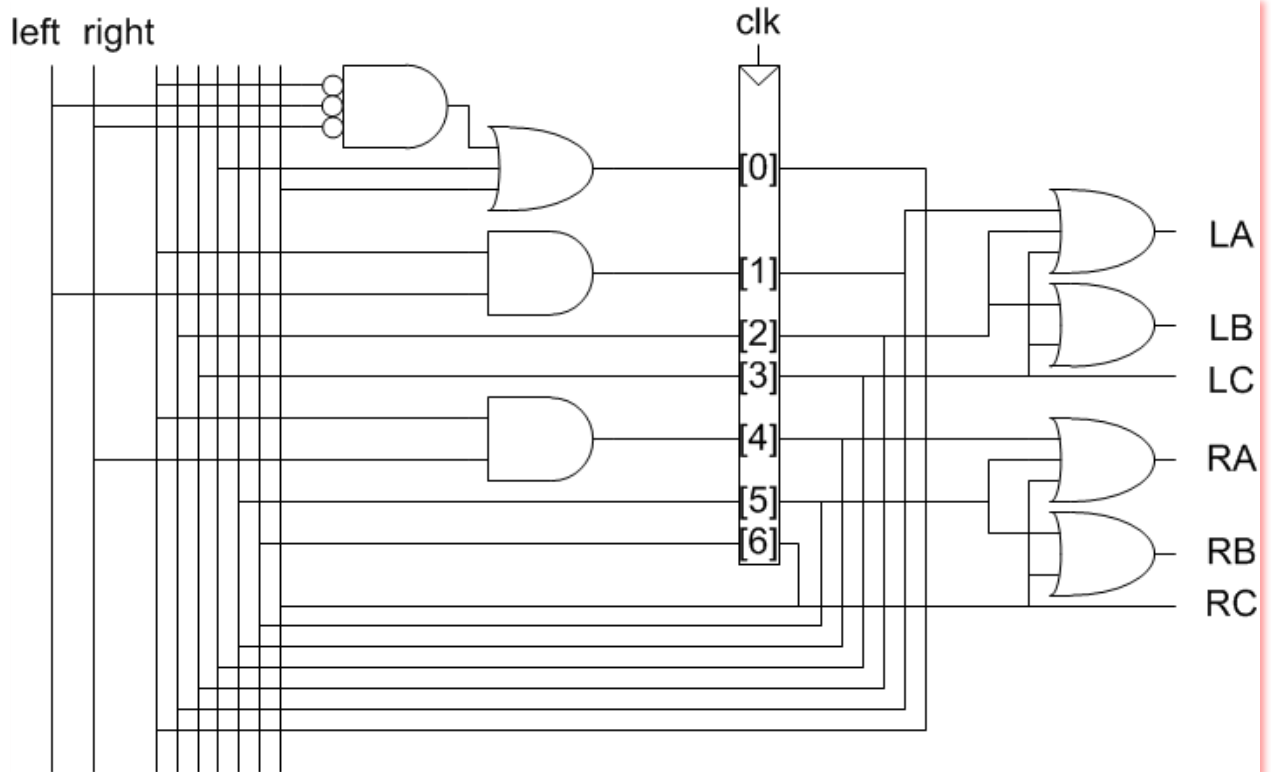
$$LC = S[3];$$

$$RA = S[4] \mid S[5] \mid S[6];$$

$$RB = S[5] \mid S[6];$$

$$RC = S[6];$$

4. [2] Sketch of circuit



##### 5. [2] Structural SystemVerilog code.

Note: question specifically asked for structural SystemVerilog. If the solution uses assign statements, Don't deduct points here but indicate that they should have used structural SystemVerilog.

```

module lab3_dh(input logic clk,
               input logic reset,
               input logic left, right,
               output logic la, lb, lc, ra, rb, rc);

    logic [6:0] state, nextstate;
    logic      leftb, rightb, stay0;

    // state register
    always @(posedge clk or posedge reset)
        if (reset) state <= 7'b0000001;
        else      state <= nextstate;

    // nextstate logic
    not nl(leftb, left);
    not nr(rightb, right);
    and as0(stay0, state[0], leftb, rightb);
    or  os0(nextstate[0], state[3], state[6], stay0);
    and as1(nextstate[1], state[0], left);
    buf bf2(nextstate[2], state[1]);

```

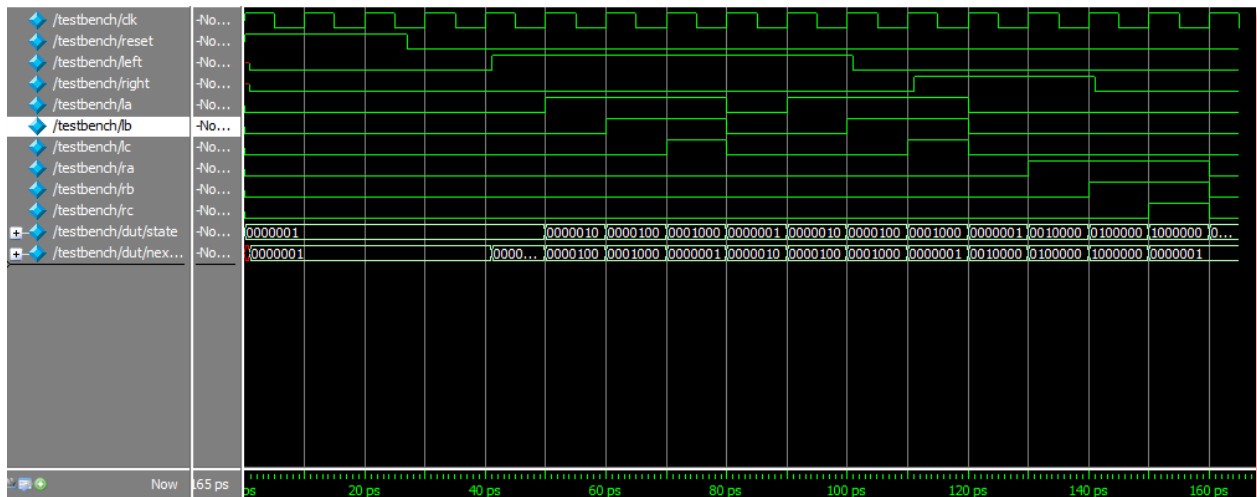
```

buf bf3(nextstate[3], state[2]);
and as4(nextstate[4], state[0], right);
buf bf5(nextstate[5], state[4]);
buf bf6(nextstate[6], state[5]);

// output logic
or ola(la, state[1], state[2], state[3]);
or olb(lb, state[2], state[3]);
buf blc(lc, state[3]);
or ora(ra, state[4], state[5], state[6]);
or orb(rb, state[5], state[6]);
buf brc(rc, state[6]);
endmodule

```

6. [2] Simulation waveforms showing the FSM inputs and outputs. Did it pass the self-checking testbench? **YES**



7. [1] RTL Viewer schematics. Do they match your expectations? **YES**

