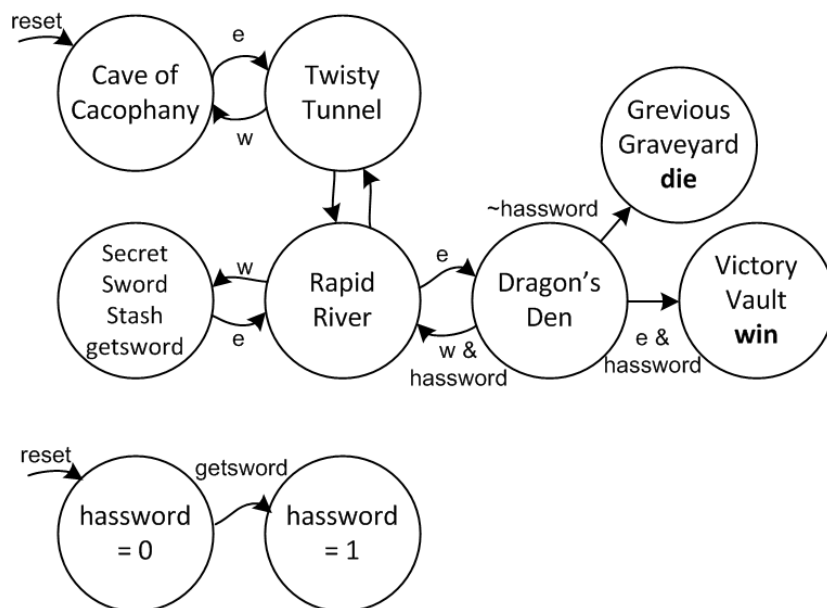


Lab 5: Finite State Machine Design SOLUTIONS

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

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.

- [1] State transition diagrams for the room and sword FSMs.



- [3] Behavioral SystemVerilog code for the system.

```

module lab4_dh(input logic clk, reset,
               input logic n, s, e, w,
               output logic win, die);

    logic hassword, getsword;

    roomfsm room(clk, reset, n, s, e, w, hassword, getsword, win, die);
    swordfsm sword(clk, reset, getsword, hassword);
endmodule

module roomfsm(input logic clk, reset,
               input logic n, s, e, w,
               input logic hassword,
               output logic getsword,
               output logic win, die);

    typedef enum logic [2:0] {CC, TT, RR, SS, DD, GG, VV} roomtype;

    roomtype state, nextstate;

    // state register
    always_ff @(posedge clk, posedge reset)
        if (reset) state <= CC;
        else state <= nextstate;

    // next state logic; takes advantage of assumption that one
    // legal input is asserted on each cycle
    always_comb
        case(state)
            CC: nextstate <= TT;
            TT: if (w) nextstate <= CC;
                else nextstate <= RR;
        endcase
    
```

```

        RR: if (w) nextstate <= SS;
            else nextstate <= DD;
        SS: nextstate <= RR;
        DD: if (~hassword) nextstate <= GG;
            else if (w) nextstate <= RR;
            else nextstate <= VV;
        GG: nextstate <= GG;
        VV: nextstate <= VV;
        default: nextstate <= CC;
    endcase

    // output logic
    assign getsword = (state == SS);
    assign die = (state == GG);
    assign win = (state == VV);
endmodule

module swordfsm(input logic clk, reset,
               input logic getsword,
               output logic hassword);

    logic state, nextstate;

    // state register
    always @(posedge clk, posedge reset)
        if (reset) state <= 0;
        else state <= nextstate;

    assign nextstate = state | getsword;

    assign hassword = state;
endmodule

```

4. [2] A single testbench and test vectors illustrating both the win and die cases.

```

module testbench();
    logic clk, reset;
    logic n, s, e, w, win, die, winexpected, dieexpected;
    logic [31:0] vectornum, errors;
    logic [5:0] testvectors[10000:0];

    // instantiate device under test
    lab4_dh dut(clk, reset, n, s, e, w, win, die);

    // generate clock
    always
    begin
        clk=1; #5; clk=0; #5;
    end

    // at start of test, load vectors
    // and pulse reset
    initial
    begin
        $readmemb("lab4.tv", testvectors);
        vectornum = 0; errors = 0; reset = 1; #22; reset = 0; #70; reset = 1; #10; reset = 0;
    end

    // apply test vectors on rising edge of clk
    always @(posedge clk)
    begin
        #1; {n, s, e, w, winexpected, dieexpected} = testvectors[vectornum];
    end

    // check results on falling edge of clk
    always @(negedge clk)
    if (~reset) begin // skip during reset
        if (win != winexpected || die != dieexpected) begin // check result
            $display("Error: inputs = %b", {n, s, e, w});
            $display(" state = %b", dut.room.state);
            $display(" outputs = %b %b (%b %b expected)",

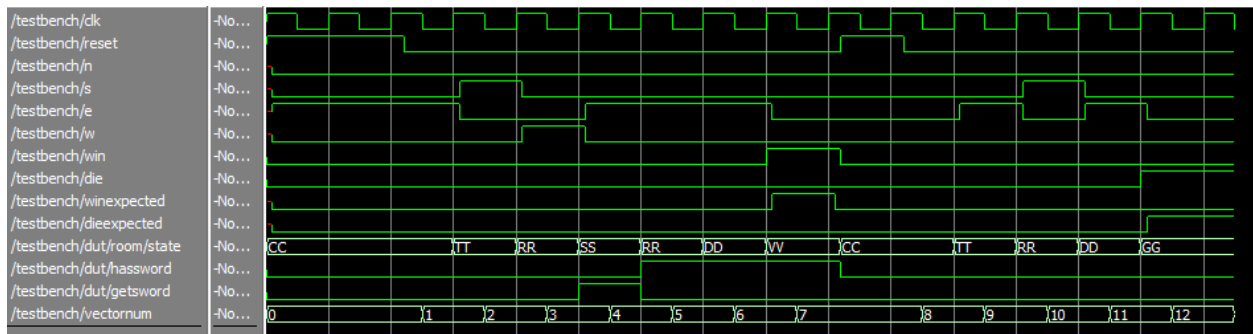
```

```

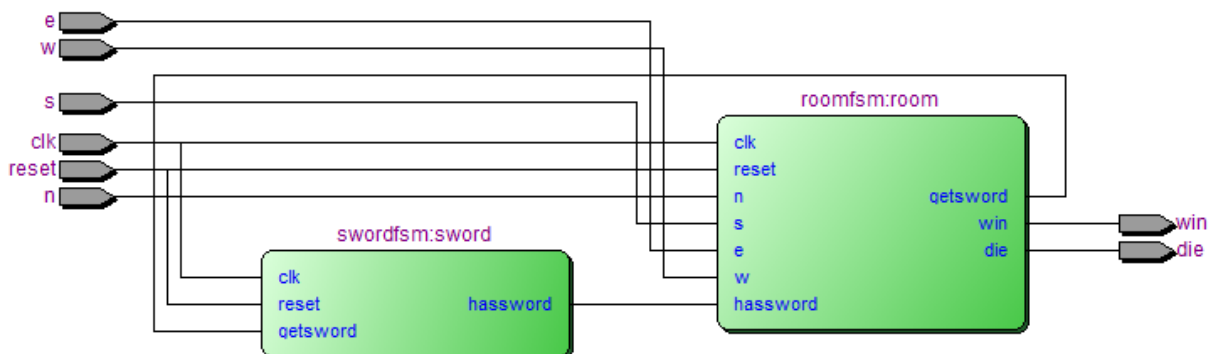
        win, die, winexpected, dieexpected);
    errors = errors + 1;
end
vectornum = vectornum + 1;
if (testvectors[vectornum] === 6'bx) begin
    $display("%d tests completed with %d errors", vectornum, errors);
    $stop;
end
end
endmodule

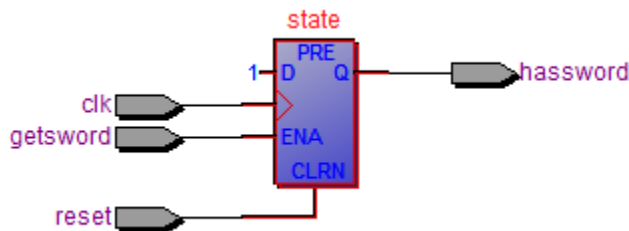
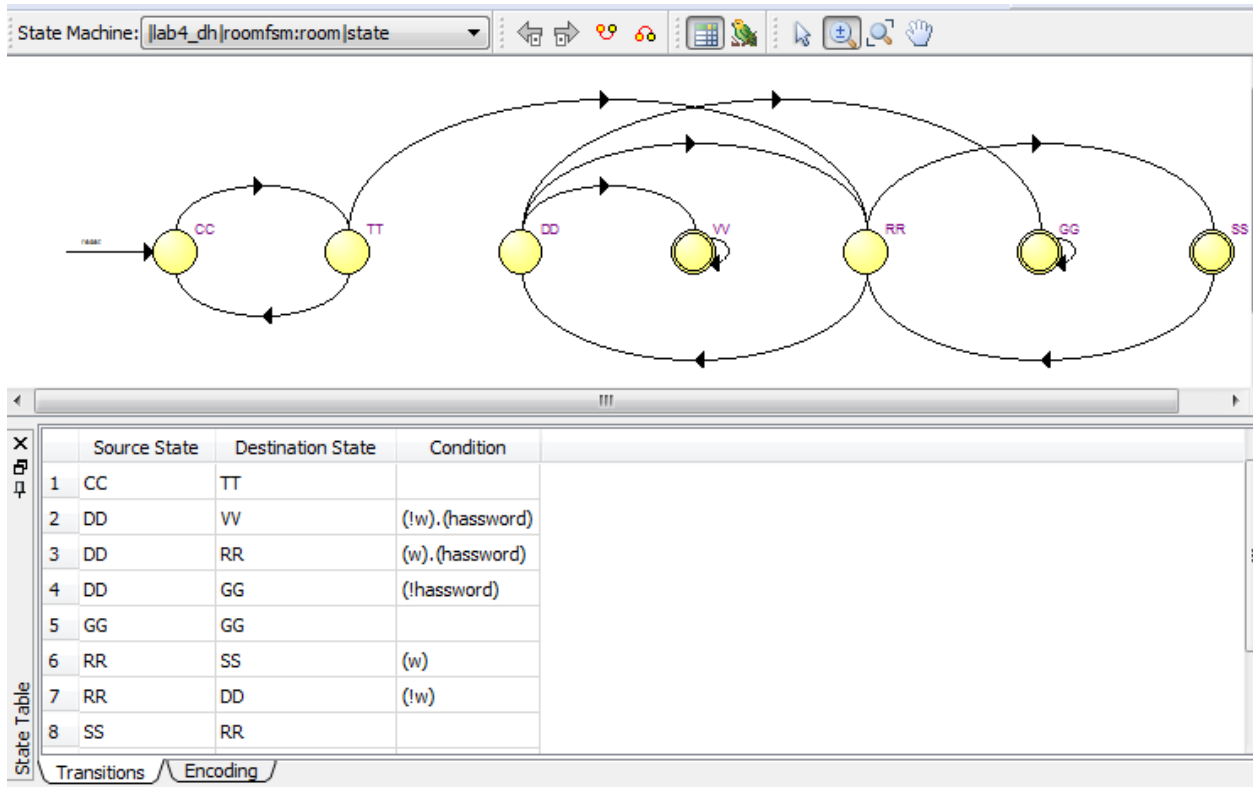
```

5. [1] Simulation waveforms including the inputs, outputs, and current room. Do they pass your testbench and match expectations? **YES**



6. [2] RTL Viewer schematics (including each FSM). Do they match your expectation? **YES**





7. [1 point, 2 for something very elaborate] EXTRA CREDIT: It is a little known fact that the Twisty Tunnel is located beneath Hoch and that by heading north one can reach the dormitories. Extend your adventure game with more interesting rooms or objects. There will be a prize for the most interesting working enhancement!

If you have suggestions for further improvements of this lab, you're welcome to include them at the end of your lab.

Spjut's alternate solution

Behavioral Verlog for system

```
module room(input logic clk, reset,
            input logic [3:0] direction, // mapping N--0001[0], E--0010[1], S--0100[2], W--1000[3]
            input logic hassword,
```

```

        output logic getsword, youwon, youredead);

typedef enum logic [3:0] {CC, TT, RR, SSS, DD, GG, VV} statetype; // left room for expansion to more states
statetype state, nextstate;

// state register
always_ff @(posedge clk, posedge reset)
    if (reset) state <= CC;
    else state <= nextstate;

// next state logic
always_comb
    case (state)
        CC: if (direction[1]) nextstate = TT;
            else nextstate = CC; // the else is not strictly needed according to lab

        TT: if (direction[2]) nextstate = RR;
            else if (direction[3]) nextstate = CC;
            else nextstate = TT; // the else is not strictly needed according to lab

        RR: if (direction[0]) nextstate = TT;
            else if (direction[1]) nextstate = DD;
            else if (direction[3]) nextstate = SSS;
            else nextstate = RR; // the else is not strictly needed according to lab

        SSS: if (direction[1]) nextstate = RR;
            else nextstate = SSS; // the else is not strictly needed according to lab

        DD: if (~havesword) nextstate = GG;
            else if (direction[1]) nextstate = VV;
            else if (direction[3]) nextstate = RR;
            else nextstate = DD; // the else is not strictly needed according to lab

        GG: nextstate = GG;

        VV: nextstate = VV;

        default: nextstate = CC;
    endcase

// output logic
assign youredead = (state == GG);
assign youwon = (state == VV);
assign getsword = (state == SSS);

endmodule

module sword(input logic clk, reset,
             input logic getsword, // inss is in Secret Sword Stash
             output logic hasesword);

    logic state, nextstate;

    // state register

```

```

always_ff @(posedge clk, posedge reset)
    if (reset) state <= 0;    // 0 is don't have sword state
    else    state <= nextstate;

// next state logic
always_comb
    if (getsword) nextstate = 1;
    else nextstate = state;

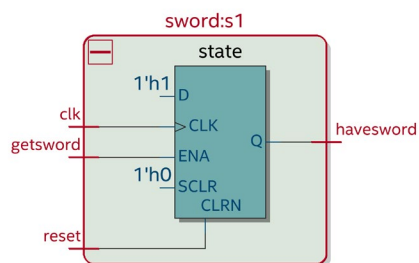
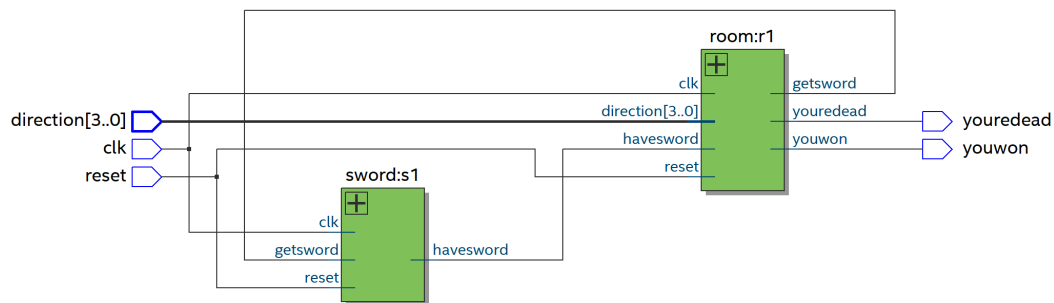
// output logic
assign havesword = state;
endmodule

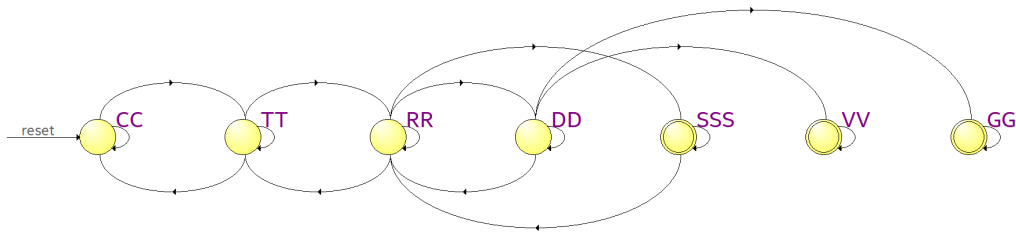
module adventure(input logic clk, reset,
                input logic [3:0] direction, // mapping N--0001[0], E--0010[1], S--0100[2], W--
                1000[3]
                output logic youwon, youredead);

    logic getsword, havesword;

    sword s1(clk, reset, getsword, havesword);
    room r1(clk, reset, direction, havesword, getsword, youwon, youredead);
endmodule

```





Behavioral Verilog for testbench

```

module testbench();
  logic clk, reset;
  logic [3:0] direction;
  logic youwon, youredead, youwonexpected, youredeadexpected;
  logic [31:0] vectornum, errors;
  logic [5:0] testvectors[10000:0];

  // instantiate device under test
  adventure dut(clk, reset, direction, youwon, youredead);

  // generate clock
  always
  begin
    clk=1; #5; clk=0; #5;
  end

  // at start of test, load vectors
  // and pulse reset
  initial
  begin
    $readmemb("adventure.tv", testvectors);
    vectornum = 0; errors = 0; reset = 1; #22; reset = 0; #100; reset = 1; #10; reset = 0; //fix reset
  end

  // apply test vectors on rising edge of clk
  always @(posedge clk)
  begin
    #1; {direction, youwonexpected, youredeadexpected} = testvectors[vectornum];
  end

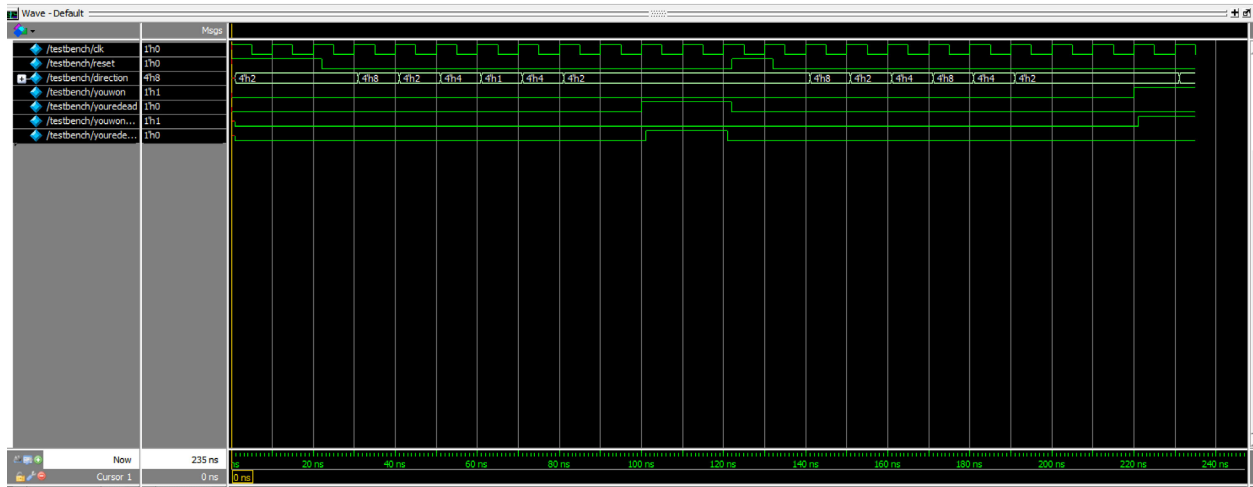
  // check results on falling edge of clk
  always @(negedge clk)
  if (~reset) begin // skip during reset
    $display("To travel in %b direction. Currently in room %s. Sword=%b Dead=%b, Won=%b",
      direction, dut.r1.state.name, dut.r1.havesword, youredead, youwon);
    if (youwon !== youwonexpected || youredead !== youredeadexpected) begin // check result
      $display("Error: inputs = %b", direction);
      $display(" state = %s", dut.r1.state.name);
      $display(" outputs = %b %b (%b %b expected)",
        youwon, youredead, youwonexpected, youredeadexpected);
      errors = errors + 1;
    end
  end
end

```

```

end
vectornum = vectornum + 1;
if (testvectors[vectornum] === 6'bx) begin
$display("%d tests completed with %d errors", vectornum,
errors);
$stop;
end
end
endmodule

```



```

run 500
# To travel in 0010 direction. Currently in room CC. Sword=0 Dead=0, Won=0
# To travel in 1000 direction. Currently in room TT. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room CC. Sword=0 Dead=0, Won=0
# To travel in 0100 direction. Currently in room TT. Sword=0 Dead=0, Won=0
# To travel in 0001 direction. Currently in room RR. Sword=0 Dead=0, Won=0
# To travel in 0100 direction. Currently in room TT. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room RR. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room DD. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room GG. Sword=0 Dead=1, Won=0
# To travel in 0010 direction. Currently in room GG. Sword=0 Dead=1, Won=0
# To travel in 0010 direction. Currently in room CC. Sword=0 Dead=0, Won=0
# To travel in 1000 direction. Currently in room TT. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room CC. Sword=0 Dead=0, Won=0
# To travel in 0100 direction. Currently in room TT. Sword=0 Dead=0, Won=0
# To travel in 1000 direction. Currently in room RR. Sword=0 Dead=0, Won=0
# To travel in 0100 direction. Currently in room SSS. Sword=0 Dead=0, Won=0
# To travel in 0010 direction. Currently in room SSS. Sword=1 Dead=0, Won=0
# To travel in 0010 direction. Currently in room RR. Sword=1 Dead=0, Won=0
# To travel in 0010 direction. Currently in room DD. Sword=1 Dead=0, Won=0
# To travel in 0010 direction. Currently in room VV. Sword=1 Dead=0, Won=1
# To travel in 1000 direction. Currently in room VV. Sword=1 Dead=0, Won=1
#      21 tests completed with          0 errors
#      **      Note:      $stop          :      C:/Users/spjut/Documents/E85
#      F17/Lab_4_sol_res/testbench.sv(47)
#      Time: 235 ns Iteration: 1 Instance: /testbench
#      Break in Module testbench at      C:/Users/spjut/Documents/E85
#      F17/Lab_4_sol_res/testbench.sv line 47

```