Problem 1: [10 points]
Filename: hw0.sim

```
Chronologic VCS simulator copyright 1991-2022
Contains Synopsys proprietary information.
Compiler version T-2022.06_Full64; Runtime version T-2022.06_Full64;  Aug 30 ...
```

Line length of 87 (max is 80)

```
state: SLEEP alarm: 0 good: 0 happy: 0 fun_n 1
state: SLEEP alarm: 1 good: 0 happy: 0 fun_n 1
state: STUDY alarm: 0 good: 0 happy: 1 fun_n 1
state: STUDY alarm: 0 good: 1 happy: 0 fun_n 0
state: PLAY alarm: 0 good: 0 happy: 1 fun_n 0
state: STUDY alarm: 1 good: 0 happy: 0 fun_n 0
state: PLAY alarm: 1 good: 0 happy: 1 fun_n 0
$finish called from file "hw0.sv", line 90.
$finish at simulation time                   19
           V C S   S i m u l a t i o n   R e p o r t
Time: 19
CPU Time:      0.690 seconds;       Data structure size:   0.0Mb
Wed Aug 30 11:14:55 2023
```

Problem 1: [10 points]
Filename: hw0.sv

```
//not a complete test bench since we didn't test every possible input
//combination at every state(ie not exhuastiely tested)
`default_nettype none

module FSM
  (input  logic alarm, good, clock, reset,
   output logic happy, fun_n);

  enum logic [1:0] {SLEEP = 2'b00, STUDY = 2'b01, PLAY = 2'b10}state,nextState;

  //next state logic
  always_comb begin
    case (state)
      SLEEP: begin
        nextState = alarm ? STUDY : SLEEP;
      end
      STUDY: begin
        if (alarm) begin
          nextState = PLAY;
        end else if (good) begin
          nextState = PLAY;
        end else begin
          nextState = STUDY;
        end
      end
      PLAY: begin
        nextState = STUDY;
      end
    endcase
  end

  //output logic
  always_comb begin
    fun_n = 1'b1;
    happy = 1'b0;
    case (state)
      SLEEP: begin
      end
      STUDY: begin
        if (alarm) begin
          fun_n = 1'b0;
        end else if (good) begin
          fun_n = 1'b0;
        end else begin
          happy = 1'b1;
        end
      end
      PLAY: begin
        fun_n = 1'b0;
        happy = 1'b1;
      end
    endcase
  end

  always_ff @(posedge clock, posedge reset) begin
    if (reset) begin
      state <= SLEEP;
    end else begin
      state <= nextState;
    end
  end
endmodule: FSM

module testBench();
  logic clock, reset, alarm, good, happy, fun_n;

  FSM fsm(.*);

  initial begin
    clock = 1'b0;
    forever #(5) clock = ~clock;
  end

  initial begin
    reset <= 1'b0;
    #(1) reset <= 1'b1;
         alarm <= 1'b0;
         good <= 1'b0;
    $monitor("state: %s alarm: %d good: %d happy: %d fun_n %d", fsm.state,
              alarm, good, happy, fun_n);
    @(posedge clock);
    reset <= 1'b0;
    @(posedge clock); //self loop
    @(posedge clock);
    alarm <= 1'b1; //go to Study
    @(posedge clock);
    alarm <= 1'b0; //self loop
    @(posedge clock);
    good <= 1'b1; //go to play on right
    @(posedge clock);
    //back to Study, resetting good to 0 unnecessary just for clarity
    good <= 1'b0;
    @(posedge clock);
    alarm <= 1'b1; //back to Play on left path
    @(posedge clock);
    @(posedge clock) $finish;
  end
endmodule: testBench
```